Quantifying Patient-Specific Plantar Soft Tissue Stiffness using Magnetic Resonance Imaging and a Hydraulic Loading Device

Evan D. Williams

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering

University of Washington

2015

Committee:

William R. Ledoux

David R. Haynor

Per G. Reinhall

Program Authorized to Offer Degree:

Department of Mechanical Engineering

University of Washington

Abstract

Quantifying Patient-Specific Plantar Soft Tissue Stiffness using Magnetic Resonance Imaging and a Hydraulic Loading Device

Chair of the Supervisory Committee:

Affiliate Associate Professor William R. Ledoux

Departments of Mechanical Engineering and Orthopaedics & Sports Medicine

Diabetes mellitus is a prevalent health concern in the United States that has many adverse symptoms, one of which is increased plantar soft tissue stiffness. This change in the mechanical properties of foot tissue has been hypothesized to lead to large internal peak stresses that could cause ulcers. Ulcers can lead to foot tissue infections and eventually amputations. Of all non-traumatic lower limb amputations for subjects greater than 20 years old, 60% follow this pathology. It is important to increase our understanding of the stress distribution in the foot under various conditions, such as loading rate/magnitude and tissue healthiness, in order to improve computational foot models and predictions for *in vivo* subjects with and without diabetes. This study aims to achieve this by combining three-dimensional (3D) internal tissue deformation imaging (gated MRI) with a non-metallic hydraulic loading device (the hydraulic plantar soft tissue reducer – HyPSTR).

There were two major aspects of the HyPSTR project. The first was part to design, build, and verify an MRI-compatible hydraulic device to load a subject's foot in a controlled manner within

the MRI's magnetic core. Preliminary testing was performed with load cells and rigid aluminum jigs to ensure the motor and piston design was capable of producing physiologic loading conditions. The second aspect was to use the HyPSTR with ultrasound (pilot testing) and gated MRI protocols to capture deformations under dynamic forces. The data collected were processed to generate stiffness curves and subsequently used with inverse finite element analysis (FEA). The inverse FEA modeling was done by Vara Isvilanonda for his PhD dissertation in 2015.

The construction and verification of the HyPSTR was a significant undertaking and highlighted several issues with the hydraulic performance and jig used to secure the subjects' legs when under load. Primarily, the loading platen was only able to operate at a maximum of 0.2 Hz, which is slower than the frequency content of gait (the goal rates were 1 Hz to 6 Hz). This problem was caused by a water hammer (pressure ringing) effect due to the rapid momentum changes when the hydraulics reversed direction (load vs unload movement). Additionally, the load magnitudes were only approximately 25% of subject body weight. Discomfort on the dorsal aspect of the foot, at the interface with an ankle foot orthosis (AFO), prevented testing with larger forces. Even with these problems, the study tested one subject with diabetes and one without, and demonstrated that diabetic plantar soft tissue was stiffer than healthy tissue. These results are only for two subjects, but they provide cursory proof that the HyPSTR can measure patient-specific stiffnesses. Furthermore, the procedures and devices include all of the functionality needed to make *in vivo*, patient-specific, dynamic, 3D deformation measurements. The next challenge is not to add more components, but rather to improve the control and range of what already exists.

Table of Contents

List of Figures

List of Tables

Abbreviations

1D – one-dimensional

2D – two-dimensional

3D – three-dimensional

AFO – ankle foot orthosis

CDAS – control and data acquisition system

DAQ – data acquisition board

DM - diabetes mellitus

FE/FEA – finite element or finite element analysis

HR – heart rate

HyPSTR – hydraulic plantar soft tissue reducer

HyPSTR_1 – hydraulic plantar soft tissue reducer version 1 (2010-2012)

HyPSTR_2 – hydraulic plantar soft tissue reducer version 2 (2012-current)

IRB – internal review board

LVDT – linear variable differential transformer

MR/MRI – magnetic resonance or magnetic resonance imaging

PNS – peripheral nerve stimulation

PPU – peripheral pulse unit

RAM – random access memory

RFOV – rectangular field of view

RMSE – root mean square error

SAR – specific absorption rate

TFE – turbo field echo

TTL – transistor-transistor logic (pulse signal)

US – ultrasound

UW – the University of Washington

VA – Veterans Affairs

VI – virtual instrument

Acknowledgements

Thank you to everyone who provided encouragement and support throughout the course of this project. It has been a substantial learning experience that breaks the laws of Physics, where I have received twice as much potential as I put into the HyPSTR. Specifically, thank you Bil Ledoux, for being an adviser, teammate, and friend every step of the way. My contributions to this project are the product of two brains, and the summation of many more.

Thank you Mike Fassbind, Vara Isvilanonda, and Matt Kindig for the technical engineering assistance and suggestions. I also owe much appreciation to Mike Stebbins for continuing to help with his original design (HyPSTR_1) even after graduating. Mike's meticulous notes were, and will continue to be, invaluable to any related work.

All of this would not have been possible without the funding from the U.S. Department of Veterans Affairs VA RR&D Grant A6973R. Thank you to the VA, and all who believe that this work can lead to valuable progress in the field of limb loss prevention, especially in regard to diabetes.

Chapter 1: Overview

The motivation behind this thesis was to identify the primary limitations with the first version of the Hydraulic Plantar Soft Tissue Reducer (HyPSTR_1), overcome them, and add the necessary functionality to achieve the Specific Aims outlined in the proposal (VA RR&D Grant A6973R, PI: William R. Ledoux, PhD). Michael J. Stebbins, MS built the HyPSTR_1 as his Master's thesis topic; it was designed to measure the material properties of the plantar soft tissue, with the long term goal of contributing to the prevention of ulcer formation and ensuing limb loss for people with diabetes. One path to achieving this goal is to build better mechanical finite element analysis (FEA) models of feet by including patient specific material properties and anatomy obtained from magnetic resonance imaging (MRI). The crux of the project is to create a machine of non-metallic components to simulate walking force magnitudes and loading rates for a person lying in an MR core. The HyPSTR controls this motion while also recording the pressure, from which plantar force is derived, while the MRI scan provides soft tissue deformations. By conducting an inverse FEA, material properties can be calculated for skin, fat and muscle in the feet of living subjects. (The FE modeling was conducted as part of Vara Isvilanonda's PhD thesis.)

In its completed state, the HyPSTR_1 (Figure 1.1) provided an excellent, and well documented,[1] base for future improvements. The hydraulic system was already particularly good at producing reliable displacements for both unloaded and loaded applications. Frequency targets were matched to within 0.3% for 0.1 Hz tests and displacement targets were within 2.8% for up to 9 mm of travel (351.8 N) against a silicone backstop, braced by an 8020 aluminum frame. Loading curve profiles were customizable and sine and triangle waves were tested thoroughly.

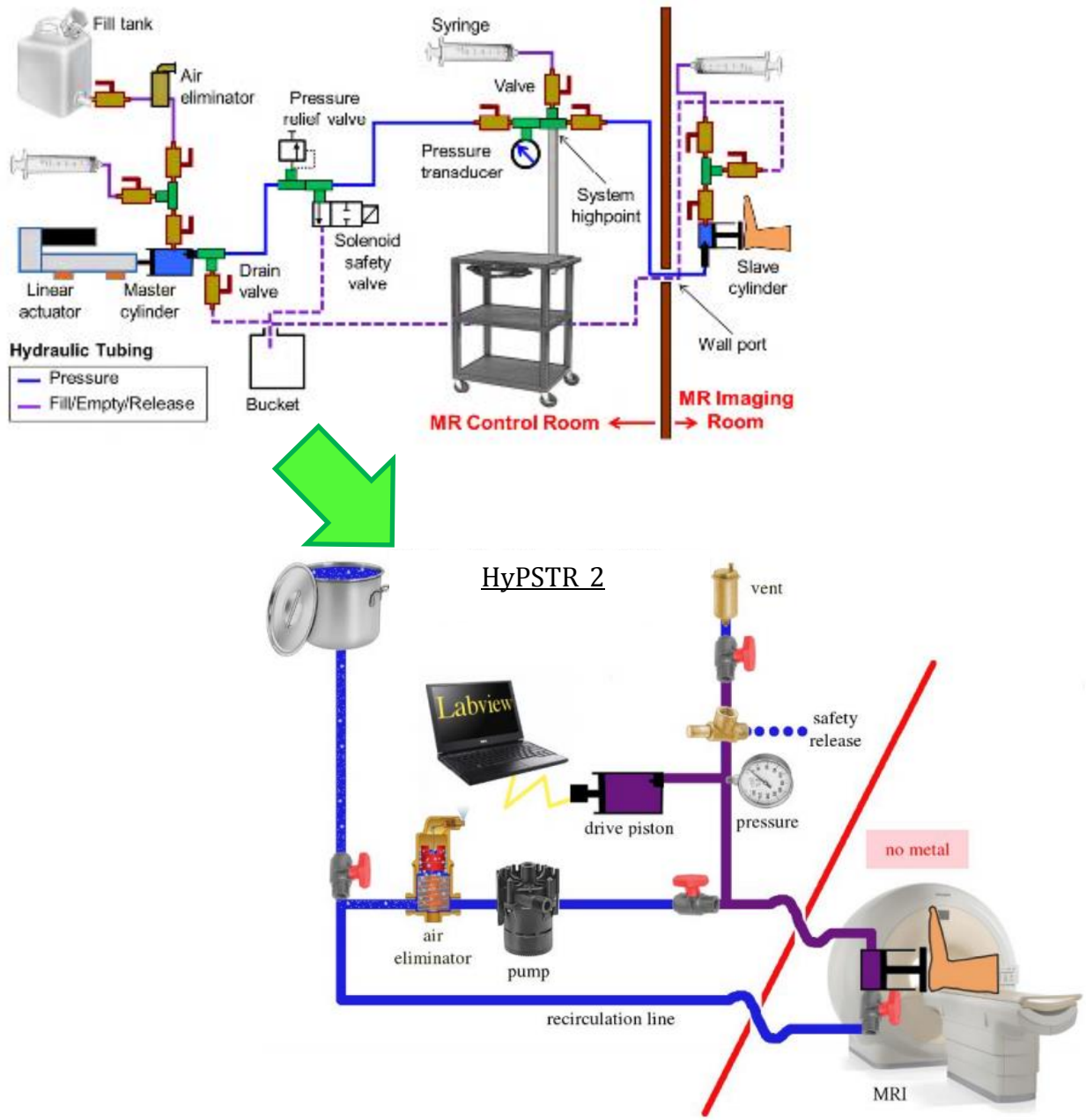


Figure 1.1. The HyPSTR_1 as developed by Stebbins[1] (top). In addition to a simpler schematic, the HyPSTR_2 (bottom) underwent some major structural changes in the plumbing layout, primarily to allow for a recirculation line and pump to remove dissolved gasses. The electronics control boxes were left in the same position on the cart, as were many other critical components.

Under all tested displacement shapes, the HyPSTR_1 actuator was able to produce forces that exceeded the requirements for the study, though there were sometimes performance issues due to the loaded object/subject not being adequately secured. If the forces reached unsafe levels, a

system with five levels of mechanical or electrical redundancy safeguarded against potential injury to human subjects and instrumentation. MRI compatibility and electromagnetic noise reduction design parameters were incorporated into every subsection of the system. The HyPSTR_1 successfully convinced the research team that the concept was worthy of further research and development.

Stebbins suggested some important areas in need of improvement[1] for the second version of the MRI compatible plantar loading device (HyPSTR_2). The system performance verification tests needed to be redone for 0.2 Hz motion. A better mathematical procedure for converting hydraulic pressure to force was advised. Using oil/glycol instead of water as the energy transfer medium was suggested. Hydraulic pumps and electronic valves should be investigated to potentially solve loading rate limitations. The addition of shear forces could be explored for more accurate gait simulation. Independent instruments, other than the MRI scanner and HyPSTR, should be considered for measuring platen displacement and load during the MRI scan; this would confirm accurate test values in real-time, allowing for quick adjustment before running costly imaging procedures. And lastly, more research into the gated MRI protocols was necessary to improve image quality and/or the number of data points on stress/strain curves.

In conjunction with Stebbins' suggestions, a prioritized list directed our efforts for the HyPSTR_2. The importance of some of the following items was discovered during the first several weeks of becoming familiar with the system. The parenthetical text refers to the location within this thesis.

1. (Chapter 4) Reorganize and check analog signal filter circuitry
2. (Chapter 4) Prevent air bubble formation in the high pressure tubes during tests
3. (Chapter 4) Repeat verification tests for 0.2 Hz
4. (Chapter 4) Reduce skeletal shifting by securing the lower leg better
5. (Chapter 3) Repeat ultrasound pilot tests

6. (Chapter 3) Track platen movement during an MRI scan
7. (Chapter 4) Assess MRI timing accuracy
8. (Chapter 4) Improve MRI quality and/or add more phases to gated MRI
9. (Chapter 3) Standardize stiffness calculation procedures
10. (Appendix 3) Update HyPSTR preparation and MRI scan protocol for test day
11. (Chapter 5) Brainstorm solutions for remaining limitations

As the HyPSTR_1 project had collected nearly enough data for publication, the first tasks were focused on filling the remaining gaps for the "verification" paper. Demonstrating the reliability and functionality of the HyPSTR turned out to be a formidable, but necessary, task before recruiting subjects for the FEA model data collection and preparing a second paper, informally called the "imaging" paper. These two manuscripts are presented as Chapters 2 and 3 in this thesis as they are the most complete and organized representation of the work done. Following these, more detailed explanations, specific procedures, notable observations, successes/failures, suggestions, and lessons learned are given in the Modifications and Additions chapter (i.e., Chapter 4).

Aside from the convincing diabetes statistics discussed in Stebbin's thesis,[1] and summarized in Chapter 2, I am personally motivated to work on any project that promotes human motion and activity as a solution toward health and happiness. The foot is of special interest because of my intrigue with balance and the remarkable mechanics of the structure itself. The familiar Leonardo Da Vinci quote says it best, "The human foot is a masterpiece of engineering and a work of art."

References

1. Stebbins MJ. Obtaining Material Properties of the Plantar Soft Tissue for a Patient-Specific Finite Element Model of the Foot. *University of Washington*. 2012; Mechanical Engineering Department.

Chapter 2: Design and Verification of a Gated MRI Technique to Determine Mechanical Properties of Plantar Soft Tissue

Evan D. Williams,[1, 2] Michael J. Stebbins,[1,2] Peter R. Cavanagh,[2,3] David R. Haynor,[4] Baocheng Chu,[4] Michael J. Fassbind,[1] Vara Isvilanonda,[1, 2] William R. Ledoux,[1, 2, 3]

[1]RR&D Center of Excellence for Limb Loss Prevention and Prosthetic Engineering,
VA Puget Sound, Seattle, WA, 98108;

[2]Department of Mechanical Engineering,
[3]Department of Orthopaedics & Sports Medicine,
[4]Department of Radiology
University of Washington, Seattle, WA, 98195

Corresponding author:
William R. Ledoux, PhD
ms 151, VA Puget Sound
1660 S. Columbian Way
Seattle, WA 98108
(206) 768-5347
(206) 764-2127 (f)
wrledoux@uw.edu

Keywords: heel pad; diabetes; MRI-compatible; mechanical properties

Abstract

Changes in the mechanical properties of the plantar soft tissue in people with diabetes may contribute to the formation of plantar ulcers. Such ulcers have been shown to be in the causal pathway for lower extremity amputation. The hydraulic plantar soft tissue reducer (HyPSTR) was designed to measure in-vivo, rate-dependent plantar soft tissue compressive force and three-dimensional deformations to help understand, predict, and prevent ulcer formation. These patient-specific values can then be used in an inverse finite element analysis to determine tissue moduli, and subsequently used in a foot model to show regions of high stress under a wide variety of loading conditions. The HyPSTR uses an actuator to drive a magnetic resonance imaging (MRI) compatible hydraulic loading platform. Pressure and actuator position were synchronized with MRI acquisition. Achievable loading rates were slower than those found in normal walking because of a water-hammer effect (pressure wave ringing) in the hydraulic system when the actuator direction was changed rapidly. The subsequent verification tests were therefore performed at 0.2 Hz. The unloaded displacement accuracy of the system was within 0.31%. Compliance, presumably in the system's plastic components, caused a displacement loss of 5.7 mm during a 20 mm actuator test at 1354 N. This was accounted for with a target to actual calibration curve. The positional accuracy of the HyPSTR during loaded displacement verification tests from 3 to 9 mm against a silicone backstop was 95.9% with a precision of 98.7%. The HyPSTR generated minimal artifact in the MRI scanner. Careful analysis of the synchronization of the HyPSTR and the MRI scanner was performed. With some limitations, the HyPSTR provided key functionality in measuring dynamic, patient-specific plantar soft tissue mechanical properties.

Introduction

Diabetes Mellitus (DM) is a major public health problem in the United States with 9.3% of the population having the disease in 2014. A diagnosis of DM is present in 60% of patients undergoing non-traumatic lower limb amputations.[1] The disease has been shown to increase the modulus of plantar soft tissue in cadaveric samples,[2] and this affects the ability of the foot to absorb and distribute energy from loading. As a result, peak stresses in plantar tissue near bony prominences can increase fourfold or more[3] and are thought to be the location of initial ulcer formation.[3, 4] There is a need to improve current methods for predicting internal stress due to increases in plantar tissue stiffness. This can be accomplished by advancing the technology for measuring patient-specific soft tissue material properties. When combined with finite element analysis (FEA), this will allow improved understanding of foot mechanics and the development of patient-specific treatment options.

Ultrasound and fluoroscopic imaging have been previously used in plantar tissue analyses.[5-7] Cavanagh[6] mounted an ultrasound probe flush with an acrylic plate that measured skin-to-metatarsal head displacements upon foot contact. These measurements were one-dimensional (1D) and load data were collected in a subsequent test using separate equipment. Erdemir et al.[7] utilized an ultrasound indenter in conjunction with dynamic loading and inverse FEA to develop hyperelastic material properties of the plantar fat pad. The device was capable of controlling the loading rate and recording patient-specific thickness, but the study approximated the fat pad as axi-symmetric with idealized boundary conditions and considered only a single dimension. Gefen et al.[5] used fluoroscopy and a force plate for simultaneous load and deformation collection, but this work was also limited to a single-dimension and exposed the patient to ionizing radiation. Although the above studies are the foundation for subsequent work, it is

difficult to extrapolate their results to the generation of three-dimensional (3D) patient-specific FEA models.

Progressing beyond ultrasound and fluoroscopy, plantar tissue properties can also be determined by magnetic resonance imaging (MRI), in conjunction with various non-metallic loading instruments. Petre et al.[8] developed a device that applied compressive loads to feet by holding hydraulic pressures for 3.5 minute static MRI scans. The resulting material property estimates could potentially be improved by including strain rate dependent effects, which might be significant.[2] Gefen et al.[9] employed an MRI-compatible spherical indentation device to measure load vs. displacement of the plantar soft tissue between metatarsal heads. In addition to also being a static test, the less than 2N loads were much smaller than would occur during normal loading. In addition, the skin, fat, and other subcutaneous tissues were lumped into one generic soft tissue composite. Another indenter study by Brown et al.[10] addressed the viscoelastic dependency of tissue with another MRI-compatible piezoelectric device that accounted for strain and strain rate. The system has been used with tissue samples, but it is not understood if this technique can be applied to living subjects. Weaver et al.[11] performed dynamic MR elastography and determined that shear moduli differed with applied load. However, subjects voluntarily controlled the force and only two loading points were studied.

The purpose of this study was to develop a plantar soft tissue testing apparatus that allows for dynamic plantar loading at the hindfoot or forefoot in living subjects at physiologic force during gait while simultaneously capturing 3D soft tissue deformation. Subsequent FEA models will incorporate patient-specific anatomy and differentiation between soft tissue types. Hereafter, we report on the development and validation of the hydraulic plantar soft tissue reducer (HyPSTR),

which loads the foot with a hydraulic system and records 3D tissue deformations with gated MRI.

Methods

System design

The HyPSTR was designed to acquire gated MR images from multiple phases of internal tissue deformation during displacement-controlled cyclic loading of the plantar aspect of the foot. Sixteen phases were used in the experiments reported here, a single-acting, master/slave hydraulic loading device produced triangle or sine shaped displacement curves of 10 mm maximum amplitude at theoretical loading rates up to 6 Hz and a maximum load of 1500N. These parameters exceed the largest displacement necessary to approximate strain under body weight loading of the maximum plantar fat pad thickness expected.[12-18] They also simulate the frequency content of gait[19]; and allow for 1.2 x body weight forces for most people.

All equipment in the MRI room was non-metallic, except for a few small, non-replaceable, but well secured, hardware items. Equipment in the control room, including all actuation, control, and data acquisition components did not have metallic restrictions (Figure 2.1). Electrical components were designed to minimize electromagnetic interference from the MR by utilizing twisted-pair shielded wiring, metal enclosures, and analog filtered signal lines. The displacement control system was built from a stepper motor-driven linear actuator (The Bug; Ultra Motion, Cutchogue, NY) and a stand-alone stepper driver (ST5-Si; Applied Motion Products Inc., Watsonville, CA).

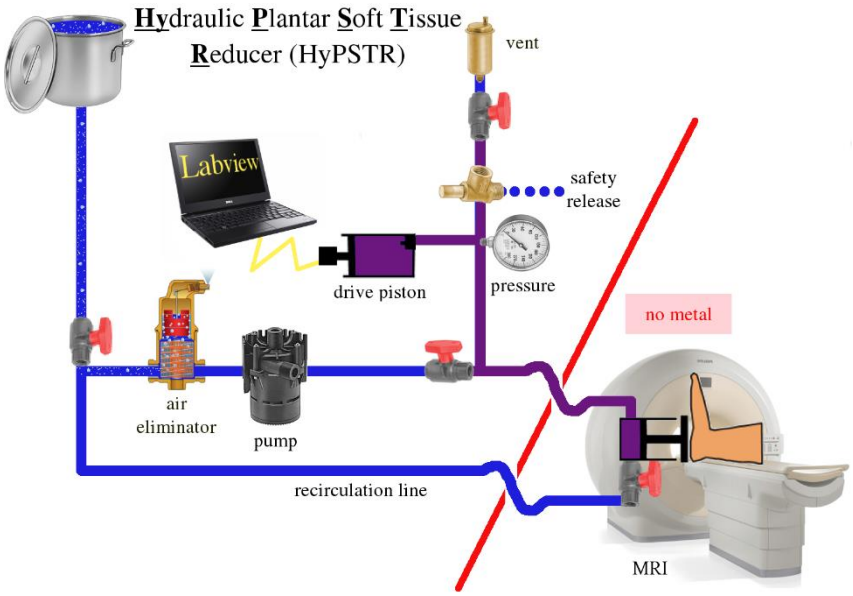


Figure 2.1. Instrument schematic showing fill and recirculation lines in blue and high pressure hydraulic lines in purple. Valves were closed to isolate the purple lines prior to testing. The red line separates the MRI control and imaging rooms.

A custom-designed, single acting, aluminum master piston/cylinder was attached to the actuator. The slave piston/cylinder was a similarly designed, single acting hydraulic system with equal bore diameter, but made of non-ferrous acetal plastic. A polycarbonate loading platen, threaded onto the end of the slave piston shaft, applied load to the plantar surface of the foot. The master and slave cylinders were connected via 9 m of 9.65 mm internal diameter, vacuum-rated nylon tubing that could withstand more than the 20 kPa negative pressure during the hydraulic retraction stage. Water was chosen as the hydraulic fluid because of its relative safety, and ease of handling compared to various oils and gels. A recirculating pump moved the water through a hydronic air eliminator (VJR075TM; Spirotherm, Glendale Heights, IL) to lower the dissolved gas below 50% to prevent entrained air bubbles from forming during system operation.

Manufacturer-provided software (Q-Programmer) controlled the stepper driver. Custom LabVIEW software acquired and logged data, sent the displacement-synchronized trigger signal to the MR control center, and monitored the solenoid hydraulic safety valve. The software ran on a laptop computer (Intel Pentium M, 1.6 GHz, 2.0 GB RAM), which hosted an external data acquisition board (USB-6212, 400 kS/s, 16-bits; National Instruments, Austin, TX). All signals were acquired at 2500 Hz to allow for digital post-processing. A rotary encoder affixed to the stepper motor (The Bug; Ultra Motion, Cutchogue, NY) measured the position and velocity of the linear actuator. A pressure transducer (PX209-200; Omega Engineering, Stamford, CT) installed in the hydraulic system monitored the load applied to the foot. System pressure measurements during each of the sixteen phases were translated to applied force values with a calibration curve that was generated by using the HyPSTR to compress a piece of silicon in series with a 500 N load cell (MC3A-500; AMTI, Watertown, MA).

The test subject lay supine in an apparatus that supported the slave cylinder and loading platen and restrained the subject's foot, leg, and torso in order to minimize movement in the imaged volume that was not due to foot tissue compression (Figure 2.2). The lower leg was secured with a hindfoot ankle-foot orthosis (American Artificial Limb, Seattle, WA) made from carbon fiber, leather, fiberglass, and Pe-Lite foam secured. A cutout in the sole allowed loading access to the heel (Figure 2.3). This could be swapped for a second, forefoot ankle-foot orthosis that provided a backstop for the foot dorsum during forefoot testing. The slave cylinder could be adjusted in the anterior/posterior direction to align the loading platen with the center of the hindfoot or forefoot, and in the medial/lateral direction to adjust for left or right feet. To further minimize motion, the subject's torso was held in place with adjustable straps over the shoulders and around the waist connected to a backboard that is integrated into the apparatus. All frame

components and fasteners were polycarbonate, acetal plastic, nylon, fiberglass, polypropylene, or polyethylene (i.e., non-metallic).

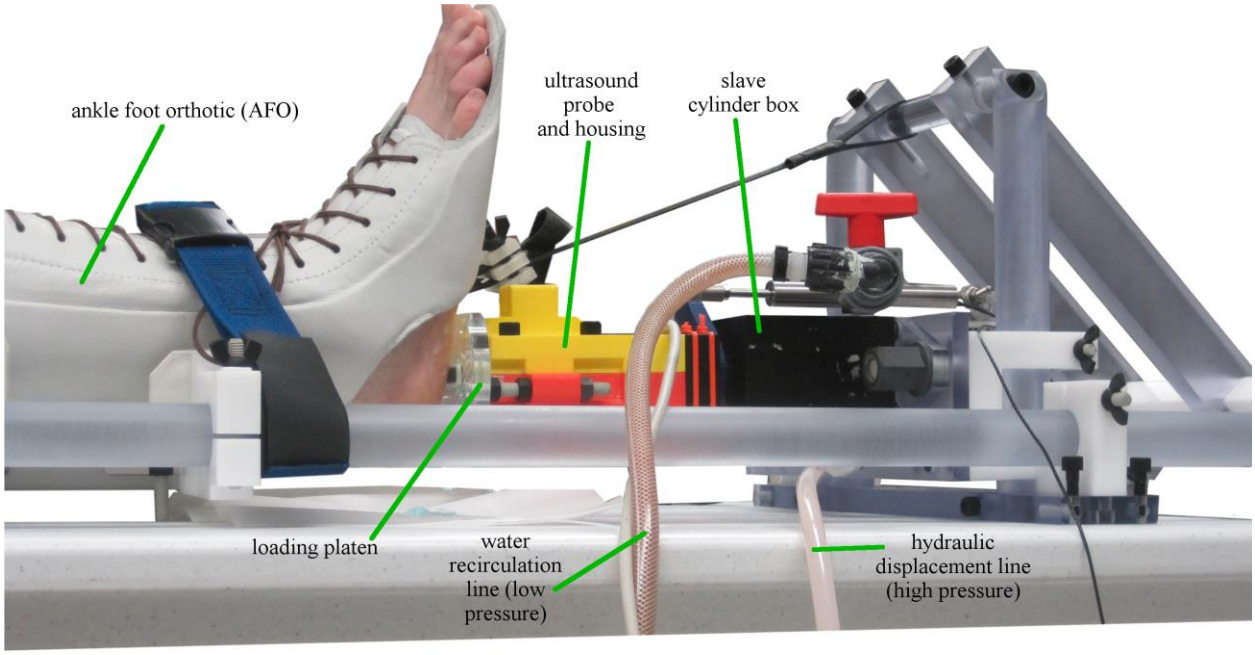


Figure 2.2. An ankle-foot orthosis (white) attached to polycarbonate rails secured the subject's leg and foot. The slave cylinder (black) is in series with an ultrasound probe held in ABS plastic (yellow and red) and with a polycarbonate platen. An ultrasound probe (not shown) was used to validate the system and was removed before any MRI scanning was conducted.

Figure 2.3. Hindfoot (white) and forefoot (black) ankle-foot orthosis. The bolts on posterior surface are for mounting to the HyPSTR loading frame.

In an MRI, moving tissue can be imaged with cardiac gating, where several short image phase acquisitions are triggered by the subject's heartbeat (one cycle). Only objects with a periodic or quasi-periodic motion can be imaged in this manner,[20] which, in this study, referred to a foot being moved by the loading platen. Instead of using a heartbeat, phase synchronization was accomplished with a 2 ms peripheral pulse unit (PPU) signal sent to the MRI scanner from a custom LabVIEW VI via a serial-to-fiber optic signal converter (Versalink; Electro Standards Laboratory, Cranston, RI). A pulse was sent upon initial movement of the actuator from its zero-position and was then repeated at the start of each displacement cycle until scan completion. The 3.0T Philips Achieva MRI Control and Data Acquisition System (Amsterdam, Netherlands) received this PPU and initiated the phase recording protocol in order to capture small portions of an image volume at specific times during a loading cycle (Figure 2.4). These partial acquisitions from each cycle were combined to generate complete images at each phase - typically 16.

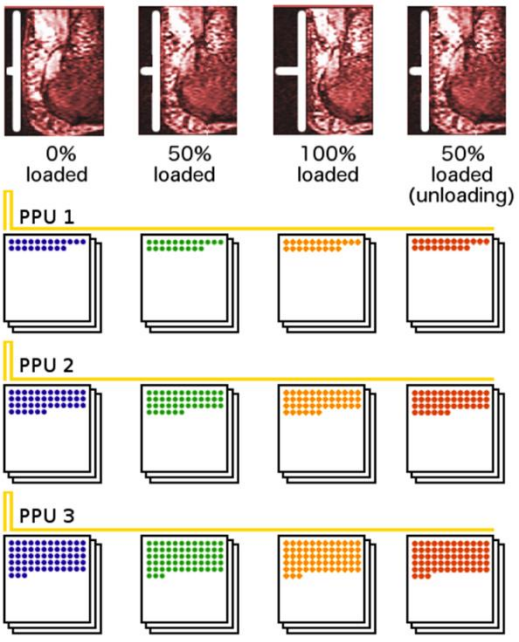


Figure 2.4. Gated MRI schematic showing 4 phase acquisitions for 3 cycles. The HyPSTR collected approximately 16 phases over 250 cycles during subject trials.

Several redundant safety measures were incorporated in the HyPSTR to protect the subject from over-loading and/or painful loading. An electronic, solenoid-operated hydraulic valve (Skinner 71295; Parker Hannifin, Cleveland, OH) was installed in the hydraulic system. When power was removed from the solenoid, the valve opened and the pressurized hydraulic fluid exited the system into a waste container, thereby removing load from the platen. Power to the solenoid could be removed by any of the following: a) an emergency-stop button near the test operator; b) an emergency-stop button at the test subject's side inside the MRI; c) by the system software if a patient-specific not-to-exceed pressure was surpassed; d) by a virtual button on the LabVIEW front panel. The not-to-exceed pressure was the system pressure at the subject's ground reaction force increased by a factor of 1.2 to account for pressure surges. The emergency-stop button inside the MRI was fiber-optic and interfaced with a controller (both from Banner Engineering, Minneapolis, MN) inside the MRI control room. The hydraulic system also included an adjustable mechanical pressure relief valve in case the solenoid failed. This valve was set to release any pressure greater than the patient-specific not-to-exceed pressure. As a final measure, the master piston was positioned in the master cylinder such that if it were to extend beyond a failed electronic limit switch, it could travel less than 1 mm before contacting the rigid aluminum cylinder bottom.

Verification testing

A series of verification tests were conducted to ensure the safety system and quantify the device's performance, MRI compatibility, and PPU signal generation timing. Data post-processing was performed in MATLAB (The Mathworks Inc., Natick, MA), Excel (Microsoft, Redmond, WA), and Image J (National Institute of Mental Health, Bethesda, MD).

Safety system. The hydraulic plumbing integrity was tested by loading the system against a coil spring until the maximum design pressure of 1034 kPa was attained. The system was held at that level for a period of 10 minutes. Each element of the safety system was then tested with these conditions to ensure operation in the most critical situation.

Device performance. The feasibility of simulating conditions similar to gait was assessed by implementing a 1 Hz triangle wave loading pattern. The repeatability of the HyPSTR was tested by conducting three separate fill/bleed/displacement test cycles of the hydraulic system at room temperature on three separate days. For each test, the empty hydraulic system was filled with fluid and air bubbles were bled. As a gold standard, a linear variable differential transformer (LVDT) was mounted in parallel with the platen. After bleeding, the system was cycled in a 0.2 Hz sinusoid ten times to 10 mm with no slave platen resistance to obtain baseline performance. We expected to observe 10 mm of platen movement for every 10 mm of driving actuator movement.

A pressure step test was performed by cycling the instrument through six staircase profiles against a piece of silicone gel between the platen and a rigid backstop with the LVDT mounted in parallel with the platen. The silicone gel measured approximately 75 cm^2 with a thickness of 2 cm, and was chosen due to its viscoelastic similarity to biological soft tissue. For each cycle, the actuator was advanced in 2 mm increments from 0 to 20 mm with a three-second pause following each move. After pausing at 20 mm, the actuator returned to 0 mm in one continuous motion and then the next cycle commenced (n = 6). This entire process was repeated three times, approximately two hours apart, yielding a total of 18 staircase data sets. These data were used to generate a target to actual (i.e., slave to master) calibration curve for all loaded tests.

Sine wave displacement profiles with peak platen amplitudes of 3.0, 5.0, 7.0, and 9.0 mm (as determined by the LVDT mounted in parallel with the platen) were cycled for 30 minutes each. The same piece of silicone gel was placed between the loading platen and the load cell. A literature review found that unloaded adult plantar soft tissue under the metatarsals and calcaneus is generally between 5.4 and 26.5 mm thick, and that the maximum strain in the soft tissue beneath either location under body weight is approximately 0.45.[12-18] For each test, the actual displacement of the loading platen vs. the prescribed displacement was analyzed at maximum extension for each cycle. Data from the first 10 cycles were excluded from that analysis due to preconditioning of the silicone gel. The root mean square error (RMSE) between a best-fit curve to the actual motor displacement and the prescribed displacement was calculated. Calibration curves correlating load on the platen with pressure in the system during loading and unloading were calculated for each displacement test.

MRI compatibility. The MRI-compatibility of the HyPSTR was determined by quantifying the influence of the loading device upon the obtained images.[21] A fluid-filled MRI phantom was imaged under four separate conditions: the loading device not present on the MRI table, the device in place, the device in place with electrical power on, and the device in place with electrical power on and the platen moving. This was repeated with an MRI-compatible encoder attached to the platen. A central, 2D slice of the phantom was taken from each condition and the MRI-compatibility was calculated by computing the total pixel intensity delta between tests.

PPU signal generation timing. Gated MRI quality also depends on precise system synchronization. If the HyPSTR actuator motor was out of phase with the imaging protocol, then parts of the loading cycle would not be recorded and/or phases would seem out of focus, similar to motion blur. The three major sources of timing error are: 1) delays or advances in the transmission of the PPU signal to the MRI control hardware (i.e., the PPU offset); 2) discrepancy in the PPU generation period relative to the HyPSTR platen period (i.e., the PPU phase offset); and 3) other unknown sources. The total error was determined by comparing the platen position as measured with the LVDT outside of the MRI core room to a subsequent scan with fiduciary markers adhered to the platen (Figure 2.5).

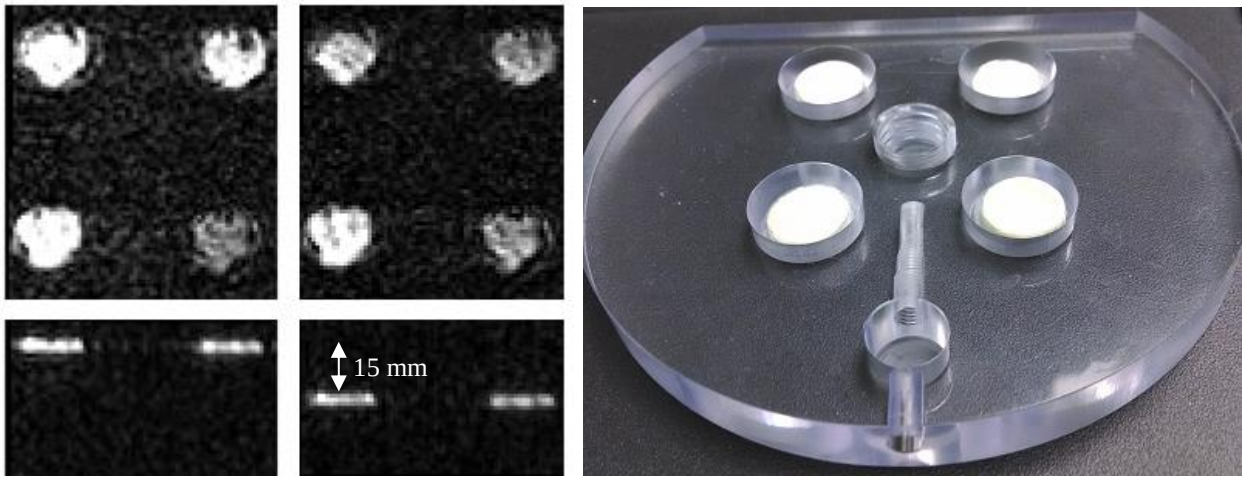


Figure 2.5. Platen markers from the starting position and the most extended position of a timing verification test. The platen is pictured on the right.

Results

Safety system. A simple static pressure test with the coiled spring proved that the system did not have any major leaks, and high-pressure tests confirmed the proper functioning of all safety relief mechanisms. No bubbles were observed in the semi-translucent high-pressure tubing, indicating that the dissolved gas was being removed.

Device performance. Prior to testing at 6 Hz, problems with the quality of the pressure signal at 1 Hz (Figure 2.6a) were identified during the 5mm triangle-wave tests. Because of the rapid starting and stopping of the water column, a water hammer was observed. Subsequently, 5mm sine wave signals with a maximum frequency of 0.2 Hz were used in order to obtain usable pressure data (Figure 2.6b).

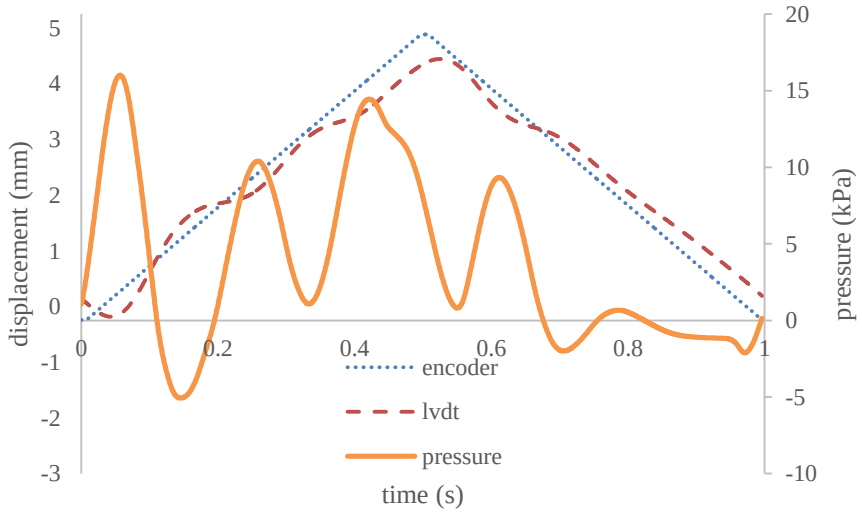


Figure 2.6a. Average of 240 triangle-wave cycles at 1 Hz and 5 mm actuator displacement.

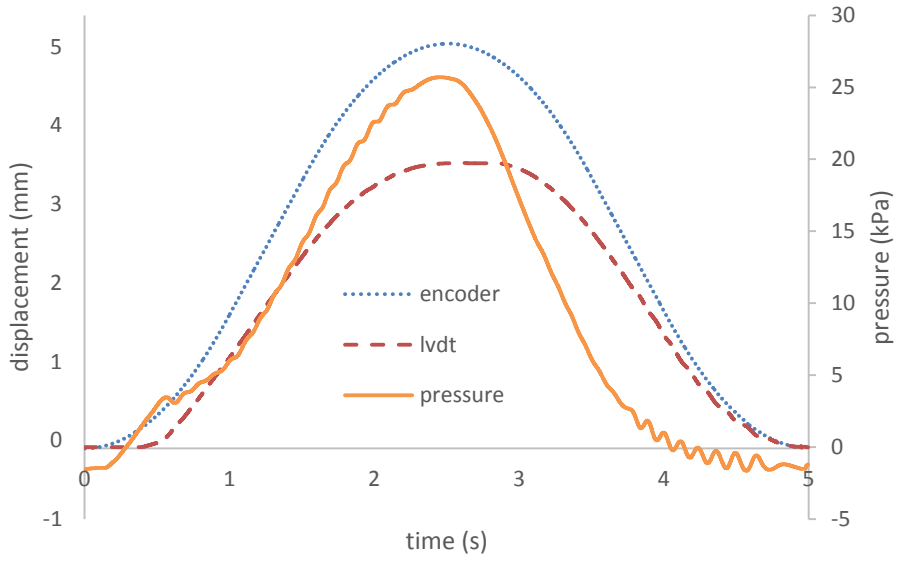


Figure 2.6b. Average of 240 sine-wave cycles at 0.2 Hz and 5 mm actuator displacement.

Additional unloaded 10 mm sine wave tests were very accurate and precise over 3 trials with 10 cycles each. However, there was some displacement loss: the maximum slave platen movement was 9.68 +/- 0.01 mm, 9.65 +/-0.01 mm, and 9.66 +/-0.01 mm for the three separate trials. Further investigation showed that, due to machining inconsistencies, the master hydraulic piston diameter was approximately 96.5% as wide as the equivalent slave piston component. Hence the maximum displacement for the slave piston should have been approximately 9.65 mm; all trials were within 0.03mm or 0.31% of this value.

As load was incrementally increased against the silicone pads for the stepped pressure test, the encoder and LVDT measured a near-linear relationship between the master and slave displacements up to 8 mm (Figure 2.7a, b). Due to system compliance, non-linear losses occurred at the higher pressure values, where a 20 mm master displacement moved the slave platen only 14.3 mm. For each set of six staircases, the first staircase was considered a pre-

conditioning phase (i.e., the results were different from the other 5) and not included in displacement averaging, thus 15 stair-case profiles remained (Figure 2.7b).

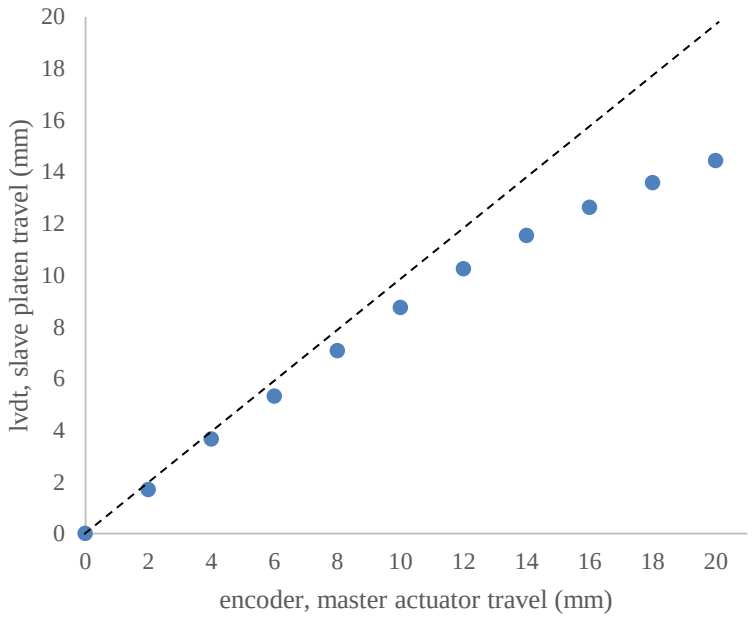


Figure 2.7a. Slave cylinder displacement vs. master cylinder displacement.

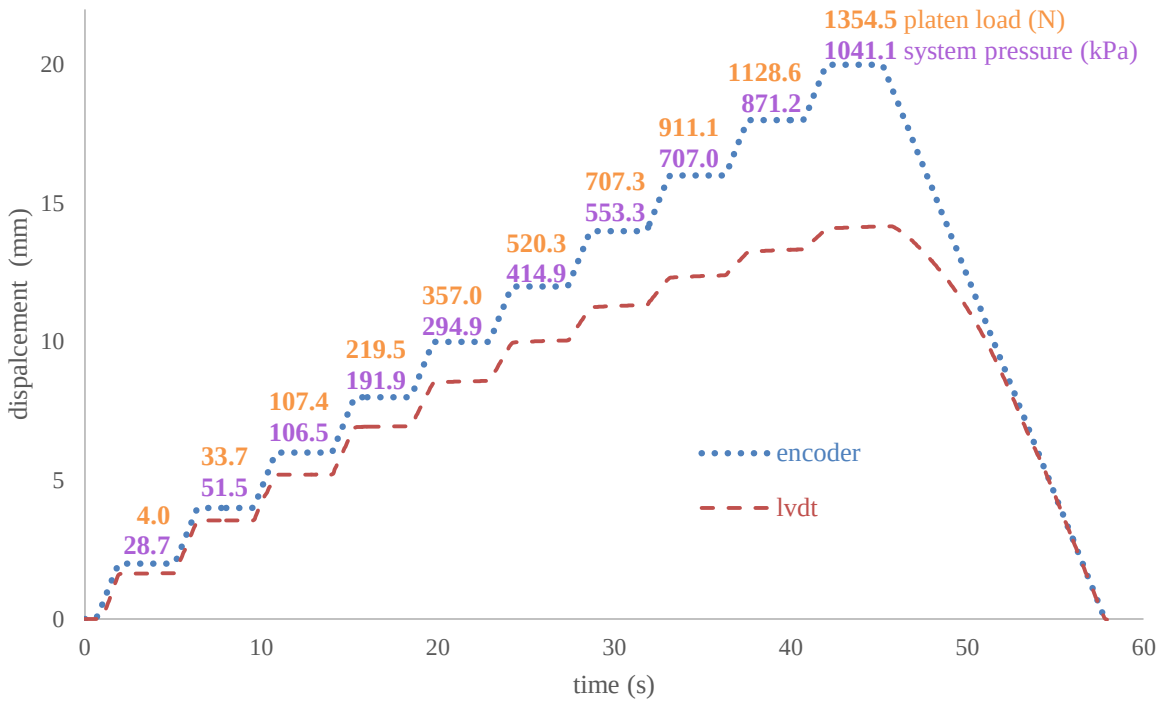


Figure 2.7b. Slave platen and. master cylinder displacement. The displacement loss was non-linear, becoming more significant with increasing master movement. It was approximated with a polynomial and used to calculate motor parameters for future tests.

Displacement and frequency accuracies were near or below 1.0% in all but one condition for the 30-minute loaded cycling tests for slave platen target positions. The displacements were least accurate in the 9 mm trial, while the frequencies deviated the most in the 3 mm trial (Table 2.1). Displacement precision (standard deviation divided by actual displacement) was 0.3% to 1.3% and frequency precision was 0.3% to 0.4% from mean values, depending on the target. (Note that these platen displacements generate the expected pressure range for subsequent *in vivo* testing.) The average encoder movement and the target curve used to command the encoder demonstrated good tracking, with an RMSE of 0.002 mm (Figure 2.8). The actuator/encoder had to move 3.526 mm to make the slave platen move 3.000 mm due to system compliance and there was a

lag between the actuator and platen displacement (Figure 2.8). Pressure-to-force calibration curves showed different patterns for loading vs. unloading (Figure 2.9).

Table 2.1. Accuracy and precision values for the HyPSTR during loading of a piece of silicon. All data were from an LVDT attached to the slave platen.

Target displacement (mm)	Actual displacement (mm)	Error (%)	Target frequency (Hz)	Actual frequency (Hz)	Error (%)	Pressure (kPa)
3.000	3.05 ± 0.04	1.67	0.1775	0.1780 ± 0.0008	0.28	48.7
5.000	4.99 ± 0.06	0.20	0.1814	0.1816 ± 0.0006	0.11	104.1
7.000	7.074 ± 0.02	1.06	0.1794	0.1796 ± 0.0006	0.11	203.5
9.000	9.373 ± 0.08	4.14	0.1782	0.1785 ± 0.0005	0.17	351.8

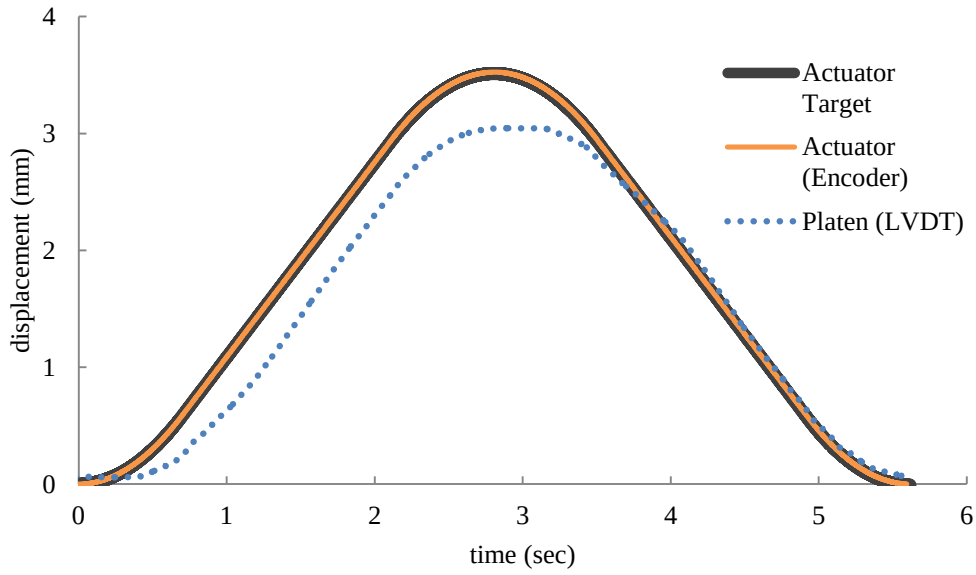


Figure 2.8. Overlap of the target sine wave used to set the motor control parameters and resulting encoder movement. The encoder path is nearly identical to the prescribed curve (shown with an exaggerated thickness).

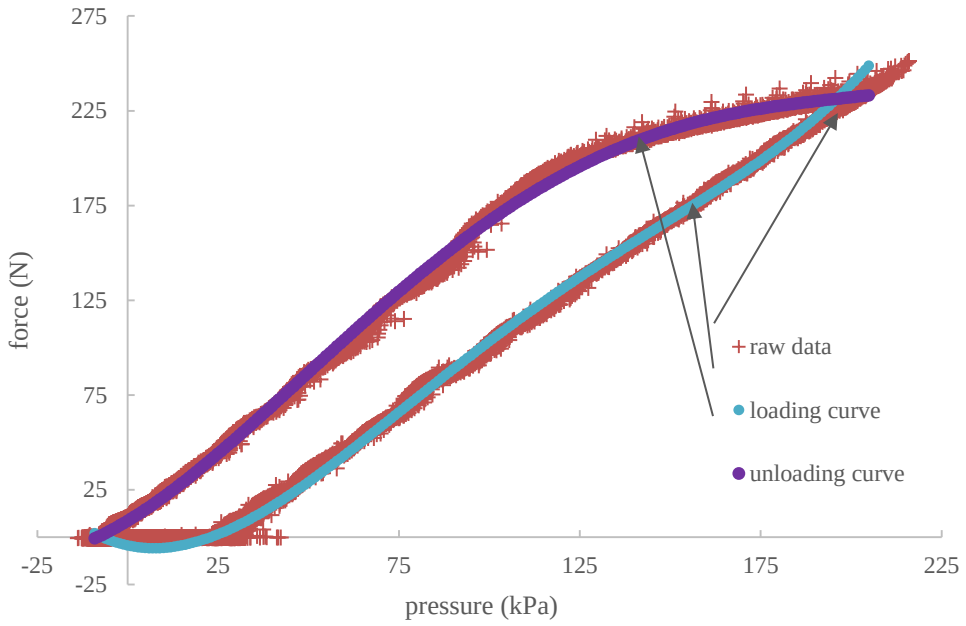


Figure 2.9. Pressure to force calibration curve from the 7-mm loaded verification test.

MRI compatibility. MRI imaging with a cylindrical phantom container in the acquisition volume showed no major artifacts (Figure 2.10a). For comparison, a temporary addition of two 5 VDC copper wires to power an MRI-compatible encoder caused two parallel line artifacts to appear (Figure 2.10b). Numeric subtractions between summed image pixel intensities showed a difference of approximately 5% without the encoder, 7% with it (Table 2.2). Two radiologists confirmed that the pixel quality observations did not indicate any significant artifacts.

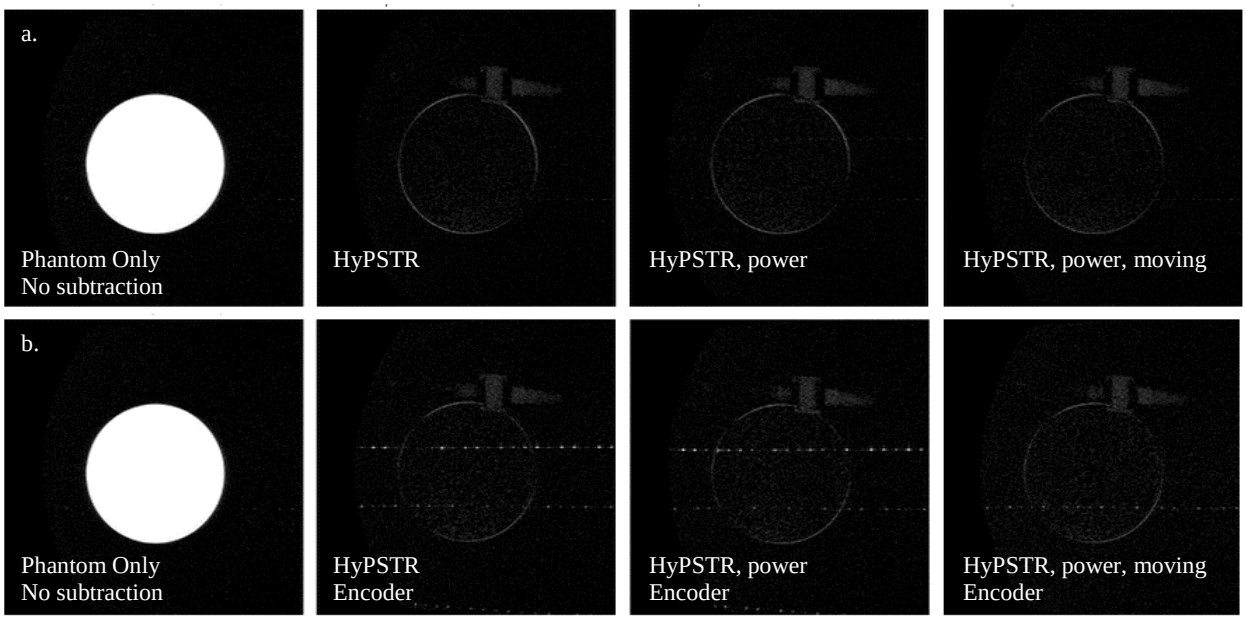


Figure 2.10. The leftmost images are of the control phantom without any hardware in the MR core. The darker images are subtractions of the same phantom, but with different system configurations. Top row (a) is without the encoder, bottom row (b) is with the encoder. Second column is only hardware; third column is hardware plus power; fourth column is hardware, plus power, plus actuator motion.

Table 2.2. Quantitative artifact analysis. The tests with the MRI-compatible encoder increased artifact pixel intensity compared to those without the encoder.

Test Condition	Pixel Intensity (normalized)	Relative Intensity to Phantom Only (%)
Phantom only (no image subtraction)	**1111615**	100.0%
HyPSTR	58526	5.1%
HyPSTR, powered	60971	5.5%
HyPSTR, powered, moving	60320	5.4%
HyPSTR **with encoder**	73912	6.6%
HyPSTR **with encoder**, powered	76031	6.8%
HyPSTR **with encoder**, powered, moving	81300	7.3%

PPU signal generation timing. The two timing verification tests showed that the platen position data from the MRI was collected 5.48 and 7.38 ms earlier than in the LVDT test. These values show a smaller than expected error; based on the measurable PPU offset and PPU phase offset, we expected the MRI data to be acquired approximately 39 ms early.[22] This means that there is still a 31.5 to 33.5 ms advance offset that we are unable to locate.

Discussion

Foot anatomy and material properties are unique to each individual and dependent on loading rates. If the tools used to assess foot mechanics (e.g., FEA models) are able to account for these subject specific differences, especially for populations with abnormally stiff plantar soft tissue such as persons with diabetes, this could lead to better treatment options. MRI is capable of providing the necessary image quality and tissue differentiation for these foot model improvements, but requires a compatible loading device. The functionality of the HyPSTR demonstrates progress toward a non-metallic, hydraulic system with repeatable performance to physiologic loading levels.

Device performance. The maximum achievable cyclic rate was 0.2 Hz because of the erratic pressure signals at higher frequencies. This was caused by a "water hammer", which is a series of pressure reflections when attempting to rapidly reverse the momentum of the water column inside the hydraulic tubing (Figure 2.6a). The high frequency oscillations to the left of the maximum and minimum peaks were likely from the piston not being able to move smoothly past its friction point at slower velocities (Figure 2.6b). A pressure snubber could potentially be used

to attenuate this ringing. Other possible solutions are to reduce the high-pressure tubing diameter to lower the mass of the water column or use a more compressible, viscous hydraulic fluid, such as glycol or oil, which can dampen the surges. The tradeoff for all of these changes is lower platen force per unit of pressure, but it is still worth considering because the plantar fat pad modulus has been shown to change significantly between 1Hz and 10Hz loading in cadaveric samples.[2]

The frequencies of the vertical ground reaction force of human gait would be more accurately simulated by impact loading profiles instead of sine waves. While virtually any motion profile may be programmed into the HyPSTR, it is currently limited to sinusoidal patterns that give cleaner pressure signals than a more appropriate (multi-frequency) triangle wave. Ultrasound instruments that permit true subject motion on a walking platform have a great advantage in this regard,[6] but the studies to date have generated two-dimensional data and are subject to trial-to-trial variability.

The HyPSTR produces accurate and precise movement for both unloaded and loaded platen conditions (Table 2.1). Though there was some system compliance (Figure 2.7a), the hydraulics can apply enough force to equal the body weight of 138kg (1354.5 N) subject (Figure 2.7b). Forces near this magnitude have been achieved before with MRI compatible devices,[8] but not in conjunction with dynamic loading. The LVDT displacement magnitude lagged behind the encoder because of the time required for the system to build up the approximately 30 kPa of pressure that was needed to overcome static friction in the slave cylinder (Figure 2.8). The effect of the static friction can be seen on the bottommost and topmost parts of the loading curve (i.e., hysteresis), where changes in pressure did not immediately affect the applied force (Figure 2.9).

MRI compatibility. Pixel comparisons of a phantom object in the MR core during various configurations of equipment, power state, and movement show that there is little concern for artifact contamination in future foot data sets (Table 2.2). The noncircular, grey shapes in the background were thought to be areas of water in the plumbing fittings and are located outside of the foot image volume. An obvious problem was caused by a copper wire leading to a platen encoder; the encoder was removed to be certain that we recorded the highest image quality. Though the encoder artifacts probably would not have affected the ability to measure mechanical properties (i.e., 7.3% is relatively similar to 5.4% added pixel brightness), there were additional issues with providing enough voltage to the sensor head over a 9m long small gauge wire. A fiber optic potentiometer might have been a better choice,[23] but commercial options were limited. Instead, the four inset MRI fiduciary markers provided a means of tracking the platen by generating a best fit plane parallel to the foot contact surface.

PPU signal generation timing. There was an unknown timing error between the MR image acquisition and device actuator movement of 32.5ms that has been empirically verified. This error might be due to the serial to fiber optic PPU conversion that is required to read the PPU signal into the MRI scanner. Additionally, once the PPU signal is received by the MRI scanner, there could be an internal delay within the system.

Future work and conclusion. Future work will focus on exploring different hydraulic fluids, installing a double acting hydraulic piston, considering fiber optic encoders, and precisely determining gated MRI timing. These efforts will address the limitations regarding the water

hammer, pressure-force hysteresis, MRI displacement tracking, and HyPSTR-to-MRI synchronization. We will also use the HyPSTR to test forefoot tissue properties. The device and methods from this work will be used to collect in vivo data for inputs and benchmarks for patient specific FEA foot models under development.

The HyPSTR aims to compliment the capabilities of MRI technology in order to determine patient specific material properties. It allows for non-invasive, internal measurements of the foot's plantar soft tissue in dynamic loading conditions. With this information and FEA modeling techniques, it may be possible to locate and quantify high stress regions within the foot that can help explain the cause and prevention of structural tissue damage.

Acknowledgments

This work was supported by VA RR&D Grant A6973R. John Shaffer helped with design consultation and construction of the ankle-foot orthotics at American Artificial Limb, Seattle, WA.

References

1. National diabetes fact sheet, 2014. *Centers for Disease Control and Prevention*2014.
2. Pai S, Ledoux WR. The compressive mechanical properties of diabetic and non-diabetic plantar soft tissue. *Journal of biomechanics*. 2010; 43: 1754-60.
3. Gefen A. Plantar soft tissue loading under the medial metatarsals in the standing diabetic foot. *Medical engineering & physics*. 2003; 25: 491-9.
4. Bakker K, Apelqvist J, Schaper NC, International Working Group on Diabetic Foot Editorial B. Practical guidelines on the management and prevention of the diabetic foot 2011. *Diabetes/metabolism research and reviews*. 2012; 28 Suppl 1: 225-31.
5. Gefen A, Megido-Ravid M, Itzchak Y. In vivo biomechanical behavior of the human heel pad during the stance phase of gait. *Journal of biomechanics*. 2001; 34: 1661-5.
6. Cavanagh PR. Plantar soft tissue thickness during ground contact in walking. *Journal of biomechanics*. 1999; 32: 623-8.
7. Erdemir A, Viveiros ML, Ulbrecht JS, Cavanagh PR. An inverse finite-element model of heel-pad indentation. *Journal of biomechanics*. 2006; 39: 1279-86.
8. Petre M, Erdemir A, Cavanagh PR. An MRI-compatible foot-loading device for assessment of internal tissue deformation. *Journal of biomechanics*. 2008; 41: 470-4.
9. Gefen A, Megido-Ravid M, Azariah M, Itzchak Y, Arcan M. Integration of plantar soft tissue stiffness measurements in routine MRI of the diabetic foot. *Clinical biomechanics*. 2001; 16: 921-5.
10. Brown PJ, Tan H, Stitzel JD. Displacement control device for dynamic tissue deformation in MRI - biomed 2010. *Biomedical sciences instrumentation*. 2010; 46: 99-104.
11. Weaver JB, Doyley M, Cheung Y, et al. Imaging the shear modulus of the heel fat pads. *Clinical biomechanics*. 2005; 20: 312-9.
12. Fields ML, Greenberg BH, Burkett LL. Roentgenographic measurement of skin and heel-pad thickness in the diagnosis of acromegaly. *The American journal of the medical sciences*. 1967; 254: 528-33.
13. Gooding GA, Stess RM, Graf PM, Moss KM, Louie KS, Grunfeld C. Sonography of the sole of the foot. Evidence for loss of foot pad thickness in diabetes and its relationship to ulceration of the foot. *Investigative radiology*. 1986; 21: 45-8.
14. Kanatli U, Yetkin H, Simsek A, Besli K, Ozturk A. The relationship of the heel pad compressibility and plantar pressure distribution. *Foot & ankle international*. 2001; 22: 662-5.
15. Kwan RL, Zheng YP, Cheing GL. The effect of aging on the biomechanical properties of plantar soft tissues. *Clinical biomechanics*. 2010; 25: 601-5.
16. Prichasuk S. The heel pad in plantar heel pain. *The Journal of bone and joint surgery British volume*. 1994; 76: 140-2.
17. Prichasuk S, Mulpruek P, Siriwongpairat P. The heel-pad compressibility. *Clinical orthopaedics and related research*. 1994: 197-200.
18. Uzel M, Cetinus E, Ekerbicer HC, Karaoguz A. Heel pad thickness and athletic activity in healthy young adults: a sonographic study. *Journal of clinical ultrasound : JCU*. 2006; 34: 231-6.
19. Antonsson EK, Mann RW. The frequency content of gait. *Journal of biomechanics*. 1985; 18: 39-47.

20. Van de Walle R, Lemahieu I, Achten E. Magnetic resonance imaging and the reduction of motion artifacts: review of the principles. *Technology and health care : official journal of the European Society for Engineering and Medicine*. 1997; 5: 419-35.
21. Tsekos NV, Khanicheh A, Christoforou E, Mavroidis C. Magnetic resonance-compatible robotic and mechatronics systems for image-guided interventions and rehabilitation: a review study. *Annual review of biomedical engineering*. 2007; 9: 351-87.
22. Williams ED. Hydraulic Plantar Soft Tissue Reducer - structural properties of the human heel pad. *University of Washington*. 2015; Mechanical Engineering Department.
23. Belforte G, Eula G. Design of an active-passive device for human ankle movement during functional magnetic resonance imaging analysis. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*. 2011; 226: 21-32.

Chapter 3: Patient-Specific Mechanical Properties of Diabetic and Healthy Plantar Soft Tissue from Gated MRI

Evan D Williams,[1, 2] Michael J Stebbins,[1,2] Peter R Cavanagh,[2,3] David R Haynor,[4] Baocheng Chu,[4] Michael J Fassbind,[1] Vara Isvilanonda[1, 2] and William R Ledoux,[1, 2, 3]

[1]RR&D Center of Excellence for Limb Loss Prevention and Prosthetic Engineering,
VA Puget Sound, Seattle, WA, USA;

[2]Department of Mechanical Engineering,
[3]Department of Orthopaedics & Sports Medicine,
[4]Department of Radiology
University of Washington, Seattle, WA, USA

Corresponding author:
William R. Ledoux, PhD
ms 151, VA Puget Sound,
1660 S. Columbian Way,
Seattle, WA 98108, USA.
(206) 768-5347 (p)
(206) 764-2127 (f)
Email: wrledoux@uw.edu

Abstract

Foot load rate, load magnitude, and the presence of diseases such as diabetes can all affect the mechanical properties of a person's plantar soft tissue. The hydraulic plantar soft tissue reducer (HyPSTR) is a tool designed to gain insight into which variables are the most significant in determining these properties (namely, stiffness). It can be used with gated magnetic resonance imaging (MRI) to capture three-dimensional images of a foot under dynamic loading conditions. Custom electronics and Labview software simultaneously record system pressure, which is then translated to applied force values with calibration curves. Data were collected for two subjects,

one without diabetes (Subject A) and one with diabetes (Subject B). For a 0.2 Hz loading rate, and strain 0.16 to 0.22, Subject A's heel pad stiffness was 10 N/mm and Subject B's stiffness was 24 N / mm. Maximum test loads were approximately 200 N. Load rate and magnitude limitations (currently both are lower than physiologic values) will continue to be addressed in the next version of the instrument. However, the current HyPSTR did produce a data set for healthy verse diabetic tissue stiffness that agrees with previous findings. In combination with the stiffness data, the MR images are also being used to improve FEA models of the foot as part of a related project.

Keywords

Heel pad, diabetes, MRI-compatible, mechanical properties

Introduction

The mechanical properties of plantar soft tissue affect the foot's ability to distribute loads experienced during normal locomotion. This is especially true for the increasing number of people (9.3% in the U.S.) with diabetes mellitus,[1] but it may also be important among healthy subjects. Research on sub-calcaneal cadaveric samples shows that the modulus of adipose tissue depends on both health (can be twice as high if diabetes is present) and loading rate.[2] When paired with unique anatomy, the difference in plantar soft tissue stiffness causes patient-specific high stress magnitudes and locations. Recent hypotheses suggest that ulcers, which often lead to

amputations, form internally at these points and expand outward to the skin.[3] We designed the Hydraulic Plantar Soft Tissue Reducer (HyPSTR) as a tool to gain understanding of the variables that define high stress in the foot.

The first studies to quantify plantar tissue stiffness measured deformation external to the skin while compressing with a linear or pendulum impact tester.[4-7] These studies lack details of internal foot mechanics, but are very useful for stiffness comparisons if approximating the heel pad as a uniform material. The quality of the data depends on how well the calcaneus is braced against the loading force. If the calcaneus shifts axially, then the deformation of the heel tissue are smaller than reported by an external measuring device.

Ultrasound and fluoroscopy can track the distance between the surface of the skin and the foot's bony structures for living subjects.[8-11] Ultrasound deformation data gives uniaxial information and does not easily differentiate between muscle, fat, and skin tissue types. The major benefits are direct tracking of the calcaneus[8-11] and physiologic kinematics and kinetics,[8, 9, 11] thus producing realistic load magnitudes[12] and rates.

In order to move beyond axisymmetric material properties, MRI has been used to capture three-dimensional (3D) images of feet under various loads.[13-15] Petre et al. have performed a series of static loading tests that represent various phases of the applied force during gait.[14] Their system used a hydraulic apparatus to apply pressure in an MR scanner. Heel tissue stiffness is nonlinear with respect to force as well as loading rate,[2, 9, 16] thus testing these parameters requires adding dynamic loading control to an MRI compatible loading device.

The Hydraulic Plantar Soft Tissue Reducer (HyPSTR) aims to conduct dynamic, in-vivo testing of healthy and diabetic foot tissue, while acquiring 3D deformations with MRI. These

data will then be used to determine heel and forefoot moduli with an inverse finite element analysis (FEA).[17] The end result will be a patient specific model that can be used to study the effect of an individual's foot mechanics on internal stresses.

Methods:

Device overview

The design and verification of the HyPSTR has been described previously in detail.[18] It consists of a Labview controlled metal actuator and data acquisition circuit board, paired with a water-filled plastic hydraulic system that is both MRI compatible and capable of applying body weight forces on a subject's foot (Figure 3.1). Displacement between the actuator and platen is transmitted via vacuum rated tubing. The system uses gated MR image acquisition to record 3D deformations of internal foot tissue while under dynamic, cyclic loading from the platen. Generally, gated MRI is used for imaging lung, heart, or other circulatory anatomy and each image capture sequence is triggered from the subject's monitored heart pulse. In order to increase control over imaging rates and to synchronize them with the HyPSTR's loading period, we used Labview to generate a peripheral pulse unit signal (PPU) that travels via fiber optic cable to the MR instrumentation. The ultimate goal was to record several complete image volumes of a foot at various loading and unloading states. Gated MRI records a small snapshot of the 3D space during each pass of the system. After approximately 20 min of imaging cycles, the data are stitched together to complete the image at each loading phase (Figure 3.2). The HyPSTR was also designed to accept a platen fitted with an ultrasound probe for preliminary testing before carrying out more extensive MRI tests.

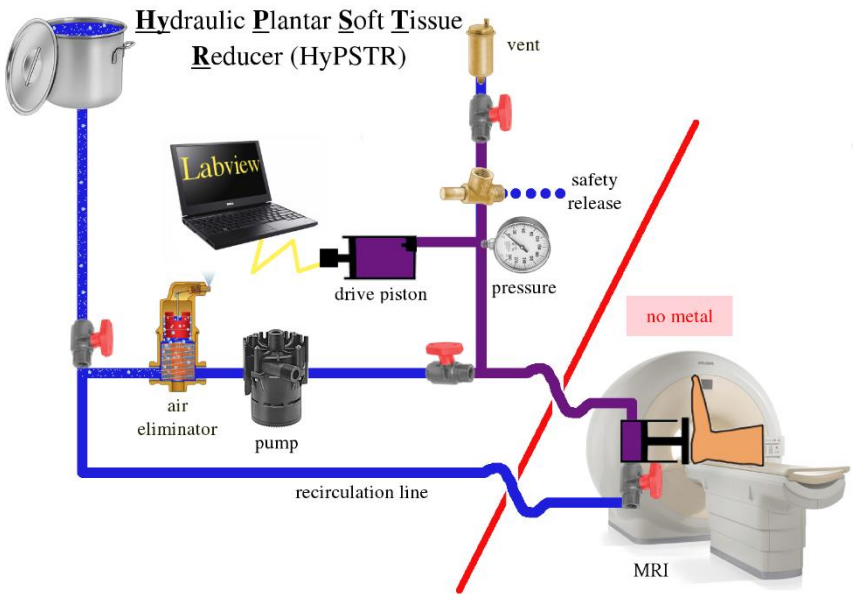


Figure 3.1. Instrument schematic showing fill and recirculation lines in blue and high pressure hydraulic lines in purple. Valves were closed to isolate the purple lines prior to testing.

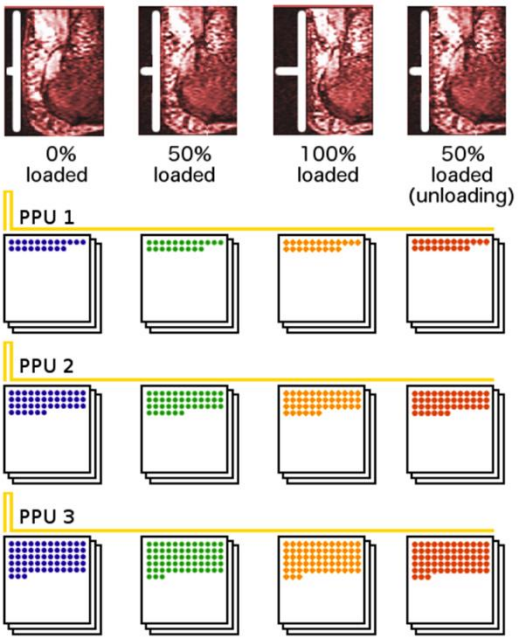


Figure 3.2. Gated MRI schematic showing 4 phase acquisitions for 3 cycles. The HyPSTR will collect approximately 16 phases over 270 cycles during subject trials.

Protocol

Two subjects were enrolled in this study, which was approved by the VA Puget Sound Health Care System IRB. Subject A had no history of diabetes and was 93 kg in mass, 180 cm tall, and 43 years old. Subject B had Type II diabetes and was 70 kg in mass, 170 cm tall, and 31 years old.

Regardless of the imaging technology (ultrasound or MR), the pre-test protocol was the same. The HyPSTR's hydraulic tubing was filled and air bubbles were removed from the system with a recirculation pump connected to a hydronic air eliminator (VJR075TM; Spirotherm, Glendale Heights, IL). The subject laid horizontally on a backboard on a table that was large enough to also accommodate the hydraulic system. The loading platen and backboard were secured relative to each other with polycarbonate rails and a set of plastic clamps (Figure 3.3). The right foot and lower leg were held tightly by a custom ankle foot orthotic (AFO) that was also fastened to the jig. Shoulder and chest straps were used to provide additional contact points to counteract skeletal movement from the platen pushing on the calcaneus. The platen position was adjustable, and the AFO was replaceable, to allow for either hind foot or forefoot testing and different size legs.

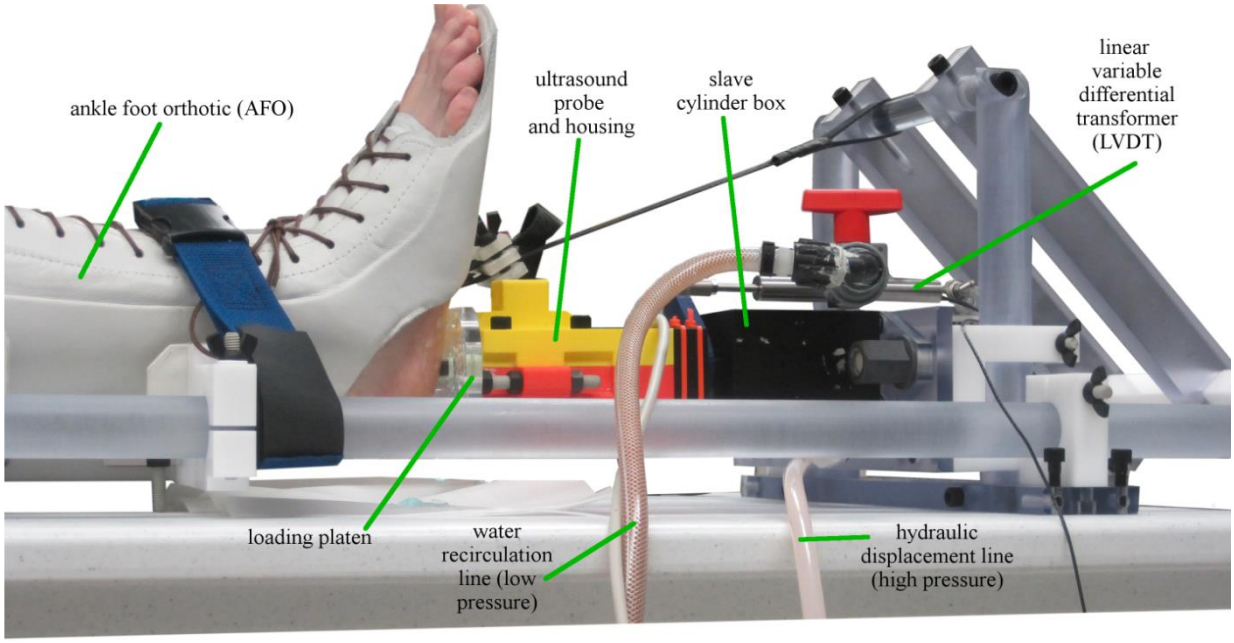
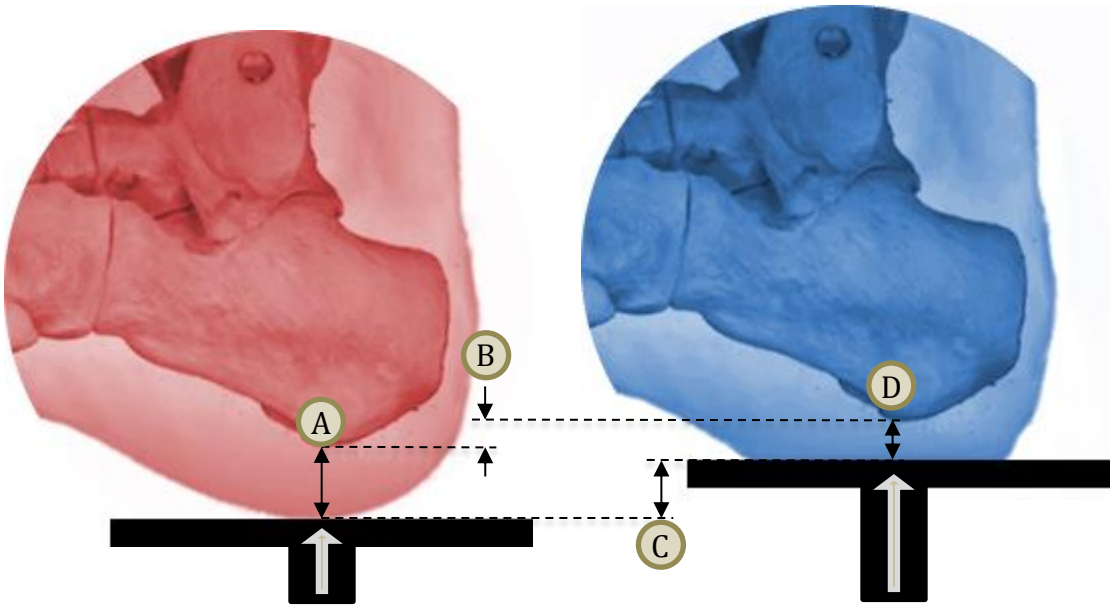


Figure 3.3. Subject's leg and foot was secured by the (white) ankle-foot orthosis that is attached to polycarbonate rails. The slave cylinder (black) is in series with an ultrasound probe held in plastic (yellow and red) and with a polycarbonate platen. Note the ultrasound probe was used to validate the system and it was removed before any MRI scanning was conducted.

Ultrasound testing provided a means of checking protocols and making fit adjustments to the jig before going to the MRI facility. Because there were no metal restrictions in the lab (as opposed to in the MR core), a Linear Variable Differential Transformer (LVDT) was attached to track the platen position (Figure 3.3). Thus, platen displacement could be compared to foot tissue deformation in order to quantify any calcaneus movement due to loading (Figure 3.4). Some amount of skeletal shift was acceptable as long as it was repeatable. If calcaneal motion was not repeatable, this would lead to image blur in the gated MRI acquisitions.



A = unloaded fat pad thickness B = calcaneus shift C = platen movement D = loaded fat pad thickness

Figure 3.4. Even if the subject's leg is well secured, his/her skeletal structure will slide in the same direction as the platen load. The left (red) foot shows the unloaded state. The right (blue) foot shows the maximally loaded state. B = C + D – A.

Ultrasound test parameters (i.e., displacement of 19 mm and 270 cycles) were set to match those expected during subsequent MRI scans. The subject's mass and a pre-defined pressure-force calibration curve[18] were used to approximate the amount of hydraulic pressure needed to reach the desired loading, ideally 50% to 100% bodyweight. A second calibration curve, the displacement between the master cylinder and slave platen, was determined along with the corresponding pressure. Because of the calcaneal shift, the required motor displacement was generally higher than calculated from the calibration curves, which were determined with a rigid backstop and a silicone block.[18] The loading platen was advanced to make visual contact with the subject's foot and then cycled ten times to precondition the plantar tissue. Previous HyPSTR verification tests showed a pressure ringing, or water hammer, problem with platen speeds

greater than 0.2 Hz.[18] A stable pressure signal was needed to calculate the applied force on the platen, thus all tests were completed with 0.2 Hz cycles even though the motor was capable of a 6 Hz rate for the given piston travel.

For MRI tests the same protocol was followed with the addition imaging synchronization via PPU timing control. Platen position was tracked with four fiduciary markers set into holes machined in the back of the MRI-specific loading platen (Figure 3.5). The heel pad thickness was reported as the distance between the platen and inferiormost point of the calcaneus. Rates and magnitudes of the sine wave loading profile were kept as similar as possible for each respective subject between the two types of imaging. Small variations in rate (+/- $2x10^{-4}$ Hz) and displacement occurred because of the impossibility of securing a subject in the HyPSTR in precisely the same manner on different days. The AFO and chest harness system relies primarily on friction to resist skeletal shifting. It was not geometrically possible to have a subject bend his/her knee (a common backstop used in other foot stiffness studies) and still fit into the narrow MR core.

Precise synchronization between the MR and LabVIEW VI was necessary to match force values to deformations. For example, if the PPU was late arriving to the MR controls, a particular deformation would correspond to an earlier force reading, or vice versa. Elsewhere, we have discussed the sources and solutions for all identifiable timing inconsistencies.[18, 19] For the scope of this study, it is sufficient to know that timing errors were minimized through careful procedures and real-time displacement feedback during the imaging and then accounted for with custom functions in MatLab post-processing.

For the ultrasound and MR data, an average, representative five second (0.2 Hz) loading curve was calculated from 260 platen cycles (270 minus 10 preconditioning). Gated MR images,

and thus the skin, fat, and muscle tissue deformations, were aligned with the loading curve by obtaining each phase's timestamp from image header files (Figure 3.5). These acquisitions were centered on each time stamp, but not instantaneous. Thus, the force for each phase was averaged over the 152 ms acquisition window.

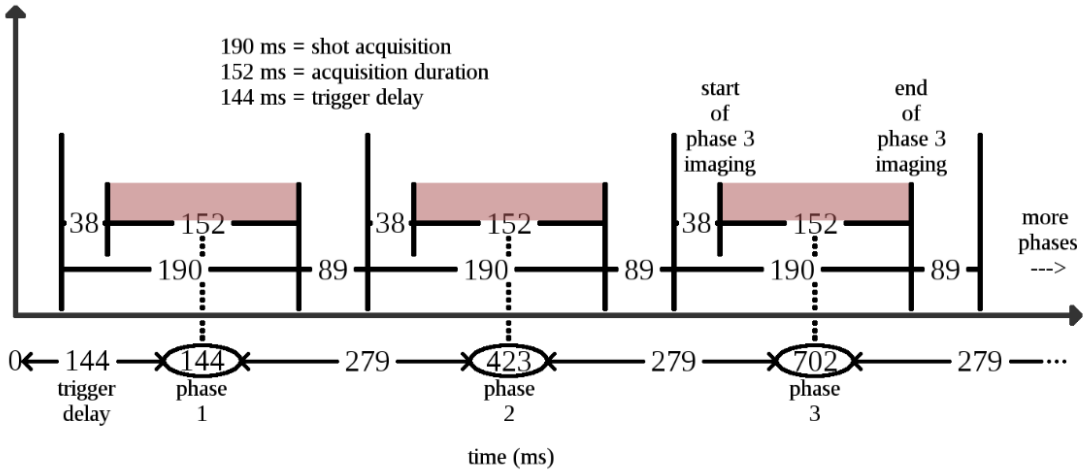


Figure 3.5. Timing schematic showing image acquisition and gap details. The 38 ms and 89 ms gap are periods that the MRI is not recording data. They depend on the other, user defined parameters and can shrink if a longer shot acquisition or greater number of phases are desired. The circled timestamps, which are reported in the MRI header files, mark the center of each phase acquisition. For the first phase, the acquisition spans from 144 ms ± 76 ms.

Results

Skeletal shifting (Figure 3.6) tracked during the ultrasound tests was 12.65 ± 0.07 mm (mean ± standard deviation) for Subject A and 12.45 ± 0.04 mm for Subject B. The 0.2 Hz target frequency was matched closely at 0.1998 ± 0.0008 Hz.

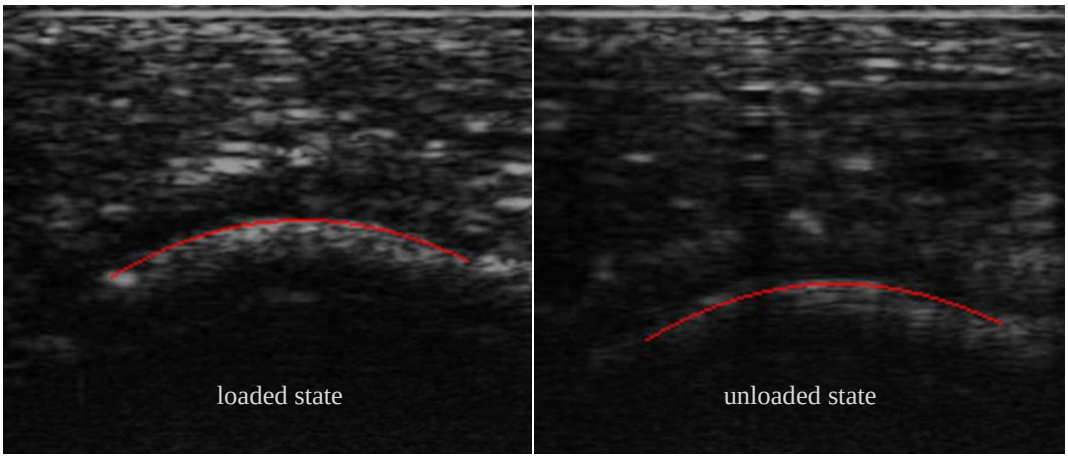


Figure 3.6. Tracking the calcaneus position with in-house Matlab software during ultrasound tests.

Sixteen MR image volumes were captured for both Subject A and B (Figure 3.7). Plantar tissue forces were analyzed with respect to time (Figure 3.8) and deformation (Figure 3.9). The peak force for Subject A was 216.0N at 30% strain (5.69 mm deformation). The peak force for Subject B was 179.6N at 21% strain (4.00 mm deformation). Linear approximations (Table 3.1) show that the heel pad stiffness was between 1.73 and 3.64 times stiffer for Subject B near the peak deformation.

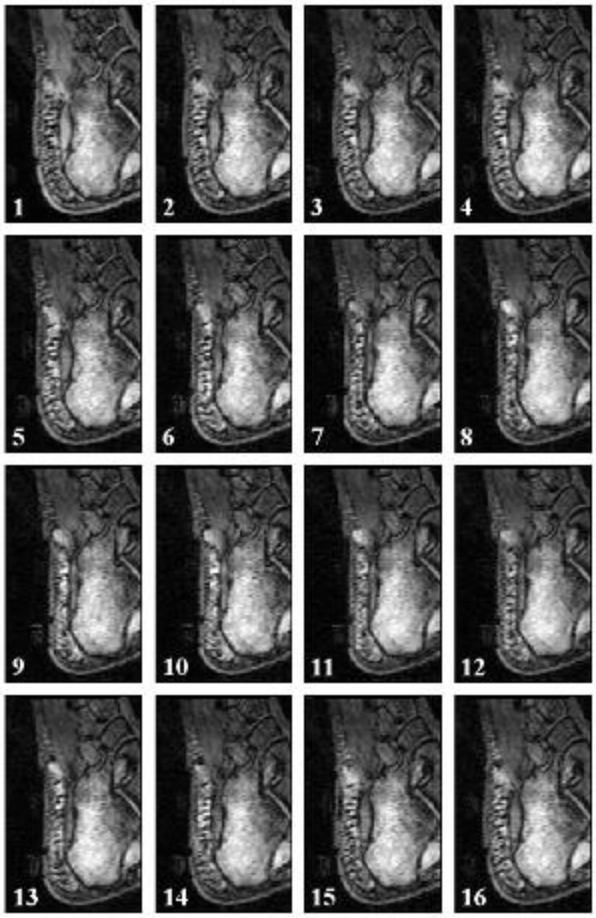


Figure 3.7. Center image slice of each gated MRI phase for Subject B.

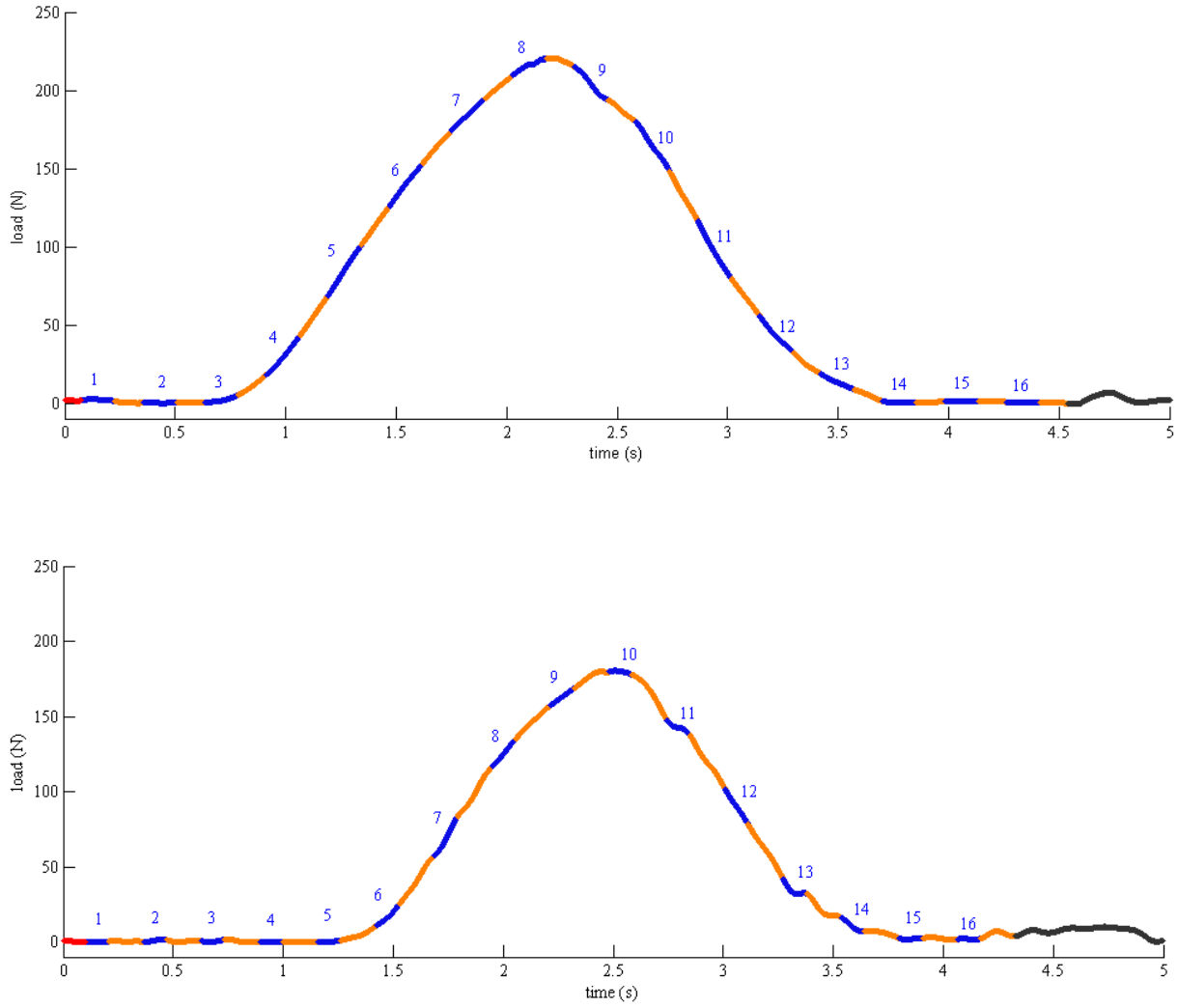


Figure 3.8. Subject A (top) and Subject B (bottom) load phases during gated MRI acquisition. Red, orange, and black segments represent times that the MR hardware is not recording image data. Blue segments (152ms in duration) are the acquisition windows.

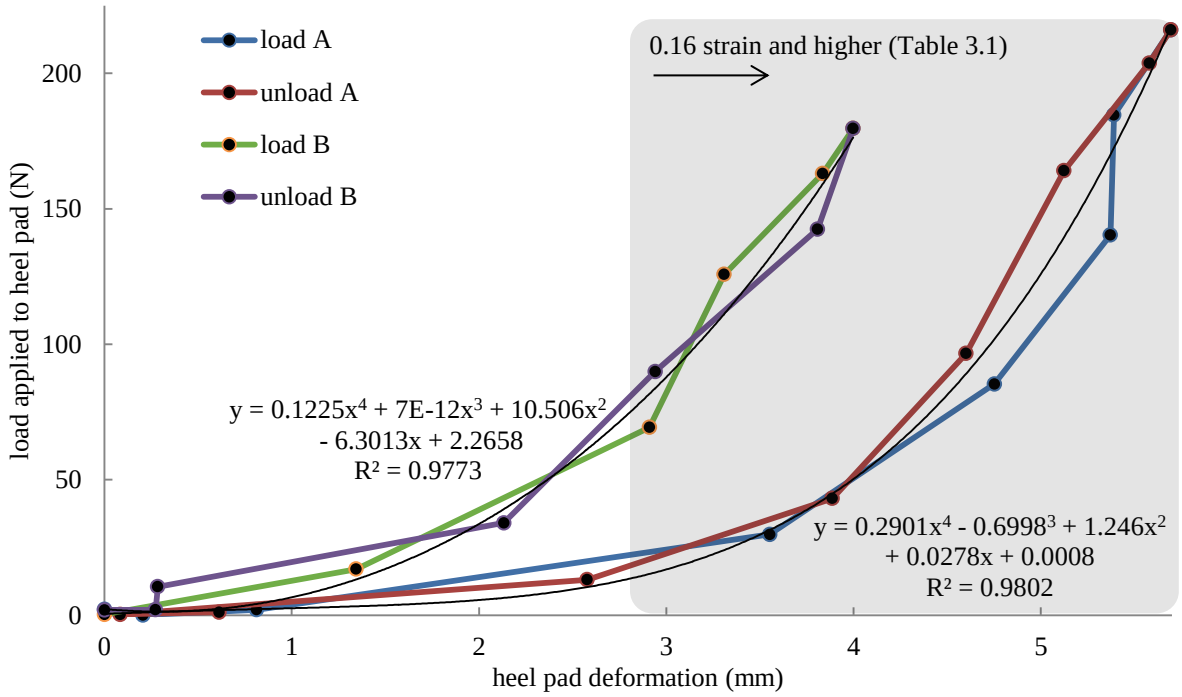


Figure 3.9. Load vs. deformation curve for Subject A (top) and Subject B (bottom) and 4th order polynomial regressions fit to the data.

Table 3.1. Heel pad deformation, force, and stiffness for 0.16 strain and higher, calculated from the derivative of the regressions in Figure 3.9. The maximum strain for Subject B was 0.22. The maximum strain for Subject A was 0.30.

Strain	Deformation		Force		Stiffness (tangent)		Relative stiffness
	Subj. A	Subj. B	Subj. A	Subj. B	Subj. A	Subj. B	Subj. B / Subj. A
(mm/mm)	(mm)	(mm)	(N)	(N)	(N/mm)	(N/mm)	(N/mm) / (N/mm)
0.16	3.02	2.99	16.22	87.16	20.29	69.63	3.43
0.18	3.39	3.36	25.55	115.66	29.65	83.04	2.80
0.20	3.77	3.74	38.92	149.42	41.78	97.83	2.34
0.22	4.15	4.11	57.43	188.99	57.03	114.16	2.00
0.24	4.52	-	82.35	-	75.80	-	-
0.26	4.90	-	115.06	-	98.44	-	-
0.28	5.28	-	157.09	-	125.34	-	-
0.30	5.66	-	210.12	-	156.87	-	-

Discussion

The HyPSTR was designed as a tool to investigate the 3D material properties of plantar soft tissue in diabetic and healthy subjects, with respect to loading rate/magnitude and tissue type. This is particularly relevant to the growing population of people with diabetes[1] who are at risk of developing foot ulcers that can lead to limb amputation. While there were some limitations in the testable range of loading magnitudes and rates, the instrumentation made significant progress toward achieving dynamic, *in vivo*, and patient-specific material property analysis. Furthermore, the data agreed with previous findings that diabetic tissue is stiffer than healthy tissue[2] and the results were used as inputs (volumetric images) and benchmarks (force vs deformation curves) for FEA models.[17]

Calcaneus tracking with an ultrasound probe showed that subjects' skeletal shifts were repeatable along the loading axis with standard deviations less than 0.1 mm. The magnitude of the shift was approximately 12.5 mm. The AFO prevented all motion between the skin and the loading jig, but shear between the skin and bone was not preventable. To compensate for this skeletal shift, the platen was moved farther than the original displacement target, which did not seem to affect the precision of system displacement and load magnitude.

The MRI load vs deformation curves (Figure 3.9) showed that Subject B had stiffer tissue than Subject A for comparable strains (Table 3.1). Pai and Ledoux report that the modulus of cadaver adipose tissue with diabetes is 1.93 times greater than that of healthy tissue. The HyPSTR stiffness values for Subject B (diabetes) were 2.65 times stiffer at 0.2 Hz (average for 16% to 22% strain). To determine an estimate for modulus, we used the MRI data to measure Subject A and B initial heel pad thicknesses, 18.85 mm and 18.69 mm, and loaded platen contact

areas 3850 mm^2 and 3888 mm^2, respectively. The values, 67 kPa (Subject A) and 115 kPa (Subject B), show the same trend as the stiffness data (Subject B was greater than Subject A).

A pendulum impact *in vivo* materials study by Aerts and De Clercq found that heel pad stiffness varied between 50 and 150 N/mm for the loading rate range 370 to 960 mm/s.[4] The initial heel pad thickness was not measured, thus the strains are not known, but loading was large enough to reach a mostly linear region on the stiffness plot. The HyPSTR data set had smaller maximum loads (approximately 200 N vs 200 to 1000 N), a slower loading rate (8 mm/s), and a different strategy for securing the lower leg (calcaneal shift was minimized and then accounted for with bone tracking in the MR images vs mechanical grounding of a bent knee). These differences make a direct comparison difficult, but HyPSTR stiffness values (Table 3.1) are within the range reported by Aerts and De Clercq. To approximate the stiffness in the same linear region, the last 1 mm of deformation for Subject A and Subject B has a stiffness of 71 N/mm (0.24 to 0.30 strain) and 91 N/mm (0.16 to 0.22 strain), respectively.

Limitations for this study include foot loading rates and magnitudes that were smaller than physiological levels[12] and MRI image resolution restrictions due to the control computer hardware. The rate limit, due to pressure ringing in the hydraulic column, might be solvable by using narrower tubing, a pressure snubber, or more compressible hydraulic fluids such as glycol mixtures or oil. Load magnitudes higher than approximately 200 N were uncomfortable to the dorsal surface of test subjects' feet in the AFO brace over sustained (20 min) cycling. Image resolution in the MRI data was coarser than expected. The MR software was capable of selecting higher quality parameters, but would have required more computer RAM/processing capacity. The 1 mm x 1 mm x 1 mm voxel (3D pixel) was not sufficient to differentiate between skin and fat tissue deformations. If fewer gated phases are acquired, perhaps by removing some that are

unloaded at the start and end of the load cycle (Figure 3.8), more computational resources will be available for image quality. There may be a more optimal balance between scan duration (subject compliance/cost), number of gated phases, voxel resolution, and total image volume.

The HyPSTR has successfully collected data that can be used to improve our understanding of stress distribution in the foot. Its primary value is being able to capture dynamic (though currently with limitations) material behavior simultaneously with patient-specific internal and external anatomy. The force and 3D deformation data set gives a complete mechanical representation of the foot for use in inverse FEA modeling projects.[17] Future versions of the HyPSTR will follow the same conceptual goals, while improving the range of the load magnitude/rate and image resolution variables.

Acknowledgments

This work was supported by VA RR&D Grant A6973R.

Greg Wilson PhD, UW – for help with determining gated MRI timing schematics and important dynamic scanning variables.

John Shaffer – for design consultation and construction of the ankle-foot orthotics at American Artificial Limb, Seattle, WA.

References

1. National diabetes fact sheet, 2014. *Centers for Disease Control and Prevention*2014.
2. Pai S, Ledoux WR. The compressive mechanical properties of diabetic and non-diabetic plantar soft tissue. *Journal of biomechanics*. 2010; 43: 1754-60.
3. Gefen A. Plantar soft tissue loading under the medial metatarsals in the standing diabetic foot. *Medical engineering & physics*. 2003; 25: 491-9.
4. Aerts P, De Clercq D. Deformation characteristics of the heel region of the shod foot during a simulated heel strike: the effect of varying midsole hardness. *J Sports Sci*. 1993; 11: 449-61.

5. Kinoshita H, Ogawa T, Kuzuhara K, Ikuta K. In vivo examination of the dynamic properties of the human heel pad. *Int J Sports Med*. 1993; 14: 312-9.
6. Robbins SE, Gouw GJ, Hanna AM. Running-related injury prevention through innate impact-moderating behavior. *Medicine and science in sports and exercise*. 1989; 21: 130-9.
7. Valiant GA. An in vivo determination of the mechanical characteristics of the human heel pad. 1985.
8. Cavanagh PR. Plantar soft tissue thickness during ground contact in walking. *Journal of biomechanics*. 1999; 32: 623-8.
9. De Clercq D, Aerts P, Kunnen M. The mechanical characteristics of the human heel pad during foot strike in running: An in vivo cineradiographic study. *Journal of biomechanics*. 1994; 27: 1213-22.
10. Erdemir A, Viveiros ML, Ulbrecht JS, Cavanagh PR. An inverse finite-element model of heel-pad indentation. *Journal of biomechanics*. 2006; 39: 1279-86.
11. Gefen A, Megido-Ravid M, Itzchak Y. In vivo biomechanical behavior of the human heel pad during the stance phase of gait. *Journal of biomechanics*. 2001; 34: 1661-5.
12. Chin Teoh J, Bena Lim Y, Lee T. Minimum indentation depth for characterization of 2nd sub-metatarsal head and heel pad tissue properties. *Journal of biomechanics*.
13. Gefen A, Megido-Ravid M, Azariah M, Itzchak Y, Arcan M. Integration of plantar soft tissue stiffness measurements in routine MRI of the diabetic foot. *Clinical biomechanics*. 2001; 16: 921-5.
14. Petre M, Erdemir A, Cavanagh PR. An MRI-compatible foot-loading device for assessment of internal tissue deformation. *Journal of biomechanics*. 2008; 41: 470-4.
15. Weaver JB, Doyley M, Cheung Y, et al. Imaging the shear modulus of the heel fat pads. *Clinical biomechanics*. 2005; 20: 312-9.
16. Ledoux WR, Blevins JJ. The compressive material properties of the plantar soft tissue. *Journal of biomechanics*. 2007; 40: 2975-81.
17. Isvilanonda V. Finite Element Modeling of the Foot. *University of Washington*. 2015; Mechanical Engineering Department.
18. Williams ED, Ledoux WR, Cavanagh PR. Design and Verification of a Gated MRI Technique to Determine Mechanical Properties of Plantar Soft Tissue. 2015.
19. Williams ED. Hydraulic Plantar Soft Tissue Reducer - structural properties of the human heel pad. *University of Washington*. 2015; Mechanical Engineering Department.

Chapter 4: Modifications and Additions to the Hydraulic Plantar Soft Tissue Reducer (HyPSTR)

The following sections are explanations of improvements, repairs, and lessons that were part of the larger project and are not discussed in detail in the paper manuscripts, Chapters 2 and 3. For example, the gated MRI to HyPSTR platen timing synchronization is summarized in Chapter 2, but the three contributing factors are divided into sub-variables with explanations here. The progression of topics is loosely chronological and covers both hardware and software aspects. For a complete list, see the Table of Contents at the beginning of this document (page v).

Analog filter circuitry

The first challenge was to reconnect the linear variable differential transformer (LVDT) to the data acquisition board. It had been removed for the last MR testing day because it was made primarily of metal and could not be used in its normal function of platen displacement measurement. We took this opportunity to drill several bulkhead holes into the metal project box that contained the data acquisition board (DAQ M Series NI USB 6212, National Instruments, Austin, TX) (Figure 4.1) and pass locking coaxial plugs through to the outside. The load cell, pressure, and LVDT wire connections could then be quickly connected without changing breadboard connectors, soldering, or exposing the project box electronics. No noticeable signal degradation was observed when the plugs were disconnected and capped with electrical tape.

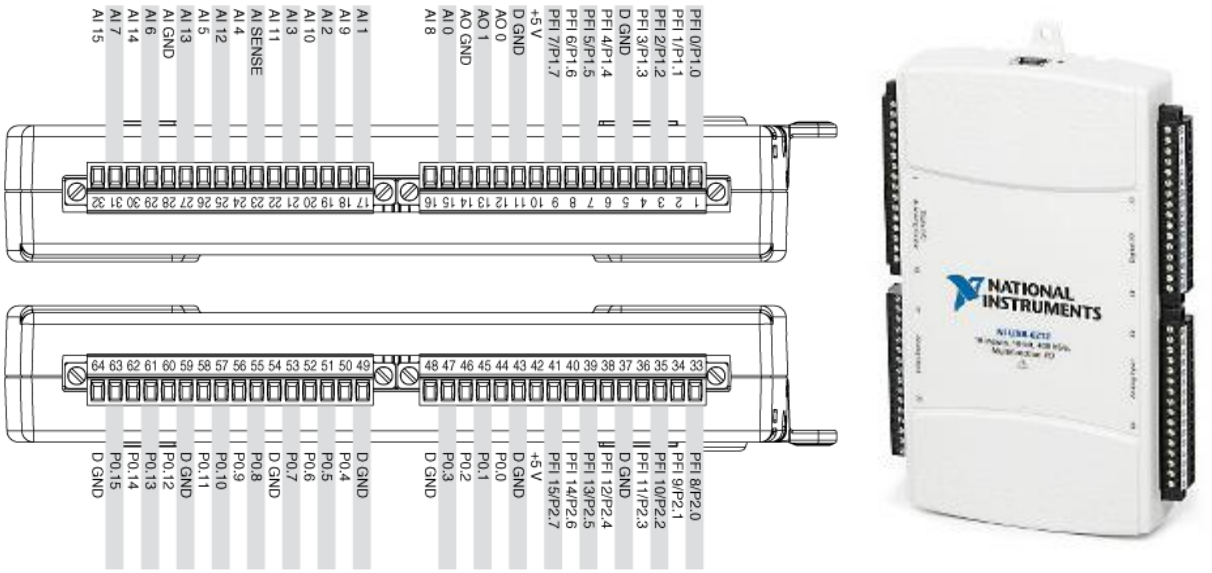


Figure 4.1. Pin inputs and output for the data acquisition board shown on the right. All digital and analog signals were routed through this device and then to LabVIEW via a USB interface. Figure is from www.ni.com.[1]

Three of four analog low bandpass filters were intact, while one required complete rewiring. Based on the relevant material[2, 3] referred in the HyPSTR_1 documentation, I designed an RC filter connected to a voltage follower (provided by an op amp) to perform this filtering. I eventually understood that this was very similar to the previous configuration, with small changes to the grounding wire layout, and it reused the existing resistor-capacitor pair (Figure 4.2). The LVDT outputs a high and low signal and the difference in voltage between the two wires is read by a comparator in the DAQ board, which means that it is equally important to filter high frequency noise from both signals.

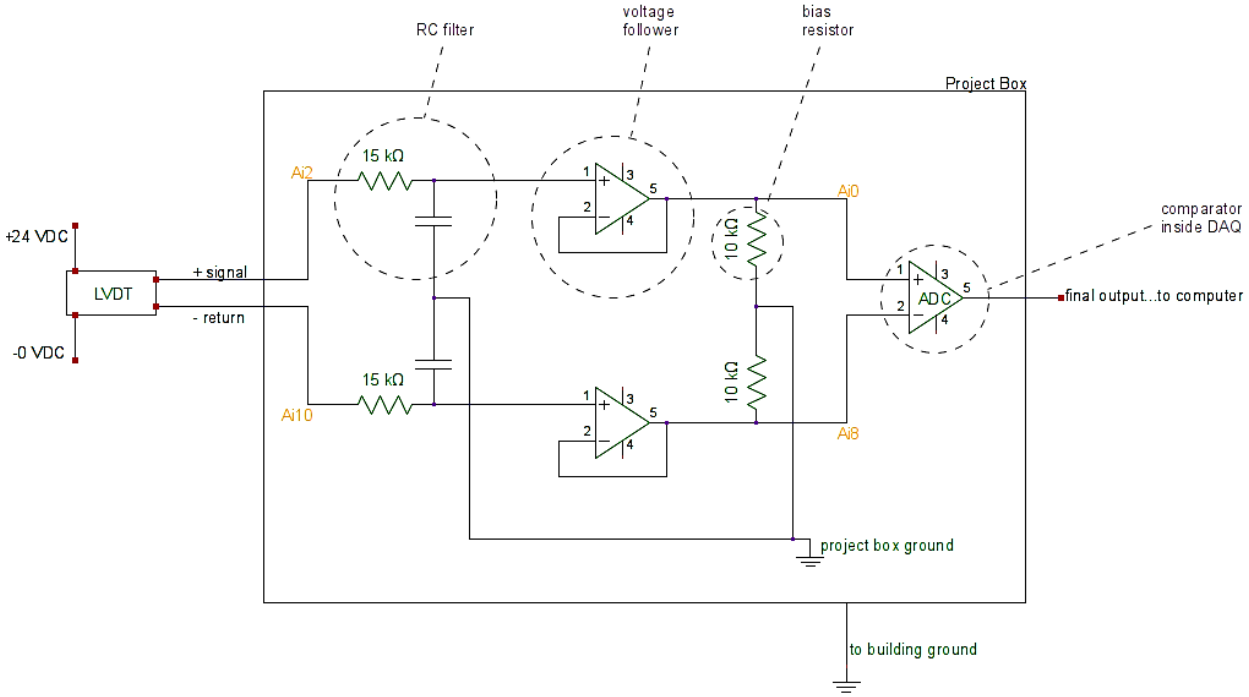


Figure 4.2. Wiring diagram for the low bandpass filter between the analog instrumentation and the DAQ. The primary function of this circuitry was to pre-filter noise before the data was smoothed in post processing software.

Stebbins found noise spikes in the electrical system at 500 Hz, 1 kHz, 20 kHz, 40 kHz, and 45 kHz.[4] The RC filter pair that he chose (15 kΩ and 0.1 μF) removes noise (Figure 4.3) above 106 Hz from the circuit and the voltage follower isolates the filter circuitry from the DAQ circuitry. If the follower was excluded and the instrument had a different output impedance, usually higher, than the input impedance of the DAQ, the signal voltage would be affected in the transfer between circuits and no longer give accurate readings.[5] An additional benefit is protection of the DAQ circuit from unexpected loading in the instrument circuit. In its most simple form, the voltage follower copies its input voltage to its output while providing electrical isolation. The resistor in the ground wire is a bias resistor that is recommended by the DAQ board manual for measuring voltage differences between two floating sources. The resistor ensures that the positive and negative wires have a common reference voltage from which to deviate.[2]

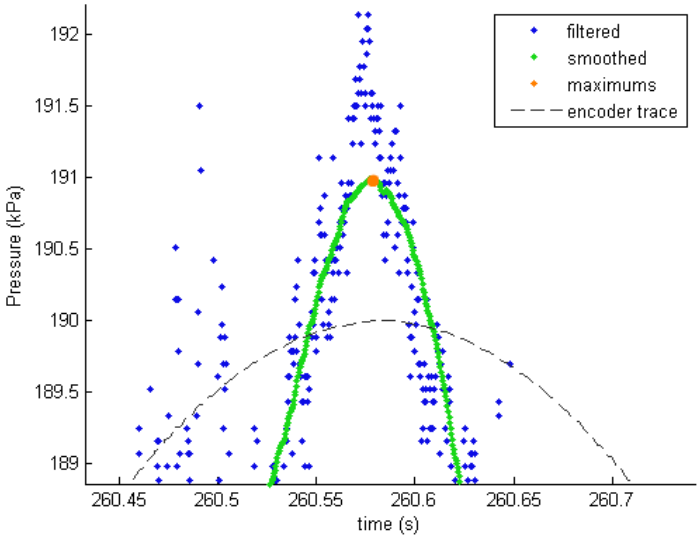


Figure 4.3. The blue data in this magnified sine wave peak is the result of analog filtering. The green data has also been smoothed with a Savitzky-Golay algorithm.

Proof of the importance for twisted pair wiring and protection from electromagnetic interference was sometimes observed in the load cell output. The LVDT wiring was routed parallel to the load cell wiring, which allowed voltage in one circuit to be inducted in the other. Even though the load cell was not usually connected for most tests, a low current signal that mimicked the LVDT position information was displayed on the LabVIEW force output plots. This effect was minimized by adding more shielding and twisting signal wires where possible. A signal voltage with a much higher current made the inducted voltage negligible when the load cell was connected to the system.

Air bubble prevention

After the instrumentation connections were restored, the next most critical modification was to stop air bubbles from forming in the high pressure tubing during and after actuator cycling (Figure 4.4a). Bubbles, whether entrained or entrapped, are highly compressible and therefore

add system compliance that causes displacement loss between the master and slave cylinders. While some compliance is unavoidable, the amount that we were experiencing was enough to severely limit the force we could apply to the plantar surface of the foot when combined with loss from skeletal shifting in the loading jig. Furthermore, compliance that changes during a test is very hard to account for in post processing. In some hydraulic applications, bubbles can heat up or compress enough that they raise the temperature of the fluid to unsafe levels or cause pitting on internal surfaces if they eventually collapse (Figure 4.4b).[6]

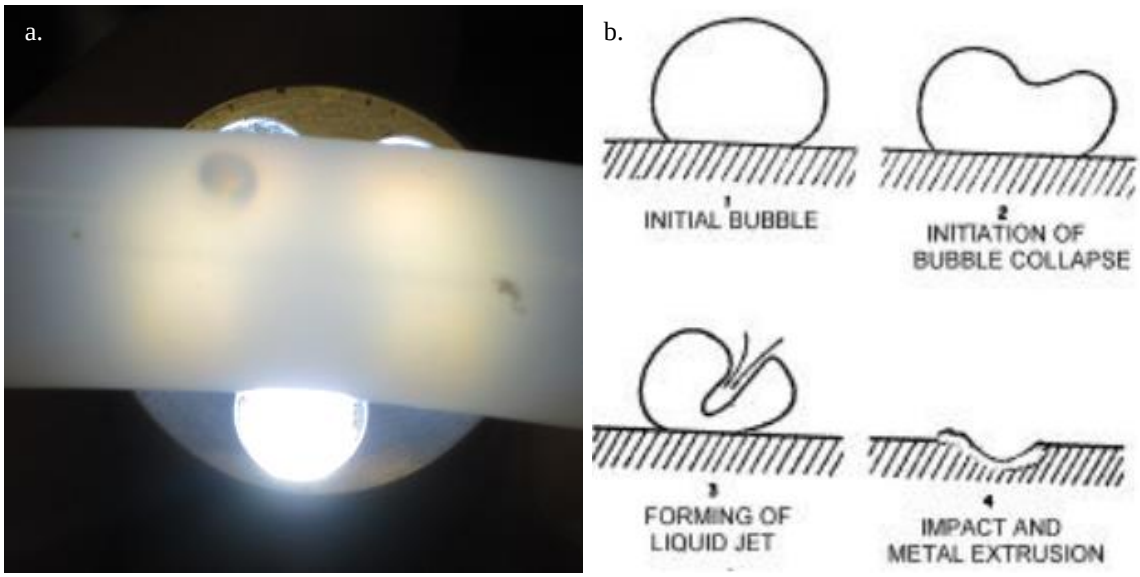


Figure 4.4. a.) A representative bubble that formed during HyPSTR_1 tests. The flashlight in the background is approximately 2 cm in diameter. b.) Damage that results from bubble collapse under high enough pressure[4].

The key to bubble control was to install an air eliminator (Spirotherm, Glendale Heights, IL) (Figure 4.5) and a centrifugal pump in a recirculation loop that could be closed to the outside environment after filling the system. The air eliminator was also a part of the HyPSTR_1, but the fill water only passed through the device once on the way into the tubing. Advice from several plumbing stores and a phone call to Spirotherm pointed me in the right direction for correct

usage. These systems are capable of removing 100% of entrapped and entrained bubbles and 99.6% of dissolved gases.[7] The syringes in the HyPSTR_1 were not able to remove the dissolved gas, which came out of solution during the hydraulic retraction phase. The retraction temporarily created a vacuum in the tubing and the gases were unable to stay dissolved at this low pressure equilibrium. After forming new entrained bubbles, the air does not immediately move back into the water.

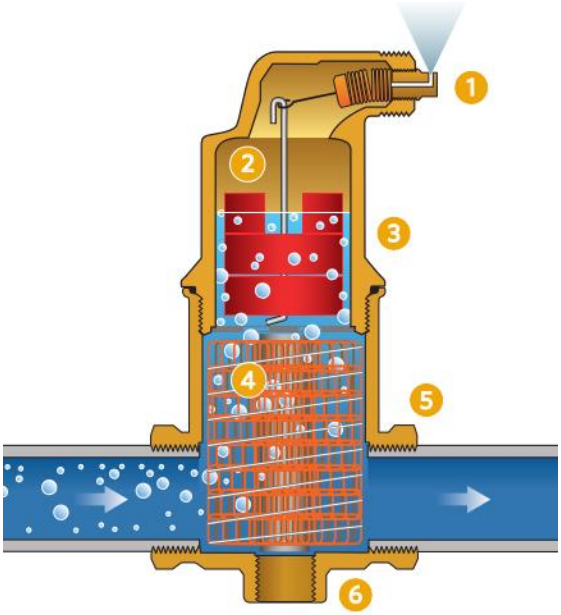


Figure 4.5. Air eliminator diagram from Spirotherm's product manual.

Low flow rates and sustained recirculation were necessary to reduce the dissolved gas to low enough levels, under 50% dissolved oxygen, as measured by a PRO-ODO meter from YSI (Yellow Springs, OH). The recirculating pump should always be placed immediately after the air eliminator so that the low pressure on the draw side promotes bubbles to come out of solution as they enter the vent. Spirotherm technicians explained that the air eliminator works most effectively around 1 gallon per minute, as opposed to the 10 gallons per minute that our pump was capable of creating. Ball valves were used to restrict the flow rate and a simple dial pressure

gauge was installed as an indirect measure of velocity. From trial and error, the optimal settings were discovered to be 30% of maximum flow capacity for 4-5 hours.

The vacuum from the actuator retraction was approximately 20 kPa below ambient, which was around 100 kPa for normal weather patterns. At 80 kPa, the water is oversaturated with dissolved gases by 20% and bubbles begin to form. This suggested that lowering the hydraulic water to 80% dissolved oxygen, before initiating the test, was sufficient to prevent bubble formation. Conservatively choosing 40% dissolved oxygen was not significantly more difficult and gave more confidence in the system. The lowest reading observed over an 18-month period of regular testing was 9.7%. Since oxygen presence is straightforward to quantify with the correct equipment, it is the standard measurement from which to extrapolate all other dissolved gases. The calculations involve knowing the partial pressures of the air in the room and then multiplying the oxygen readings by each relative magnitude.

Performance gains due to no bubbles forming during the test were seen in achievable displacement under load (Figure 4.6), as well as magnitude and frequency accuracy with respect to target inputs. Slave platen travel increased from 10.8 mm to 14.3 mm at the furthest-extended position tested. Maximum system pressure limitations prevented further testing. The pressure was 1041.1 kPa and the load was 1354.5 N at 20 mm of actuator extension.

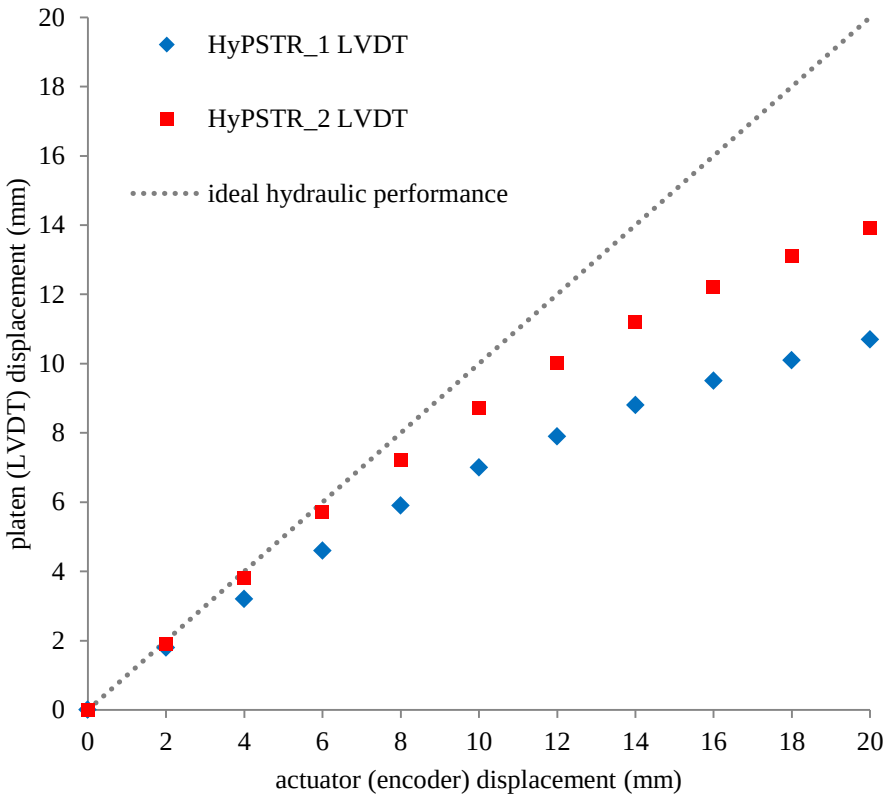


Figure 4.6. . The red squares show the increases in displacement from the HyPSTR_1 (blue diamonds) at each data point during a 20 mm actuator extension verification test. Remaining compliance in the nylon tubing still caused a non-linear displacement loss. This figure is adapted from the HyPSTR_1 thesis[6].

Actuator slipping and performance problems during 0.2 Hz verification tests

This section is covered extensively in Chapter 2, but I will provide some additional details here about my efforts to troubleshoot one particular displacement issue.

A large amount of time was spent on the HyPSTR_1 to ultimately determine that poor hydraulic efficiency was due to a slipping clutch inside of the manufacturer-provided actuator. An equally large effort went into the HyPSTR_2 to attempt to explain why efficiency dropped off similarly after one particular day in which the solenoid pressure release valve opened during a verification test. The release valve opened correctly upon exceeding a user set limit and the system was depressurized, but the motor control program and actuator did not stop. This is not a safety risk because continued motion simply pushes water out through the release valve tubing, but the retraction did pull an especially large vacuum on the hydraulic line that caused an audible "pop" in some component of the system.

The HyPSTR_2 performance was never as good after that moment. I replaced nearly every part of the system at least once and checked the motor clutch by removing the actuator and testing displacements under 250 N in a rigid jig (Table 4.1). All specifications were met and calls to Ultra Motion confirmed that the vacuum draw was not capable of damaging their motor assembly. Because the actuator is loaded at its peak travel, it will fall short between 0.04 mm to 0.1 mm from its peak position during a sine wave, or other loaded profile, in which the actuator travel was reversed. This can be used as a benchmark for future motor evaluations.

Table 4.1. Motor clutch tests. The system was cycled 9 times to a peak target of 11 mm at 0.2 Hz. Each peak displacement and load are shown below. The load decreases for every cycle because of preconditioning of the silicone squares used as a backstop in the loading jig. The cycle parameters and silicone were chosen to be similar to the conditions for our MRI trials.

Encoder *Motor* (mm)	LVDT *Motor Piston* (mm)	Load (N)
10.9998	10.9061	256.4684
10.9998	10.9134	254.9243
10.9998	10.9067	253.3066
10.9998	10.9098	252.4978
10.9998	10.9022	252.4243
10.9998	10.9001	251.7625
10.9998	10.9039	251.1007
10.9998	10.9004	250.2184
10.9998	10.9005	249.8507

Though the displacement issue was never fully resolved, some valuable lessons were learned along the way. Notably, future versions of the HyPSTR should have the motor and data acquisition be controlled by the same software. LabVIEW would be the simplest option because it already manages the timing and data acquisition loops. Output signals can be sent easily from the existing USB DAQ. In order to not overload system resources with additional motor control tasks, the laptop computer should be updated or replaced with more modern equipment.

Upon inspecting the slave piston box, I noticed that the air pressure relief holes were blocked by excess silicone grease used for lubrication of moving parts (Figure 4.7). Care should be taken to not over-grease the piston and periodic inspection should suffice to keep the holes clear. An

air-spring caused by insufficient air venting did cause significant displacement loss early in the development of the HyPSTR_1.[4]

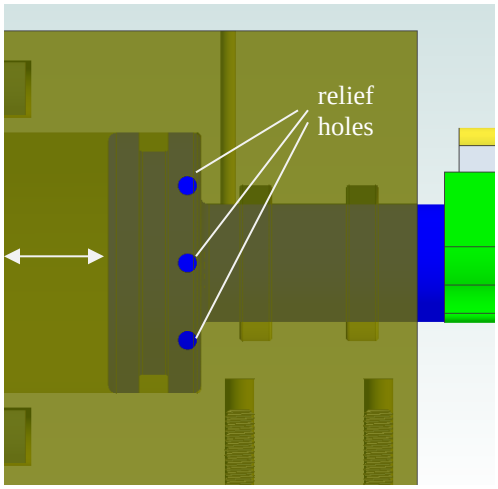
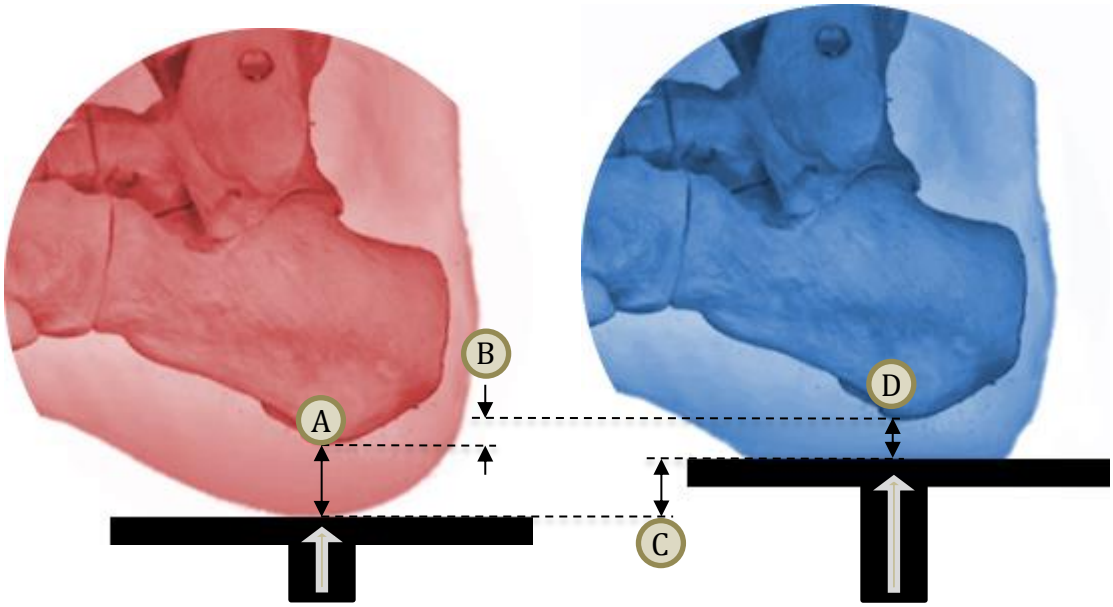


Figure 4.7. Three pressure relief holes (blue circles) were drilled on opposing sides of the slave piston box (translucent yellow). These can easily be clogged by excessive silicone grease.

Bubble formation in the hydraulic tubing still seems to be the most likely displacement issue, though none was observed during the troubleshooting. It was not possible to see bubbles in many of the opaque plumbing components, some of which have metal or plastic threads exposed to the high pressure water column. Translucent hardware was not an option for most of the fittings. It is also possible that a seal or other physical instrument interface became stressed to the point of greater compliance and now represents a weak spot in the hydraulics. This would be harder to detect because of the lack of evidence of physical failure. A complete rebuild of all components may be required. For the purposes of this thesis, the HyPSTR_2 performance was still an improvement from HyPSTR_1 and sufficient to complete the necessary MRI scanning.

Reduce skeletal shifting

The largest losses in system performance for the HyPSTR_1 during heel compression were due to the subjects' bodies shifting in the loading jig (Figure 4.8). This was primarily caused by webbing straps that could not be made tight enough to produce sufficient friction on the leg. A restraining harness was secured to a backboard and passed over the shoulders to directly counteract the axial loading, but these straps also moved when under load. Another source of unwanted motion resulting in increased calcaneal shift was the knee and/or hip joints compressing or rotating. The plantar fat pad is approximately 20 mm thick, thus even a 1 mm displacement loss is significant when attempting to measure 50% strain. Though this unwanted motion limited the test protocol strain, it was found to be very repeatable across all cycles, which is a requirement for the gated MRI protocol. If an object's movement changes characteristics over the course of a gated scan, the data from the individual cycles cannot be stitched together accurately because they will not represent the same location on the foot.



A = unloaded fat pad thickness B = calcaneus shift C = platen movement D = loaded fat pad thickness

Figure 4.8. If the subject's leg is not well secured, his/her skeletal structure will slide in the same direction as the platen load, which leaves less displacement and force remaining to compress the fat pad tissue. The left (red) foot shows the unloaded state. The right (blue) foot shows the maximally loaded state.

A very simple solution would be to increase the stroke length of the HyPSTR_1's master and slave pistons. Alternatively, a more secure brace could be designed. We chose the latter because it addressed the problem directly, rather than a symptom of the problem. Less skeletal movement also had the benefit of reducing the inconsistencies in the magnitude of cyclic motion as well as increasing subject comfort. The solution was a custom ankle-foot orthosis (AFO, American Artificial Limb, Seattle, WA) that was made to wrap securely around the lower leg.

To be comfortable and effective at stopping motion, the AFO had to be a reasonably good fit for each subject. Conversely, one can imagine a poorly-fitting hockey skate that can only be made secure by tightening the laces to the point of extreme discomfort. To avoid such

discomfort, and to ensure that a range of anthropometries were accommodated, a total of six AFOs were fabricated by American Artificial Limb in small, medium and large sizes for the right and left legs. Sizes were chosen based on a review of an anthropometric measurement database for military boot fitting[7]. According to this study, and the employees at American Artificial Limb, the following variables were thought to be the most important for fitting the lower leg/ankle/foot: calf circumference, ankle circumference, bimalleolar breadth, medial malleolar height, and lateral malleolar height (Figure 4.9). The mean value for each of these variables was plotted, along with one standard deviation in either direction. Subjects for the HyPSTR trials were expected to be more obese than the military personnel in the database. Thus, the small AFO matched the mean values, the medium AFO was one standard deviation higher, and the large AFO was two standard deviations higher. The AFO manufacturing process required making a mold from a person's leg. Volunteers were selected from the laboratory group who best fit the given dimensions (Figure 4.10).

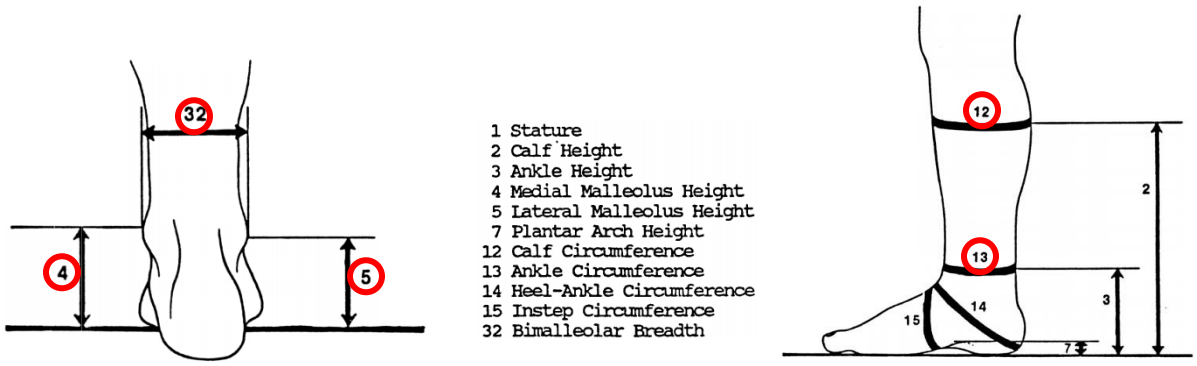


Figure 4.9. From the Natick database on Anthropometry of the Foot and Lower Leg of U.S. Army Soldiers (1985).[8] The variables circled in red were measured for determining AFO size parameters.

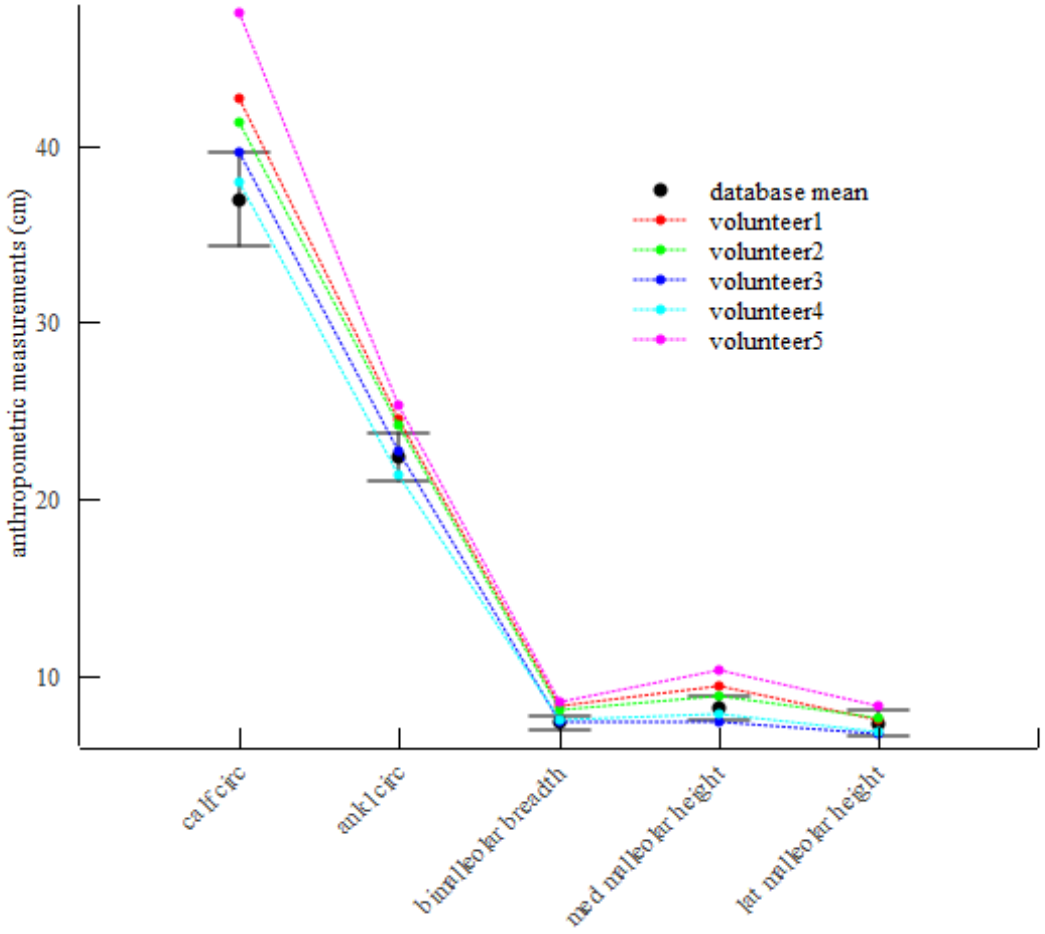


Figure 4.10. Plots of anthropometric variables for AFO sizing. Volunteer4, Volunteer2, and Volunteer5 were chosen to make the small, medium, and large molds. The error bars show the range of values within one standard deviation of the database means.

The hindfoot AFO (Figure 4.11) was made of non-metallic components. Two nylon reinforced threaded rods were adhered using fiberglass to the main carbon fiber structure, and the surfaces that come in contact with the subjects' legs were wrapped in soft white leather. This entire assembly was attached to the two plastic cuffs of the loading jig such that it was stable in all three-dimensions. The HyPSTR_2 platen passed through a cutout in the AFO onto the posterior plantar surface in order to apply force to the plantar surface of the foot.

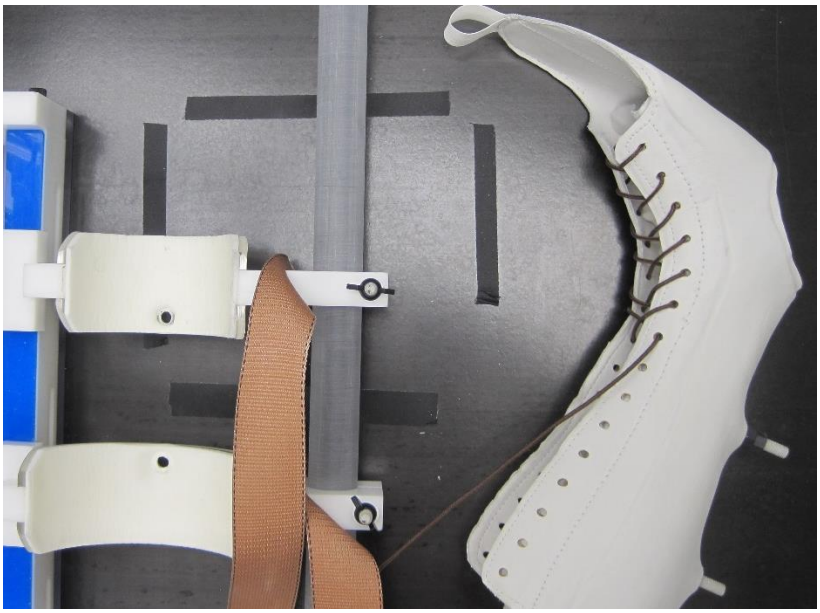


Figure 4.11. The threaded rods on the lower right fit into the holes drilled into the plastic cuffs. The original webbing strap system is still attached in this picture.

The HyPSTR_1 ultrasound (US) tests demonstrated that the calcaneus moved 7.0 ± 0.2 mm for a cyclic test that reached 120 kPa of peak pressure and 13 mm of platen travel.[4] The HyPSTR_2 US tests showed that the calcaneus moved 12.65 mm ± 0.07 mm for a cyclic test that reached 200 kPa of peak pressure and 19 mm of platen travel. The HyPSTR_1 calcaneus shift is smaller because the foot was under a smaller load. Another difference could be due to HyPSTR_2 data being processed with an automated MATLAB script that tracked the profile of the inferior most section of the calcaneus. This was previously done manually, which also explains the higher measurement uncertainty (± 0.2 mm, manual vs. ± 0.07, automated).

Most importantly, the AFO constrained the skin relative to the loading jig, thus achieving the most secure scenario possible without invasive techniques. The HyPSTR provided sufficient platen travel for all tests, even with the skeletal shifting. If more travel is needed in the future, the stroke length of the master and slave pistons should be increased from its current range of 25.4 mm.

Assess MRI timing accuracy

The following paragraph is repeated from Chapter 2 (Verification paper manuscript) for ease of reading: *"Gated MRI quality depends on precise system synchronization. If the HyPSTR actuator motor was out of phase with the imaging protocol, then parts of the loading cycle would not be recorded and/or phases would seem out of focus, similar to motion blur. The three major sources of timing error are: 1) delays or advances in the transmission of the PPU signal to the MRI control hardware (i.e., the PPU offset); 2) discrepancy in the PPU generation period relative to the HyPSTR platen period (i.e., the PPU phase offset); and 3) other unknown sources. The total error was determined by comparing the platen position as measured with the LVDT outside of the MRI core room to a subsequent scan with fiduciary markers adhered to the platen (Figure 13…below)."*

The HyPSTR was fully set up in the MRI scanning room with all tubing and fiber optic cables connected as in a normal scan. Then, taking advantage of the length of the cables, the HyPSTR was moved out onto the control room floor via the patient entrance of the MRI room; all the cabling remained connected and the LVDT was mounted in parallel to the platen. The unloaded platen was cycled 10 times, stopped, and then repeated twice more (i.e., two more 10-cycle tests were conducted.) Once the test was completed and the LVDT was removed, the HyPSTR, which remained completely connected throughout, was returned to the MRI scanning room via the patient entrance, and two unloaded MRI tests were conducted while the fiduciary markers (Figure 2.5, in Chapter 2) were imaged. The displacement of the platen in the MRI scanner was compared to displacement as measured by the LVDT. Since the volume of interest was considerably smaller than the foot, 29 phases, instead of 16, were collected. A least squares

fit of the LVDT and platen data was conducted; the difference in timing of the peaks was the sum of all temporal errors.

The PPU offset was trimodal – either a large advance or a small delay/advance. The initial generation of the PPU is triggered when the LabVIEW VI detects movement of the rotary encoder that is attached to the actuator. The encoder is made up of alternating white and black patches whose widths define the measurement resolution. A change in detected color denotes movement, but if the encoder window is spanning two patches while at rest, the detection algorithm may interpret this as if the hardware was moving immediately upon receiving power and before motion has begun. This would result in large delay. The small delay was believed to be random noise within the system resolution. This issue was solved by displaying the PPU signal and encoder position in real time in LabVIEW, which allowed the operator to measure any initial delay or advance. If there was a significant PPU offset (i.e., greater than 20ms) in the initial PPU signal, then the HyPSTR was stopped and restarted. If the difference was less than 20ms, then data collection continued and it was accounted for later in post processing. For the verification trials in this study, the PPU error was approximately 2.5ms, one early and one late (Table 4.2).

The discrepancy between the PPU generation and HyPSTR platen periods (the PPU phase offset) was also determined. While the PPU generation was found to be very accurate and precise, i.e., always less than 0.5 ms lag behind the scheduled transmission, the HyPSTR platen period was less accurate, e.g., 5012 ms instead of 5000 ms. This discrepancy could be minimized by choosing a PPU period to match the actuator movement (e.g., setting them both to 5012ms). For the verification trials, the PPU phase offset was less than 1 ms (i.e., 0.821 ms) per cycle; but the error accumulates over the course of the test (in this case, 78 cycles in 6.5 min).

The fiduciary markers (and platen) were not exactly in the center of the image volume (Figure 2.5), and since the gated MRI acquisitions progressed from one side of the volume to the other, the amount of accumulated time error from the PPU phase offset was not simply the average over 78 cycles. Instead, the number of slices in the image volume were counted (in this case, 39) and divided by 78. This gave the fraction of a slice (in this case, 0.5) acquired by each pass of the gated MRI. The average PPU phase offset for the MRI scan was then found by taking the average error accumulated by the first and last slices reached by the fiduciary markers and reported as the average PPU phase offset for region of interest (Table 4.2).

Table 4.2. MRI timing verification tests, showing expected phase acquisition offsets as calculated from measurable error. For a 6.5 min test at 0.2Hz there were 78 cycles.

Source of Timing Error	MRI marker Test #1	MRI marker Test #2
PPU offset (ms)	2.42 late	2.58 early
average PPU phase offset for region of interest (ms)	22.17 early	17.24 early
pressure wave speed (sound) (ms)	4 early	4 early
zero detection in post processing (ms)	7 early	7 early
PPU phase offset from 10 pre-image cycles (ms)	8.21 early	8.21 early
average error for the region of interest (ms)	38.96 early	39.03 early
error from timing verification	7.38 early	5.48 early
error of unknown cause	31.58 late	33.55 late

Two other relatively minor whole-test offsets (i.e., not per cycle) were realized. The speed of sound in water dictates the time for the pressure signal to travel through the hydraulic tubes to the slave platen. The platen motion was 4 ms delayed from the master actuator motion. Further, the MRI acquisition was 4 ms early (Table 4.2). There was also an offset of 7 ms caused by our post processing code. At the trough of each sine wave cycle, the encoder output a series of 0.0 mm values over a 14 ms period. The displacement and pressure values from the processing code for each phase were reported by using the first "0.0 mm" data point as the start time for each cycle. The true bottom of the trough was 7 ms later. Relative to the processed displacements, the MRI acquisition was 7 ms early (Table 4.2).

Finally, the first 10 phases of each test were ignored. The rationale was two-fold; it gave the operator time to check the PPU offset and allowed (in future loaded tests) for tissue preconditioning. This resulted in a delay of 8.21 ms (Table 4.2).

Improve MRI quality and manage safety parameters

Gated MRI is a technique that is not generally used to image periodic motion based on signals other than respiration and circulation. However, the HyPSTR's cyclic motion is chosen by the researcher and communicated to the MRI via fiber optic light pulses through the PPU. As learned throughout the study, the demands for 16, or more, hindfoot image phases over a 5 second period were often too resource-intensive (caused RAM failures) for the MRI and MRI computer control. The following is a summary of variables found to be important for modifying cardiac gated imaging protocols for use with custom load cycling parameters, starting with a few timing related terms:

Trigger Delay – This is one of two variables that control when the MRI does NOT record data because of a period of high heart activity. When imaging the heart, or any vasculature, it is best to capture anatomy in its most relaxed state (during diastole). Cardiac gating is triggered by a patient's heart beat (beginning of systole), such that imaging should not begin immediately. For the HyPSTR, the "shortest" possible value should be chosen in order to maximize the number of loading phases that can fit within one period. If there is a known period of time between platen motion and heel contact during which the system has 0 N load due to the tissue preconditioning, then the trigger delay (144 ms in this case) may be set to prevent phase recording until just prior to contact. This will avoid the unnecessary recording of multiple phases at 0 N.

RR% Window – This is the second of two variables that control when the MRI does *not* record data. It occurs at the end of each cycle (thus, approximately 4.5 s to 5.0 s into a 5.0 s cycle). In cardiac gating, it marks the time at which the MRI controls are "listening" for the beginning of the electric heart pulse. After the heart pulses, the highly active stage of heart pumping begins. The RR% window should be set similarly to the trigger delay – as short as possible, while considering the amount of time that the platen is not in contact with the foot at the end of the cycle, again due to tissue preconditioning. Shrinking the imaging window, by increasing the trigger delay and RR% window effectively increases the density of phases in the time when the heel is loaded. This leads to better temporal phase resolution.

An alternative, and maybe better, solution would be to set the MRI to record for the entire defined period and post process the acquisitions later to remove unnecessary data. However, it was not clear how to implement this level of customization with the Philips 3T system, despite many calls and emails to technicians. The physical gap created between the HyPSTR platen and foot, because of preconditioning, makes the trigger delay and RR% window controls quite useful

and perhaps necessary, even if 100% recording is possible. Lastly, the gap caused by preconditioning should be minimized by careful adjustment of the loading jig relative to the platen starting position.

Shot Acquisition (TFE dur. shot) – TFE stands for turbo field echo. There are recording pauses between phases (Figure 4.12) that are not manually controllable (89 ms in this case). These gaps are believed to be necessary delays for the MRI to reset its magnetic gradients and to save recorded data. The shot acquisition (190 ms in this case) seems to be indirectly influenced by changes to the gradient energy levels and amount of "echo" listening. All variables are input into a control sheet in the MRI software, after which an algorithm reconciles and adjusts the numbers to be within the machine's physical/hardware limits.

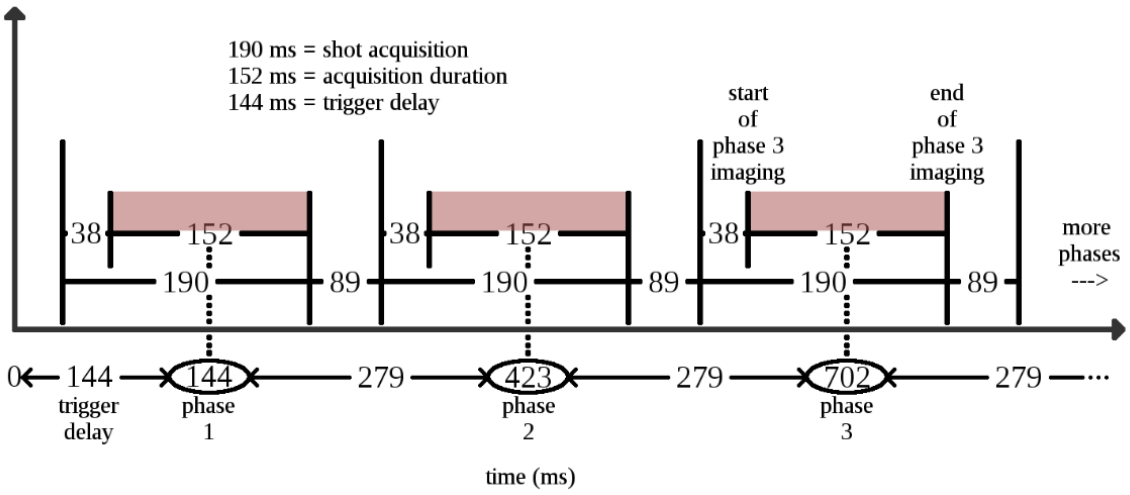


Figure 4.12. Timing schematic showing image acquisition and gap details. The 38 ms and 89 ms gap are periods that the MRI is not recording data. They depend on the other, user defined parameters and can shrink if a longer shot acquisition or greater number of phases are desired. The circled timestamps, which are reported in the MRI header files, mark the center of each phase acquisition. For the first phase, the acquisition spans from 144 ms ± 76 ms.

Acquisition Duration (TFE acq) – The acquisition duration (152 ms in this case) occurs within the shot duration (Figure 4.12) and represents the time when the MRI is actually

collecting image data. Prior to the acquisition, but still within the "shot", there is a small time gap (38 ms in this case) that the MRI uses to establish equilibrium signal levels. We believe this is related to checking signal to noise ratios or other energy-dependent aspects, which may change at a very fine scale throughout the scan. It is not a removable gap. The acquisition duration can be increased/decreased indirectly by modifying other variables. Long acquisition durations will cause losses in image quality of a moving part similar to motion blur in a standard point-and-shoot camera; however, fewer cycles are needed to collect the same amount of useful data. Total scan duration is important to minimize to reduce subject stress and/or non-controlled foot movement. The "blurriness" of this study's data sets was primarily dependent on spatial voxel resolution; thus very little time was spent trying to change the shot and acquisition durations.

Phase Interval – The number of phases designated, along with the two previous variables, defines the phase interval, i.e., the time between the midpoint of each phase. This value, 279 ms in this case (Figure 4.12), is larger than the shot acquisition and also includes an equal distribution of any leftover time not used within the 5000 ms period. If more phases are added to the protocol, the phase interval will shrink until the 89 ms gap is reduced to some (unknown) minimum. The 16-phase protocol used for the data collection in this study can be changed to record 24 phases without increasing total scan time; however, this caused computer RAM crashes and was not pursued further.

Total Scan Duration – The total amount of time per scan depends on all of the other scan parameters. Broadly speaking, more phases (temporal resolution), a larger scanning volume, and smaller voxel dimensions (spatial resolution) cause increases in total scan duration. We balanced these variables so that each scan was not be longer than approximately 20 min, for patient comfort and MRI appointment scheduling/cost considerations.

ACQ Voxel MPS (mm) – This is the voxel resolution to be captured by the MRI. Changing this 3D variable greatly affects the total scan duration. The out of plane dimension was set to 2 mm (the in-plane value was 1 mm x 1 mm) because the slice thickness was 1 mm. Thus, each voxel will overlap the previous voxel by 1 mm (out of 2 mm), allowing for a slightly lower resolution (faster) acquisition while still obtaining a 1 mm out of plane specification after the data are reconstructed.

REC Voxel MPS (mm) – The recorded voxel resolution should be set as close as possible to the ACQ voxel resolution. If these values are different, the MRI control computer will interpolate the data to conform to specified REC voxel values. It is only possible to lose raw data quality by interpolating in this manner. If a different display resolution was needed later, MATLAB and Image J post-processing sufficed. The MRI computer resources are already substantially taxed during the scanning. Reducing processing requirements (i.e., by keeping the voxel resolutions the same) at that stage is beneficial.

The REC Voxel dimensions might not always be identical to the acquired voxel dimensions because the image volume dimensions need to be a multiple of the voxel dimensions. This is easy for 1 mm x 1 mm x 1mm voxels and integer image volumes such as 142 mm x 70 mm x 90 mm. However, changing the voxel dimensions to 0.92 mm requires rounding and voxel losses on the edges, because one voxel might only be 45% inside of the image volume. The best procedure is to attempt to set the REC Voxel sizes equal to the ACQ Voxel sizes and then let the software find the closest acceptable reconstructed size.

Whole body / level – This is one of several energy density metrics used by the software to ensure that scan parameters are safe. Defer to a radiologist or MR technician regarding changes

to the defined limits. Some flexibility may be allowed to obtain better signal-to-noise ratios or resolutions if still within acceptable energy limits.

B1 RMS – B1 RMS is a magnetic strength safety metric.

PNS / Level – Peripheral nerve stimulation level, which depends on gradient reversal rates, strengths, types of scan weighting, etc. Further safety information can be found from Ham et al.[9]

Sound Pressure Level (dB) – MRI "knocking" is noisy, and caused by the highly energized electromagnetic coils bumping into each other as the magnetic forces fight against the structure holding the apparatus together. Noise level safety can be addressed with protective headphones that cover the subject's ears. Even so, some limits should not be exceeded.

Signal-to-noise – There are several software variables (some mentioned above) that affect scan time as a function of signal to noise. These (SAR – specific absorption rate, TFE – turbo field echo, and others) are related to energy magnitudes and temporal energy intensities in the changing gradients. RFOV (rectangular field of view) has the same trade-off, but has geometrical implications (magnet frequencies and imaging position are tied to each other by gradient manipulation). RFOV is a changeable value, which appears to control many other variables, including SAR and TFE. An increase in RFOV increases the signal to noise ratio, but makes the scan take longer. We increased RFOV slightly because our scan was still under 20 min. The next step up in RFOV increases scan time to 22.5 min, which exceeded our target scan duration.

References

1. Anonymous. Bus-Powered M Series Multifunction DAQ for USB - 16-Bit, up to 400 kS/s, up to 32 Analog Inputs, Isolation. National Instruments, 2014, p. 18.
2. Anonymous. How To Measure Voltage. National Instruments, 2014, p. 6.
3. Baker BC. Anti-Aliasing, Analog Filters for Data Acquisition Systems. 1999.

4. Stebbins MJ. Obtaining Material Properties of the Plantar Soft Tissue for a Patient-Specific Finite Element Model of the Foot. *University of Washington*. 2012; Mechanical Engineering Department.
5. Horowitz PaH, W. . The Art of Electronics, 2ed. Cambridge [England]: Cambridge University Press, 1989, p. 175-82.
6. Tanaka Y, Suzuki, R., Totten, G.E., Bishop, R.J. Operation and Typical Application Overview of the Use of Bubble Eliminators for De-aeration of Hydraulic and Turbine Oils.
7. Spirovent Junior. 2012.
8. Parham KR, Gordon, C. C., Bensel, C. K. Anthropometry of the Foot and Lower Leg of U.S. Army Soldier: Fort Jackson, SC –1985. *United States Army, Natick Research, Development and Engineering Center*. 1992: 18-52.
9. Ham CL, Engels Jm Fau - van de Wiel GT, van de Wiel Gt Fau - Machielsen A, Machielsen A. Peripheral nerve stimulation during MRI: effects of high gradient amplitudes and switching rates.

Chapter 5: Limitations, Potential Solutions, and Final Thoughts

While the HyPSTR_2 did allow for *in vivo*, dynamic, patient-specific, 3D mechanical testing in conjunction with gated MRI, there were certainly some major limitations. Some are similar to those experienced in the HyPSTR_1 (but perhaps different in magnitude) and others have been discovered over the last few years of research. The instrument has been improved to the point where we were able to collect useful data in support of further research on diabetic vs. healthy foot tissue properties. Hopefully, the following documentation of the limitations and possible solutions will allow the HyPSTR_3 to move another step closer to physiologic testing conditions.

Slow loading rate

The maximum loading rate during a data set collection was 0.2 Hz. The water hammer effect introduced by more rapid hydraulic direction changes at higher frequency sine waves was observed in HyPSTR_1, but not resolved in HyPSTR_2. Stebbins[1] unsuccessfully tried to fix this with a pressure snubber (cylindrical piece of plastic with small holes drilled parallel to the long axis). Other pressure snubber designs should be investigated. Glycol-water mixtures that are more compressible than pure water may also help, but they will lead to less efficient displacement transfer to the slave cylinder. Oil could provide the same benefit, but a spill near the MRI instrumentation could be very damaging and difficult to clean. Alternative hydraulic fluids could be safe if the HyPSTR is made to be more portable and robust (less fragile connections). Smaller diameter hydraulic tubing would reduce the mass of the water column. The tradeoff is that similar force magnitudes would require higher hydraulic pressure. Because our subject tests only reached 15% of the maximum system pressure, this is a viable solution.

Low magnitude loading

The lower leg securement jig was redesigned to incorporate three different sizes of ankle foot orthoses (AFO) and minimized skeletal shifting while pressing on the foot with up to approximately 200 N. However, loads higher than this were uncomfortable to both subjects during 20-plus minutes of sustained cycling at 0.2 Hz. The dorsal surface of the foot pressing against the laces (protected by thin foam and two layers of leather) left red marks on the skin of one subject (Figure 5.1), which were reduced by adding additional foam between the leather. A rigid backstop is less comfortable, but allows for greater force per mm traveled by the loading platen. A more padded backstop may or may not be more tolerable once the system reaches 200 N. In this case, the AFO was custom-built from a mold of the subject's leg and foot; thus a better contoured fit might not be possible. For subjects between 667 N (150 lbf) and 1112 N (250 lbf), 200 N is between 30% and 18% of bodyweight. (As an aside, the dorsal surface of the foot does not normally bare weight, so any system that requires that forces near bodyweight be applied will not likely be tolerable.)

Figure 5.2. Marks on the skin after removal of the AFO securement jig version 1 (of 2). More foam was added to the tongue, between layers of leather to prevent this localized pressure. The yellow oval marks the location of most discomfort.

Coarse MRI voxel resolution

High resolution is necessary for determining the boundaries between tissue types in the MRI data sets, but we were only able to collect 1 mm^3 voxels during the gated scans. Gated MRI uses more computer resources than static image capture. The three variables that needed to be optimized were: voxel resolution, quantity of imaging phases per 5 s cycle, and total image volume (field of view). It is reasonable to think that all three of these variables can be increased if a longer scan time (i.e., greater than 25 min) is permitted. The field of view was 10 x 10 x 5 cm.

Our voxel resolution was not sufficient to confidently separate skin (the 1 mm axial resolution was comparable to the magnitude of the skin thickness), but it did suffice for the fat and muscle tissue types in the data sets. However, for the purposes of this study, the compressed tissue was generalized into a single homogeneous material for the stiffness calculations; the bulk of this generalized tissue was adipose tissue inferior to the calcaneus.

The HyPSTR is adjustable such that it is capable of loading either left or right feet as well as the hindfoot or the forefoot. AFOs were made to support right and left hindfeet and right/left forefeet experiments; however, the adipose tissue inferior to the metatarsals is thinner than the calcaneus, and the forefoot anatomy, which also contains tendons, ligaments and neurovascular bundles, is considerably more complex than the hindfoot. We did collect one complete US and MRI data set for the forefoot region of one subject, but the deformations were relatively small and the anatomical structures were difficult to track, thus the data were not processed. Loads and

skeletal movement control (the HyPSTR and loading jig considerations) were comparable between the forefoot and hindfoot tests.

Coarse MRI temporal phase resolution

In addition to the voxel size considerations, there are a few additional considerations regarding the temporal resolution (number of acquired phases). Not all 16 phases provided useable data. The ten preconditioning cycles after securing the subject in the jig created a gap between the platen and skin. The first two and last two (or more) imaging phases were often during 0 N (unloaded) periods. It is good to have one unloaded phase on either end of the cycling in order to obtain an "origin" (0 N and 0 mm deformation) data point for calculations; anything more than that is not optimal. This can be solved by positioning the platen to deform the skin slightly before preconditioning and then making sure that a small gap is present after 10 cycles. The gap may also be partially due to the lower leg shifting to a final position in the AFO jig after the test load is applied. Furthermore, phase acquisition relative to platen travel (approximately 0 mm to 15 mm) depends on the timing synchronization between the HyPSTR and the MRI. Discrepancies up to 500 ms in earlier data collections caused some phases to align with unloaded states. The timing issue was resolved (to within +/- 35 ms) and is no longer a major issue regarding phase loads.

Unsynchronized HyPSTR and gated MRI acquisition

The protocol in Appendix 3 provides an acceptable workaround (within 35 ms) to the challenge of synchronizing the HyPSTR (and platen) motor control with the gated MRI acquisition timing. However, it involves the HyPSTR operator to manually monitor a Labview

VI display that shows the PPU signal generation with respect to actuator "zero" position detection. If the timing is more than 20 ms different after the 10 preconditioning cycles, the MRI acquisition is NOT started. The HyPSTR is reset in hopes that the next test will have a smaller timing offset. This offset (detailed in Chapter 4: Assess MRI timing accuracy) may not be preventable – it probably depends on the temporal accuracy of actuator motion detection, based on the width of white and black patches of an encoder that monitors the motor position. However, even after the <20 ms offset is fully accounted for, there are still some unknown offsets between the MRI platen movement (tracked by fiduciary makers) and the HyPSTR encoder. Near the end of the testing period of this research, a colleague suggested monitoring gated MRI events with the transistor-transistor logic (TTL) pulses that are emitted from the MRI control hardware. It should be straightforward to output these signals to data acquisition hardware via BNC cables.

Portability (physical)

Moving the HyPSTR to an MRI facility 8 km away was very challenging. Partly due to the prototype nature of the instrument, this disassembly and reassembly process often required a few hours of repair and reconfiguration on each end of the transportation. If the HyPSTR could be more compact without increasing the difficulty of use, data collection would be much less time-consuming and stressful. This could also be accomplished by reducing the amount of transport of the system: if possible, I would investigate options to scan at the VA Medical Center instead of downtown Seattle. I realize that scheduling MRI time might be difficult, but when all variables are considered, even night-time testing at the VA would be a simpler and less error-prone process than moving the HyPSTR in a vehicle.

Decentralized control software

The HyPSTR motor control and data collection are managed by two different software programs running on the same computer. This was done in order to simplify the development of the control interface, but ultimately it led to a complicated and sometimes problematic result. The master actuator movement is initiated with Q-programmer and then LabVIEW waits until movement is detected before sending a PPU synchronization pulse to the MRI and begin recording data. Having this step automated within LabVIEW would reduce the chance of the operator causing timing initiation mistakes by having to manage two separate programs.

Additionally, the hydraulic pressure is recorded by LabVIEW, which also triggers safety mechanisms to open relief valves in case of malfunction or user controlled "stop" switches. During one particular verification test, the high-pressure limit was tested and the relief valve opened correctly. The actuator (controlled by Q-programmer) did not stop. This did not cause an unsafe situation because the relief valve was purposefully designed to stay open after triggering. But, when the actuator retracted, it created a vacuum in some parts of the system that presumably damaged some components of the instrument (performance immediately suffered). Ultimately, this led to many months of troubleshooting and even when all components were inspected and/or replaced, the HyPSTR's ability to transfer displacement to the loading platen was never again 100%. It was, however, sufficient and consistent enough to proceed with the investigation. Controlling the actuator and recording the pressure both within LabVIEW would prevent this problem from reoccurring.

Hysteresis during unloading

Since force cannot be directly measured with metallic instruments in the MRI core, pressure-force calibration curves had to be developed; these curves showed hysteresis in the progression from loading to unloading. To determine the force at any given pressure mark, post-processing algorithms had to consider the stage within the cycle and use the appropriate curve. While this is not extraordinarily difficult, it is more of a workaround than a robust design. A double acting hydraulic system could remove this problem. The current system pushes in one direction (loading) and pulls in the other (unloading).

Silicone *vs in vivo* material force calibration

The silicone pads, used in series with a load cell and rigid aluminum backstop for the pressure-force calibration curves, are not exact representations of human foot tissue. The forces given by the calibration are correct for a given system pressure, but silicone deforms differently than human tissue. Thus, 5 mm or less of heel pad deformation (as was the case for the diabetic Subject B) may correspond to 200 N, whereas the silicone would have to deform approximately 6 mm to get the same force. (Much higher forces were also generated with silicone as it could be compressed more than the living subjects.) The preconditioning and hysteresis properties of silicone are also not the same as foot tissue.

Final thoughts

This research was completed, primarily, to fix all of the issues that were preventing dynamic, *in vivo* data collection and, secondarily, to reduce the magnitude of as many of the known problems as possible. Some limitations were not resolved (water hammer, single-acting

hydraulic hysteresis), but I believe that the instrument was greatly improved when considering its current functionality. Specific, major, improvements were consistent bubble removal from the hydraulic lines, reduction of subject skeleton movement, determining/post-processing complicated timing schematics, HyPSTR/MRI synchronization, more realistic calibration curve implementation (better curve representation at platen turn-around points), more robust plumbing and electronic layout, automated US bone tracking for pilot tests, first rounds of forefoot data collection, and greatly increased knowledge/documentation of MRI physics and scan protocol design.

We can now test all of the variables that we need to be able to test. The future challenge will be to improve the testable ranges of these variables. With the notes above and lessons learned throughout the HyPSTR_2 development, I am confident that HyPSTR_3 will continue to make significant progress.

References

1. Stebbins MJ. Obtaining Material Properties of the Plantar Soft Tissue for a Patient-Specific Finite Element Model of the Foot. *University of Washington*. 2012; Mechanical Engineering Department.

Appendix I: Matlab Post Processing

The following Matlab file is included to help future development of data processing. There are several, smaller supporting programs that are less necessary and can be implemented in many ways. The code is commented to explain the general steps of each algorithm as well as a few nuances of Matlab. It is by no means the only way to convert a .tdms LabVIEW data file to useful ASCII values. Please use any/all parts that are helpful.

final_SubAMriHeel.m 1

```
% NAME: Evan D Williams
% CREATED: August 8, 2012
% MODIFIED: January 5, 2015
% DESCRIPTION: Post processing code for HyPSTR LabVIEW .tdms data files.
% This specific file is for the Subject A heel test. Some values below
% change based on the peak pressure and displacement reached for a
% particular test.
% INPUT: a LabVIEW *.tdms data file from the HyPSTR VI.
% OUTPUT: average pressure (kPa), load (N), encoder (mm) values
% for each of the loading and unloading phase

function final_SubAMriHeel()
%% PRE-PROCESS (set consts, read data, make time vector, pick data range)
%clean up
close all; %close all previously open figures
clear all; %remove previously stored variables from RAM
clc; %clear the Matlab command window
format long; %display numerical outputs with 16 digits

%constants
filePath = 'DataWork/FinalDataSets/20130108MRI(SubA)heel/20130108_16phase_1.tdms';
averagingWindow = 501; %number of data points to use for finding initial and final
values
spanLength = 301; %used in range set of sgolay filter
sgolayDegree = 2; %used in polyfit of sgolay filter
cycleFrequency = 0.2; %Hz, of the actuator movement
cyclePeriod = 1/cycleFrequency; %inverse of cycleFrequency
numOfPhases = 16; %number of loading phases imaged

%IMPORTANT to calculate and set for each unique data set.
ppuDelay = 613; %samples, not seconds. (+) means MRI started acquiring images too late
driftDelay = 1.60575; %samples per cycle, not seconds per cycle. (+) means actuator was too fast

ink.r = [1.00 0.00 0.00]; %plot colors %Red
ink.o = [1.00 0.50 0.00]; % %Orange
ink.y = [1.00 0.85 0.00]; % %Yellow
ink.g = [0.10 0.85 0.10]; % %Green
ink.b = [0.05 0.05 0.90]; % %Blue
ink.i = [0.40 0.05 0.60]; % %Indigo
```

```
39 ink.v = [0.60 0.05 0.40]; % %Violet
40 ink.w = [1.00 1.00 1.00]; % %White
41 ink.s = [0.20 0.20 0.20]; %plot colors %Slate
42
43 %read tdms file. The convertTDMS() (openly available online) is pasted at the bottom of this file.
44 %http://www.mathworks.com/matlabcentral/fileexchange/44206-converttdms--v10-
45 allData = convertTDMS(false,filePath);
46
47 %rename read data (allData) with instrument ID variables
48 pressRaw.data = allData.Data.MeasuredData(1,3).Data;
49 presFilt.data = allData.Data.MeasuredData(1,4).Data;
50 encoder.data = allData.Data.MeasuredData(1,7).Data;
51
52 %two available encoder signals that are not used here. LabVIEW uses them to calculate the displacement,
which is
53 %reported in the MeasuredData(1,7).
54 % encA.data = allData.Data.MeasuredData(1,5).Data;
55 % encB.data = allData.Data.MeasuredData(1,6).Data;
56
57 %extract the start time, and time increment per logged value
58 wf_start_time = allData.Data.MeasuredData(1,3).Property(1,1).Value;
59 wf_start_time = datevec(wf_start_time);
60 hour = wf_start_time(4);
61 minute = wf_start_time(5);
62 sec = wf_start_time(6);
63
64 wf_increment = allData.Data.MeasuredData(1,3).Property(1,3).Value; %sample period in seconds
65 sampleFrequency = 1/wf_increment; %frequency of LabView data logging (=
1/period)
66
67 %change HH:MM:SS.SSSS format into SS.SSSS elapsed since midnight for ease of reading/processing
68 %find the longest data series. Not sure these will ever differ. There should always be the same number of
pressure
69 %and encoder data points within the .tdms data structure.
70 seriesLengths = [length(pressRaw.data), length(presFilt.data)];
71 longestLength = max(seriesLengths);
72
73 %algebra to convert
74 timeVector = ((3600*hour) + (60*minute) + sec) +...
75 (((1:longestLength)-1)*(wf_increment));
76
77 %offset time vector to be # of seconds since start of labview recording
78 plotTimeVector = timeVector-((3600*hour) + (60*minute) + sec);
79
80 %make two coarse plots of the encoder data for user to pick area of
81 %interest with GUI. This crops the data set, to remove data before the HyPSTR started moving and
82 %after it stopped.
83 encStartData = find(encoder.data>1, 10*sampleFrequency, 'first');
84 encEndData = find(encoder.data>1, 10*sampleFrequency, 'last');
85
86 plot(timeVector(encStartData(1)-(sampleFrequency):100:encStartData(end)), encoder.data(encStartData(1)- ...
87 (sampleFrequency):100:encStartData(end)),'.','color',
ink.r);
88
89 [userClickX1, ~] = ginput(1);
90 userClickX1 - ((3600*hour) + (60*minute) + sec);
91
92 plot(timeVector(encEndData(1):100:encEndData(end)+(sampleFrequency)), ...
```

```
93 encoder.data(encEndData(1):100:encEndData(end)+(sampleFrequency)),'.','color',
ink.r);
94
95 [userClickX2, ~] = ginput(1);
96 userClickX2 - ((3600*hour) + (60*minute) + sec);
97
98 close(gcf); %close the GUI. Not needed for anything later.
99
100 %find the timeVector points that are closest (rounding up in time) to the two user clicks. The user clicks
101 %are not precisely on top of a data point.
102 startIndex = find(timeVector >= userClickX1,1,'first');
103 endIndex = find(timeVector >= userClickX2,1,'first');
104
105 %remove data outside of the selected range for each data series
106 %and, determine and save initial/final time (index) values for each data series
107 %truncateAndGetEnds() is a custom function and at the bottom of this code.
108 [encoder.data,...
109 encoder.initial,...
110 encoder.final] = truncateAndGetEnds(encoder.data, startIndex, endIndex, averagingWindow);
111
112 [pressRaw.data,...
113 pressRaw.initial,...
114 pressRaw.final] = truncateAndGetEnds(pressRaw.data, startIndex, endIndex, averagingWindow);
115
116 [presFilt.data,...
117 presFilt.initial,...
118 presFilt.final] = truncateAndGetEnds(presFilt.data, startIndex, endIndex, averagingWindow);
119
120 [plotTimeVector,...
121 ~,...
122 ~,] = truncateAndGetEnds(plotTimeVector, startIndex, endIndex, averagingWindow);
123
124 %% SMOOTH the filtered data series using MatLab's Savitzky-Golay filter inside of "smooth.m"
125 %Encoder: no smoothing necessary
126 encoder.smooth = encoder.data;
127
128 %Pressure: smooth
129 presFilt.smooth = smooth(presFilt.data, spanLength, 'sgolay', sgolayDegree);
130
131 %% MINIMUMS and MAXIMUMS: find with algorithm. use filtered-smoothed data
132 %Encoder (remember, encoder data doesn't get smoothed...use .data
133 [encoder.mins, encoder.maxs,...
134 encoder.minsIndex, encoder.maxsIndex] = findMinMaxSet(encoder.data, cycleFrequency, sampleFrequency);
135
136 %Press
137 [presFilt.mins, presFilt.maxs,...
138 presFilt.minsIndex, presFilt.maxsIndex] = findMinMaxSet(presFilt.smooth, cycleFrequency, sampleFrequency);
139
140 %test plot check if encoder minimums are in the right spot
141 overlayPlot(encoder, 'Encoder', 'mm', plotTimeVector, encoder, ink);
142
143 %% PHASE PRESSURE, LOAD, & ENCODER VALUES
144 phaseDuration = 0.152; % s (152 ms)
145 indexDuration = floor(phaseDuration/wf_increment); % # of samples in 152 ms
146 allCyclesPhasePres = zeros(numOfPhases,1); %preallocate variable for storing phase pressures
147 allCyclesPhaseLoad = zeros(numOfPhases,1); %preallocate variable for storing phase loads
148 allCyclesPhaseEnco = zeros(numOfPhases,1); %preallocate variable for storing phase encoder displacements
149
```

```
150 phaseStart = [.076 ... %timestamp for the beginning of Phase 1 (obtain from MRI header files and timing
schematic)
151 .355 ... %timestamp for the beginning of Phase 2
152 .634 ... %timestamp for the beginning of Phase 3
153 .913 ... %timestamp for the beginning of Phase 4
154 1.192 ... %timestamp for the beginning of Phase 5
155 1.471 ... %timestamp for the beginning of Phase 6
156 1.750 ... %timestamp for the beginning of Phase 7
157 2.029 ... %timestamp for the beginning of Phase 8
158 2.308 ... %timestamp for the beginning of Phase 9
159 2.587 ... %timestamp for the beginning of Phase 10
160 2.866 ... %timestamp for the beginning of Phase 11
161 3.145 ... %timestamp for the beginning of Phase 12
162 3.424 ... %timestamp for the beginning of Phase 13
163 3.703 ... %timestamp for the beginning of Phase 14
164 3.982 ... %timestamp for the beginning of Phase 15
165 4.261 ... %timestamp for the beginning of Phase 16
166 4.540]; %timestamp for the beginning of (dummy Phase 17, to make a while loop work properly
later).
167
168 presSum = zeros(1,5.5*sampleFrequency); %preallocate for storing all pressure samples before phase averaging
169 loadSum = zeros(1,5.5*sampleFrequency); %.....load.....
170 encoSum = zeros(1,5.5*sampleFrequency); %.....encoder....
171
172 %Get calibration curve coefficients. For Evan's testing, the 7mm verification curve was sufficient for all tests.
173 %The referenced file is made by the Matlab script: "processtdmsVEfvp.m", fvp = "force vs pressure" calibrations.
174 %Make sure the maximum pressure of the calibration curve is higher than the maximum test pressure. If not, use the
175 %next highest calibration (9mm, in this case). The 7 mm test is good up to, 190 kPa.
176 curveCoefficients = load('DataOut/7mmCurveCoefficients.txt','-ascii');
177 curveLoad = curveCoefficients(1:5);
178 curveUnload= curveCoefficients(6:10);
179
180 %Find the time point at which the HyPSTR motor turns around (switches from loading to unloading) and write down the
181 %pressure and load. Calculate the difference between these values and the "turnaround" values of the calibration
182 %curve that you are using (7 mm calibration curve in this case). This will shift the unloading calibration curve
183 %appropriately to model the behavior of a calibration test that only reaches the maximum pressure seen in the
184 %MRI test data. See appropriate figures in the Modifications and Additions thesis chapter.
185 corrections = load('DataOut/customCalCorrection.txt','-ascii');
186 presCorr = 6.041246049772525;
187 loadCorr = 10.415936001919533;
188
189 %average all cycles pressure and load data. It looks tricky, but really just matches the MRI timing schematic
190 %(when image acquisition is occuring vs when it is not) with the LabVIEW data. Negative force values are rewritten =
0 N.
191 for a=11:1:length(encoder.mins) %skip first 10 cycles for preconditioning
192
193 loadCounter = 0;
194 unloadCounter = 0;
195
196 for b=1:1:encoder.minsIndex(a+1)-encoder.minsIndex(a)
197
198 shiftedIndex = encoder.minsIndex(a)+b-1+round(ppuDelay+driftDelay*(a-1));
199
200 presSum(b) = presSum(b) + presFilt.smooth(shiftedIndex);
201 encoSum(b) = encoSum(b) + encoder.data(shiftedIndex);
202
203
204 if b+round(ppuDelay+driftDelay*(a-1))<=(1/2) * (encoder.minsIndex(a+1)-encoder.minsIndex(a))
```

```
addLoad = polyval(curveLoad,presFilt.smooth(shiftedIndex));
if addLoad <0
addLoad = 0;
end

loadSum(b) = loadSum(b) + addLoad;
loadCounter = loadCounter+1;

elseif b+round(ppuDelay+driftDelay*(a-1)) >= (encoder.minsIndex(a+1)-encoder.minsIndex(a))
addLoad = polyval(curveLoad,presFilt.smooth(shiftedIndex));
if addLoad <0
addLoad = 0;
end

loadSum(b) = loadSum(b) + addLoad;
loadCounter = loadCounter+1;

else
addLoad = polyval(curveUnload,presFilt.smooth(shiftedIndex)+presCorr)-loadCorr;
if addLoad <0
addLoad = 0;
end

loadSum(b) = loadSum(b) + addLoad;
unloadCounter = unloadCounter+1;

end
end
end

presAvg = presSum / (length(encoder.mins)-10); %subtract 10 cycles for preconditioning
loadAvg = loadSum / (length(encoder.mins)-10);
encoAvg = encoSum / (length(encoder.mins)-10);

%figure prep for the time stamp plots below.
figure,
ylim([-50 250]),
hold on;

for b=1:1:16
%pressure
phasePresAll = presAvg(floor(phaseStart(b)*sampleFrequency):floor(phaseStart(b)*sampleFrequency) +
indexDuration);
phasePresAvg = mean(phasePresAll);
allCyclesPhasePres(b) = allCyclesPhasePres(b)+phasePresAvg;

%load
phaseLoadAll = loadAvg(floor(phaseStart(b)*sampleFrequency):floor(phaseStart(b)*sampleFrequency) +
indexDuration);
phaseLoadAvg = mean(phaseLoadAll);
allCyclesPhaseLoad(b) = allCyclesPhaseLoad(b)+phaseLoadAvg;

%encoder
phaseEncoAll = encoAvg(floor(phaseStart(b)*sampleFrequency):floor(phaseStart(b)*sampleFrequency) +
indexDuration);
phaseEncoAvg = mean(phaseEncoAll);
allCyclesPhaseEnco(b) = allCyclesPhaseEnco(b)+phaseEncoAvg;

```

```
260 %Time Stamp Plot
261 plot((floor(phaseStart(b)*sampleFrequency):floor(phaseStart(b)*sampleFrequency) +
indexDuration)/sampleFrequency,loadAvg(floor(phaseStart(b)*sampleFrequency):floor(phaseStart(b)*sampleFreque
ncy) + indexDuration),'.','MarkerSize',10,'color', ink.b),
262 hold on;
263 plot((floor(phaseStart(b)*sampleFrequency)+indexDuration:floor(phaseStart(b+1)*sampleFrequency))/sampleFrequ
ency,loadAvg(floor(phaseStart(b)*sampleFrequency)+indexDuration:floor(phaseStart(b+1)*sampleFrequency)),'.',
'MarkerSize',10,'color', ink.o),
264 hold on;
265 end
266
267 %Time Stamp Plot (more)
268 plot((1:phaseStart(1)*sampleFrequency)/sampleFrequency, loadAvg(1:phaseStart(1)*sampleFrequency), '.',
'MarkerSize',10, 'color', ink.r),
269 hold on;
270 plot((phaseStart(17)*sampleFrequency:5*sampleFrequency)/sampleFrequency,loadAvg(phaseStart(b+1)*sampleFreque
ncy:5*sampleFrequency),'.','MarkerSize', 10, 'color',ink.s),
270
plot((phaseStart(17)*sampleFrequency:5*sampleFrequency)/sampleFrequency,loadAvg(phaseStart(b+1)*sampleFreque
ncy:5*sampleFrequency),'.','MarkerSize', 10, 'color',ink.s),
271 hold off;
272
273 %plot calibration curves over the pressure data for sanity check
274 xplot = (-9:1:205);
275 yplot = polyval(curveLoad,xplot);
276 yplot2= polyval(curveUnload,xplot);
277
278 plot(presFilt.smooth(1:100:end),loadFilt.smooth(1:100:end), '+', 'MarkerSize',10,'color', ink.o);
279 hold on;
280 figure,
281 plot(xplot,yplot,'.','MarkerSize', 10, 'color', ink.b),
282 hold on;
283 plot(xplot,yplot2,'.','MarkerSize', 10, 'color', ink.g),
284 hold on;
285 plot(xplot-presCorr,yplot2-loadCorr,'.','MarkerSize', 10, 'color', ink.y),
286 hold on;
287
288 %plot phase data points on top of calibration curve plot
289 plot(allCyclesPhasePres,allCyclesPhaseLoad,'.','MarkerSize',20,'color', 'black'),
290 hold off;
291
292 %% EXPORT DATA
293 %display average phase values for pressure, load, and encoder, and time stamp.
294 allCyclesPhasePres
295 allCyclesPhaseLoad
296 allCyclesPhaseEnco
297 phaseTimeStamps = phaseStart(1:16)+phaseDuration/2; %calculates center time of each phase
298
299 %output average HyPSTR frequency
300 avgFreq = (length(encoder.maxs)-1) / (plotTimeVector(encoder.maxsIndex(end)) -
plotTimeVector(encoder.maxsIndex(1)))
301 avgPeriod = 1/avgFreq
302 commonTimeVector = (1:1:avgPeriod*sampleFrequency)/sampleFrequency;
303
304 %write phase data to files
305 %single average value for each variable for all 16 phases. Most useful output.
306 fidPhases = fopen('DataOut/mri16PhaseValues.txt','w');
307 fprintf(fidPhases,'%10.15f %10.15f %10.15f %10.15f\n', [phaseTimeStamps; allCyclesPhasePres';
```

```
allCyclesPhaseLoad'; allCyclesPhaseEnco']);
fclose(fidPhases);

%average values (at 2500 Hz sampling) for each phase over the course of an entire MRI test.
fidPres = fopen('DataOut/mriAvgPres.txt','w');
fidLoad = fopen('DataOut/mriAvgLoad.txt','w');
fidEnco = fopen('DataOut/mriAvgEnco.txt','w');

fprintf(fidPres,'%10.15f %10.15f\n', [commonTimeVector; presAvg(1:length(commonTimeVector))]);
fprintf(fidLoad,'%10.15f %10.15f\n', [commonTimeVector; loadAvg(1:length(commonTimeVector))]);
fprintf(fidEnco,'%10.15f %10.15f\n', [commonTimeVector; encoAvg(1:length(commonTimeVector))]);

fclose(fidPres);
fclose(fidLoad);
fclose(fidEnco);

%% some spare plots for error checking. Turn off if not needed. (overlay all, lvdt vs encoder, pressure vs load cell,
etc)
%make filtered, smoothed, max/min plots for specified data series
overlayPlot(presFilt, 'Pressure', 'kPa', plotTimeVector, encoder, ink);

%plot encoder vs pressure to find out where the unloading begins
figure;
plot(encoAvg, loadAvg,'.', 'MarkerSize', 10, 'color', ink.i);

end

%% LOCAL FUNCTIONS
%truncateAndGetEnds()
function [data, initial, final] = truncateAndGetEnds(data, iStart, iEnd, avgWin)
%avgs a range of points before/after the user selected start point
initial = mean(data((iStart-avgWin):iStart));
final = mean(data(iEnd:(iEnd+avgWin)));

%remove data outside of the selected start/stop range for each series
data = data(iStart:iEnd);
end


%findMinMaxSet()
function [mins, maxs, minsIndex, maxsIndex] = findMinMaxSet(data, cFreq, sFreq)
quarterWave = ((1/cFreq)*0.25)*sFreq;
tempStart = quarterWave;
halfWave = 2*quarterWave;

a = 1;

while tempStart+halfWave <= length(data)
[maxs(a), maxsIndex(a)] = max(data(tempStart:tempStart+halfWave));
maxsIndex(a) = maxsIndex(a) + tempStart - 1;

if tempStart+2*halfWave >= length(data)
[mins(a), minsIndex(a)] = min(data(tempStart+halfWave:end));
minsIndex(a) = minsIndex(a) + tempStart + halfWave - 1;
tempStart = minsIndex(a) + quarterWave;
else
[mins(a), minsIndex(a)] = min(data(tempStart+halfWave:tempStart+2*halfWave));
minsIndex(a) = minsIndex(a) + tempStart + halfWave - 1;
```

```
364 tempStart = minsIndex(a) + quarterWave;
365 end
366 a = a+1;
367 end
368 end
369
370
371 %overlayPlot()
372 function overlayPlot(dataSeries, plotTitle, plotUnit, plotTime, enc, ink)
373 %scales the encoder trace plot to be 10% of the data magnitude
374 encTrace = enc.data*10;
375
376 %plot filtered and smoothed data series
377 f = figure('visible', 'on');
378 hold on;
379 plot(plotTime, dataSeries.data,'.', 'color', ink.b, 'MarkerSize', 7);
380 plot(plotTime, dataSeries.smooth,'.', 'color', ink.g, 'MarkerSize', 7);
381
382 %plot data series maximums and minimums
383 plot(plotTime(dataSeries.maxsIndex), dataSeries.maxs,'.', 'color', ink.o, 'MarkerSize', 10);
384 plot(plotTime(dataSeries.minsIndex), dataSeries.mins,'o', 'color', ink.r, 'MarkerSize', 7);
385
386 %plot encoder trace as a visual reference for system stages
387 plot(plotTime, encTrace,'--', 'color', ink.s);
388 hold off;
389
390 %add plot specific title and axes
391 legend('filtered','smoothed','maximums','minimums','encoder trace'),
392 title([plotTitle ' - ' 'overlay summary']),
393 xlabel('time (s)'),
394 ylabel([plotTitle ' (' plotUnit ')']);
395 end
396
397
398 %% convertTDMS functions from MatLab File Exchange (file copied and pasted)
399 % http://www.mathworks.com/matlabcentral/fileexchange/28771-converttdms-v9
400
401 function [ConvertedData,ConvertVer,ChanNames,GroupNames,ci]=convertTDMS(varargin)
402 %Function to load LabView TDMS data file(s) into variables in the MATLAB workspace.
403 %An *.MAT file can also be created. If called with one input, the user selects
404 %a data file.
405 %
406 % TDMS format is based on information provided by National Instruments at:
407 % http://zone.ni.com/devzone/cda/tut/p/id/5696
408 %
409 % [ConvertedData,ConvertVer,ChanNames]=convertTDMS(SaveConvertedFile,filename)
410 %
411 % Inputs:
412 % SaveConvertedFile (required) - Logical flag (true/false) that
413 % determines whether a MAT file is created. The MAT file's name
414 % is the same as 'filename' except that the 'TDMS' file extension is
415 % replaced with 'MAT'. The MAT file is saved in the same folder
416 % and will overwrite an existing file without warning. The
417 % MAT file contains all the output variables.
418 %
419 % filename (optional) - Filename (fully defined) to be converted.
420 % If not supplied, the user is provided a 'File Open' dialog box
421 % to navigate to a file. Can be a cell array of files for bulk
```

```
422 % conversion.
423 %
424 % Outputs:
425 % ConvertedData (required) - Structure of all of the data objects.
426 % ConvertVer (optional) - Version number of this function.
427 % ChanNames (optional) - Cell array of channel names
428 % GroupNames (optional) - Cell array of group names
429 %
430 %
431 %'ConvertedData' is a structure with 'FileName', 'FileFolder', 'SegTDMSVerNum',
432 %'NumOfSegments' and 'Data' fields'. The 'Data' field is a structure.
433 %
434 %'ConvertedData.SegTDMSVerNum' is a vector of the TDMS version number for each
435 %segment.
436 %
437 %'ConvertedData.Data' is a structure with 'Root' and 'MeasuredData' fields.
438 %
439 %'ConvertedData.Data.Root' is a structure with 'Name' and 'Property' fields.
440 %The 'Property' field is also a structure; it contains all the specified properties
441 %(1 entry for each 'Property) for the 'Root' group. For each 'Property' there are
442 %'Name' and 'Value' fields. To display a list of all the property names, input
443 %'{ConvertedData.Data.Root.Property.Name}" in the Command Window.
444 %
445 %'ConvertedData.Data.MeasuredData' is a structure containing all the channel/group
446 %information. For each index (for example, 'ConvertedData.Data.MeasuredData(1)'),
447 %there are 'Name', 'Data' and 'Property' fields. The list of channel names can
448 %be displayed by typing 'ChanNames' in the Command Window. Similarly, the list
449 %of group names can be displayed by typing 'GroupNames' in the Command Window.
450 %The 'Property' field is also a structure; it contains all the specified properties
451 %for that index (1 entry in the structure for each 'Property'). Any LabView waveform
452 %attributes ('wf_start_time', 'wf_start_offset', 'wf_increment' and 'wf_samples') that
453 %may exist are also included in the properties. For each 'Property' there are 'Name'
454 %and 'Value' fields. To display a list of all the property names, input
455 %'{ConvertedData.Data.MeasuredData(#).Property.Name}" in the Command Window
456 %where '#' is the index of interest.
457 %
458
459 %-------------------------------------------------------------------------
460 %Brad Humphreys - v1.0 2008-04-23
461 %ZIN Technologies
462 %-------------------------------------------------------------------------
463
464 %-------------------------------------------------------------------------
465 %Brad Humphreys - v1.1 2008-07-03
466 %ZIN Technologies
467 %-Added abilty for timestamp to be a raw data type, not just meta data.
468 %-Addressed an issue with having a default nsmaples entry for new objects.
469 %-Added Error trap if file name not found.
470 %-Corrected significant problem where it was assumed that once an object
471 % existsed, it would in in every subsequent segement. This is not true.
472 %-------------------------------------------------------------------------
473
474 %-------------------------------------------------------------------------
475 %Grant Lohsen - v1.2 2009-11-15
476 %Georgia Tech Research Institute
477 %-Converts TDMS v2 files
478 %Folks, it's not pretty but I don't have time to make it pretty. Enjoy.
479 %-------------------------------------------------------------------------
```

```
480
481 %------------------------------------------------------------------------
482 %Jeff Sitterle - v1.3 2010-01-10
483 %Georgia Tech Research Institute
484 %Modified to return all information stored in the TDMS file to inlcude
485 %name, start time, start time offset, samples per read, total samples, unit
486 %description, and unit string. Also provides event time and event
487 %description in text form
488 %Vast speed improvement as save was the previous longest task
489 %------------------------------------------------------------------------
490
491 %------------------------------------------------------------------------
492 %Grant Lohsen - v1.4 2009-04-15
493 %Georgia Tech Research Institute
494 %Reads file header info and stores in the Root Structure.
495 %------------------------------------------------------------------------
496
497 %------------------------------------------------------------------------
498 %Robert Seltzer - v1.5 2010-07-14
499 %BorgWarner Morse TEC
500 %-Tested in MATLAB 2007b and 2010a.
501 %-APPEARS to now be compatible with TDMS version 1.1 (a.k.a 4712) files;
502 % although, this has not been extensively tested. For some unknown
503 % reason, the version 1.2 (4713) files process noticeably faster. I think
504 % that it may be related to the 'TDSm' tag.
505 %-"Time Stamp" data type was not tested.
506 %-"Waveform" fields was not tested.
507 %-Fixed an error in the 'LV2MatlabDataType' function where LabView data type
508 % 'tdsTypeSingleFloat' was defined as MATLAB data type 'float64' . Changed
509 % to 'float32'.
510 %-Added error trapping.
511 %-Added feature to count the number of segments for pre-allocation as
512 % opposed to estimating the number of segments.
513 %-Added option to save the data in a MAT file.
514 %-Fixed "invalid field name" error caused by excessive string lengths.
515 %------------------------------------------------------------------------
516
517 %------------------------------------------------------------------------
518 %Robert Seltzer - v1.6 2010-09-01
519 %BorgWarner Morse TEC
520 %-Tested in MATLAB 2010a.
521 %-Fixed the "Coversion to cell from char is not possible" error found
522 % by Francisco Botero in version 1.5.
523 %-Added capability to process both fragmented or defragmented data.
524 %-Fixed the "field" error found by Lawrence.
525 %------------------------------------------------------------------------
526
527 %------------------------------------------------------------------------
528 %Christian Buxel - V1.7 2010-09-17
529 %RWTH Aachen
530 %-Tested in Matlab2007b.
531 %-Added support for german umlauts (ï¿½,ï¿½,ï¿½,ï¿½,ï¿½,ï¿½,ï¿½) in 'propsName'
532 %------------------------------------------------------------------------
533
534 %------------------------------------------------------------------------
535 %Andrï¿½ Rï¿½egg - V1.7 2010-09-29
536 %Supercomputing Systems AG
537 %-Tested in MATLAB 2006a & 2010b
```

```
538 %-Make sure that data can be loaded correctly independently of character
539 % encoding set in matlab.
540 %-Fixed error if object consists of several segments with identical segment
541 % information (if rawdataindex==0, not all segments were loaded)
542 %--------------------------------------------------------------------------
543
544 %--------------------------------------------------------------------------
545 %Robert Seltzer - v1.7 2010-09-30
546 %BorgWarner Morse TEC
547 %-Tested in MATLAB 2010b.
548 %-Added 'error trapping' to the 'fixcharformatlab' function for group and
549 % channel names that contain characters that are not 'A' through 'Z',
550 % 'a' through 'z', 0 through 9 or underscore. The characters are replaced
551 % with an underscore and a message is written to the Command Window
552 % explaining to the user what has happened and how to fix it. Only tested
553 % with a very limited number of "special" characters.
554 %--------------------------------------------------------------------------
555
556 %--------------------------------------------------------------------------
557 %Robert Seltzer - v1.8 2010-10-12
558 %BorgWarner Morse TEC
559 %-As a result of an error found by Peter Sulcs when loading data with very
560 % long channel names, I have re-written the sections of the function that
561 % creates the channel and property names that are used within the body of
562 % the function to make them robust against long strings and strings
563 % containing non-UTF8 characters. The original channel and property
564 % names (no truncation or character replacement) are now retained and
565 % included in the output structure. In order to implement this improvement,
566 % I added a 'Property' field as a structure to the 'ConvertedData' output
567 % structure.
568 %-Added a more detailed 'help' description ('doc convertTDMS') of the
569 % returned structure.
570 %-List of channel names added as an output parameter of the function.
571 %-Corrected an error in the time stamp converion. It was off by exactly
572 % 1 hour.
573 %-Tested in MATLAB 2010b with a limited number of diverse TDMS files.
574 %--------------------------------------------------------------------------
575
576 %--------------------------------------------------------------------------
577 %Robert Seltzer - v1.8 2010-10-19
578 %BorgWarner Morse TEC
579 %-Fixed an error found by Terenzio Girotto with the 'save' routine.
580 %--------------------------------------------------------------------------
581
582 %--------------------------------------------------------------------------
583 %Robert Seltzer - v1.8 2010-10-25
584 %BorgWarner Morse TEC
585 %-Fixed an error with channels that contain no data. Previously, if a
586 % channel contained no data, then it was not passed to the output structure
587 % even if it did contain properties.
588 %-Added 'GroupNames' as an optional output variable.
589 %-Fixed an error with capturing the properties of the Root object
590 %--------------------------------------------------------------------------
591
592
593 %--------------------------------------------------------------------------
594 %Philip Top - v1.9 2010-11-09
595 %John Breneman
```

```
596 %-restructured code as function calls
597 %-seperated metadata reads from data reads
598 %-preallocated space for SegInfo with two pass file read
599 %-preallocated index information and defined segdataoffset for each segment
600 %-preallocate space for data for speedup in case of fragmented files
601 %-used matlab input parser instead of nargin switch
602 %-vectorized timestamp reads for substantial speedup
603 %------------------------------------------------------------------------
604
605 %------------------------------------------------------------------------
606 %Robert Seltzer - v1.9 2010-11-10
607 %BorgWarner Morse TEC
608 %-Fixed an error error in the 'offset' calculation for strings
609 %------------------------------------------------------------------------
610
611 %------------------------------------------------------------------------
612 %Philip Top - v1.95 2011-5-10
613 %Fix Bug with out of order file segments
614 %Fix some issues with string array reads for newer version files,
615 %------------------------------------------------------------------------
616
617 %Initialize outputs
618 ConvertVer='1.95'; %Version number of this conversion function
619 ConvertedData=[];
620
621 p=inputParser();
622
623 p.addRequired('SaveConvertedFile',@(x) islogical(x)||(ismember(x,[0,1])));
624 p.addOptional('filename','',@(x) iscell(x)||exist(x,'file'));
625 p.parse(varargin{:});
626
627 filename=p.Results.filename;
628 SaveConvertedFile=p.Results.SaveConvertedFile;
629
630 if isempty(filename)
631
632 %Prompt the user for the file
633 [filename,pathname]=uigetfile({'*.tdms','All Files (*.tdms)'},'Choose a TDMS File');
634 if filename==0
635 return
636 end
637 filename=fullfile(pathname,filename);
638 end
639
640
641 if iscell(filename)
642 %For a list of files
643 infilename=filename;
644 else
645 infilename=cellstr(filename);
646 end
647
648 for fnum=1:numel(infilename)
649
650 if ~exist(infilename{fnum},'file')
651 e=errordlg(sprintf('File "%s" not found.',infilename{fnum}),'File Not Found');
652 uiwait(e)
653 return
```

```
654 end
655
656 FileNameLong=infilename{fnum};
657 [pathstr,name,ext]=fileparts(FileNameLong);
658 FileNameShort=sprintf('%s%s',name,ext);
659 FileNameNoExt=name;
660 FileFolder=pathstr;
661
662 if fnum==1
663 fprintf('\n\n')
664 end
665 fprintf('Converting "%s"...',FileNameShort)
666
667 fid=fopen(FileNameLong);
668
669 if fid==-1
670 e=errordlg(sprintf('Could not open "%s".',FileNameLong),'File Cannot Be Opened');
671 uiwait(e)
672 fprintf('\n\n')
673 return
674 end
675
676 [SegInfo,NumOfSeg]=getSegInfo(fid);
677 channelinfo=getChannelInfo(fid,SegInfo,NumOfSeg);
678 ob=getData(fid,channelinfo);
679 fclose(fid);
680
681 %Assign the outputs
682 ConvertedData(fnum).FileName=FileNameShort;
683 ConvertedData(fnum).FileFolder=FileFolder;
684
685 ConvertedData(fnum).SegTDMSVerNum=SegInfo.vernum;
686 ConvertedData(fnum).NumOfSegments=NumOfSeg;
687 [ConvertedData(fnum).Data,CurrGroupNames]=postProcess(ob,channelinfo);
688
689 GroupNames(fnum)={CurrGroupNames};
690
691 TempChanNames={ConvertedData(fnum).Data.MeasuredData.Name};
692 TempChanNames(strcmpi(TempChanNames,'Root'))=[];
693 ChanNames(fnum)={sort(setdiff(TempChanNames',CurrGroupNames))};
694 if SaveConvertedFile
695 MATFileNameShort=sprintf('%s.mat',FileNameNoExt);
696 MATFileNameLong=fullfile(FileFolder,MATFileNameShort);
697 try
698 save(MATFileNameLong,'ConvertedData','ConvertVer','ChanNames')
699 fprintf('\n\nConversion complete (saved in "%s").\n\n',MATFileNameShort)
700 catch exception
701 fprintf('\n\nConversion complete (could not save "%s").\n\t%s:
%s\n\n',MATFileNameShort,exception.identifier,...
702 exception.message)
703 end
704 else
705 fprintf('\n\nConversion complete.\n\n')
706 end
707 end
708 ci=channelinfo;
709 end
710
```

```
711 function [SegInfo,NumOfSeg]=getSegInfo(fid)
712 %Count the number of segments. While doing the count, also include error trapping.
713
714 %Find the end of the file
715 fseek(fid,0,'eof');
716 eoff=ftell(fid);
717 frewind(fid);
718
719 segCnt=0;
720 CurrPosn=0;
721 LeadInByteCount=28; %From the National Instruments web page (http://zone.ni.com/devzone/cda/tut/p/id/5696) under
722 %the 'Lead In' description on page 2: Counted the bytes shown in the table.
723 while (ftell(fid) ~= eoff)
724
725 Ttag=fread(fid,1,'uint8');
726 Dtag=fread(fid,1,'uint8');
727 Stag=fread(fid,1,'uint8');
728 mtag=fread(fid,1,'uint8');
729
730 if Ttag==84 && Dtag==68 && Stag==83 && mtag==109
731 %Apparently, this sequence of numbers identifies the start of a new segment.
732
733 segCnt=segCnt+1;
734
735 %ToC Field
736 ToC=fread(fid,1,'uint32');
737
738 %TDMS format version number
739 vernum=fread(fid,1,'uint32');
740
741 %From the National Instruments web page (http://zone.ni.com/devzone/cda/tut/p/id/5696) under the 'Lead In'
742 %description on page 2:
743 %The next eight bytes (64-bit unsigned integer) describe the length of the remaining segment (overall length of
the
744 %segment minus length of the lead in). If further segments are appended to the file, this number can be used to
745 %locate the starting point of the following segment. If an application encountered a severe problem while
writing
746 %to a TDMS file (crash, power outage), all bytes of this integer can be 0xFF. This can only happen to the last
747 %segment in a file.
748 nlen=fread(fid,1,'uint64');
749 if (nlen>2^63)
750 break;
751 else
752
753 segLength=nlen;
754 end
755 TotalLength=segLength+LeadInByteCount;
756 CurrPosn=CurrPosn+TotalLength;
757
758 status=fseek(fid,CurrPosn,'bof'); %Move to the beginning position of the next segment
759 if (status<0)
760 warning('file glitch');
761 break;
762 end
763 end
764
765 end
766
```

```
767 frewind(fid);
768
769 CurrPosn=0;
770 SegInfo.SegStartPosn=zeros(segCnt,1);
771 SegInfo.MetaStartPosn=zeros(segCnt,1);
772 SegInfo.DataStartPosn=zeros(segCnt,1);
773 SegInfo.vernum=zeros(segCnt,1);
774 SegInfo.DataLength=zeros(segCnt,1);
775 segCnt=0;
776 while (ftell(fid) ~= eoff)
777
778 Ttag=fread(fid,1,'uint8');
779 Dtag=fread(fid,1,'uint8');
780 Stag=fread(fid,1,'uint8');
781 mtag=fread(fid,1,'uint8');
782
783 if Ttag==84 && Dtag==68 && Stag==83 && mtag==109
784 %Apparently, this sequence of numbers identifies the start of a new segment.
785
786 segCnt=segCnt+1;
787
788 if segCnt==1
789 StartPosn=0;
790 else
791 StartPosn=CurrPosn;
792 end
793
794 %ToC Field
795 ToC=fread(fid,1,'uint32');
796 kTocMetaData=bitget(ToC,2);
797 kTocNewObject=bitget(ToC,3);
798 kTocRawData=bitget(ToC,4);
799 kTocInterleavedData=bitget(ToC,6);
800 kTocBigEndian=bitget(ToC,7);
801
802 if kTocInterleavedData
803 e=errordlg(sprintf(['Seqment %.0f within "%s" has interleaved data which is not supported with this '...
804 'function (%s.m).'],segCnt,TDMSFileNameShort,mfilename),'Interleaved Data Not Supported');
805 fclose(fid);
806 uiwait(e)
807 end
808
809 if kTocBigEndian
810 e=errordlg(sprintf(['Seqment %.0f within "%s" uses the big-endian data format which is not supported '...
811 'with this function (%s.m).'],segCnt,TDMSFileNameShort,mfilename),'Big-Endian Data Format Not
Supported');
812 fclose(fid);
813 uiwait(e)
814 end
815
816 %TDMS format version number
817 vernum=fread(fid,1,'uint32');
818 if ~ismember(vernum,[4712,4713])
819 e=errordlg(sprintf(['Seqment %.0f within "%s" used LabView TDMS file format version %.0f which is not '...
820 'supported with this function (%s.m).'],segCnt,TDMSFileNameShort,vernum,mfilename),...
821 'TDMS File Format Not Supported');
822 fclose(fid);
823 uiwait(e)
```

```
824 end
825
826 %From the National Instruments web page (http://zone.ni.com/devzone/cda/tut/p/id/5696) under the 'Lead In'
827 %description on page 2:
828 %The next eight bytes (64-bit unsigned integer) describe the length of the remaining segment (overall length of
the
829 %segment minus length of the lead in). If further segments are appended to the file, this number can be used to
830 %locate the starting point of the following segment. If an application encountered a severe problem while
writing
831 %to a TDMS file (crash, power outage), all bytes of this integer can be 0xFF. This can only happen to the last
832 %segment in a file.
833 segLength=fread(fid,1,'uint64');
834 metaLength=fread(fid,1,'uint64');
835 if (segLength>2^63)
836 fseek(fid,0,'eof');
837 flen=ftell(fid);
838 segLength=flen-LeadInByteCount-TotalLength;
839 TotalLength=segLength+LeadInByteCount;
840 else
841 TotalLength=segLength+LeadInByteCount;
842 CurrPosn=CurrPosn+TotalLength;
843 fseek(fid,CurrPosn,'bof'); %Move to the beginning position of the next segment
844 end
845
846
847 SegInfo.SegStartPosn(segCnt)=StartPosn;
848 SegInfo.MetaStartPosn(segCnt)=StartPosn+LeadInByteCount;
849 SegInfo.DataStartPosn(segCnt)=SegInfo.MetaStartPosn(segCnt)+metaLength;
850 SegInfo.DataLength(segCnt)=segLength-metaLength;
851 SegInfo.vernum(segCnt)=vernum;
852
853 end
854
855 end
856 NumOfSeg=segCnt;
857 end
858
859 function index=getChannelInfo(fid,SegInfo,NumOfSeg)
860 %Initialize variables for the file conversion
861 index=struct();
862 objOrderList={};
863 for segCnt=1:NumOfSeg
864
865 fseek(fid,SegInfo.SegStartPosn(segCnt)+4,'bof');
866
867 %Ttag=fread(fid,1,'uint8');
868 %Dtag=fread(fid,1,'uint8');
869 %Stag=fread(fid,1,'uint8');
870 %mtag=fread(fid,1,'uint8');
871
872 %ToC Field
873 ToC=fread(fid,1,'uint32');
874 kTocMetaData=bitget(ToC,2);
875 kTocNewObjectList=bitget(ToC,3);
876 kTocRawData=bitget(ToC,4);
877 %kTocInterleavedData=bitget(ToC,6);
878 %kTocBigEndian=bitget(ToC,7);
879
```

```
880 segVersionNum=fread(fid,1,'uint32'); %TDMS format version
number for this segment
881
882 segLength=fread(fid,1,'uint64');
883
884 metaLength=fread(fid,1,'uint64');
885 offset=0;
886 %Process Meta Data
887 if (kTocNewObjectList==0) %use the object list from the previous segment
888 fnm=fieldnames(index);
889 for kk=1:length(fnm)
890 ccnt=index.(fnm{kk}).rawdatacount;
891 if (ccnt>0)
892 if (index.(fnm{kk}).index(ccnt)==segCnt-1)
893 ccnt=ccnt+1;
894 index.(fnm{kk}).rawdatacount=ccnt;
895 index.(fnm{kk}).datastartindex(ccnt)=SegInfo.DataStartPosn(segCnt);
896 index.(fnm{kk}).arrayDim(ccnt)=index.(fnm{kk}).arrayDim(ccnt-1);
897 index.(fnm{kk}).nValues(ccnt)=index.(fnm{kk}).nValues(ccnt-1);
898 index.(fnm{kk}).byteSize(ccnt)=index.(fnm{kk}).byteSize(ccnt-1);
899 index.(fnm{kk}).index(ccnt)=segCnt;
900 index.(fnm{kk}).rawdataoffset(ccnt)=index.(fnm{kk}).rawdataoffset(ccnt-1);
901 end
902 end
903 end
904 end
905
906 if kTocMetaData
907 numObjInSeg=fread(fid,1,'uint32');
908 if (kTocNewObjectList)
909 objOrderList=cell(numObjInSeg,1);
910 end
911 for q=1:numObjInSeg
912
913 obLength=fread(fid,1,'uint32'); %Get the length
of the objects name
914 ObjName=convertToText(fread(fid,obLength,'uint8'))'; %Get the objects name
915
916 if strcmp(ObjName,'/')
917 long_obname='Root';
918 else
919 long_obname=ObjName;
920
921 %Delete any apostrophes. If the first character is a slash (forward or backward), delete it too.
922 long_obname(strfind(long_obname,''''))=[];
923 if strcmpi(long_obname(1),'/') || strcmpi(long_obname(1),'\')
924 long_obname(1)=[];
925 end
926 end
927 newob=0;
928 %Create object's name. Use a generic field name to avoid issues with strings that are too long and/or
929 %characters that cannot be used in MATLAB variable names. The actual channel name is retained for the final
930 %output structure.
931 if exist('ObjNameList','var')
932 %Check to see if the object already exists
933 NameIndex=find(strcmpi({ObjNameList.LongName},long_obname)==1,1,'first');
934 if isempty(NameIndex)
935 newob=1;
```

```
936 %It does not exist, so create the generic name field name
937 ObjNameList(end+1).FieldName=sprintf('Object%.0f',numel(ObjNameList)+1);
938 ObjNameList(end).LongName=long_obname;
939 NameIndex=numel(ObjNameList);
940 end
941 else
942 %No objects exist, so create the first one using a generic name field name.
943 ObjNameList.FieldName='Object1';
944 ObjNameList.LongName=long_obname;
945 NameIndex=1;
946 newob=1;
947 end
948 %Assign the generic field name
949 obname=ObjNameList(NameIndex).FieldName;
950
951 %Create the 'index' structure
952 if (~isfield(index,obname))
953 index.(obname).name=obname;
954 index.(obname).long_name=long_obname;
955 index.(obname).rawdatacount=0;
956 index.(obname).datastartindex=zeros(NumOfSeg,1);
957 index.(obname).arrayDim=zeros(NumOfSeg,1);
958 index.(obname).nValues=zeros(NumOfSeg,1);
959 index.(obname).byteSize=zeros(NumOfSeg,1);
960 index.(obname).index=zeros(NumOfSeg,1);
961 index.(obname).rawdataoffset=zeros(NumOfSeg,1);
962 index.(obname).multiplier=ones(NumOfSeg,1);
963 index.(obname).skip=zeros(NumOfSeg,1);
964 end
965 if (kTocNewObjectList)
966 objOrderList{q}=obname;
967 else
968 if ~ismember(obname,objOrderList)
969 objOrderList{end+1}=obname;
970 end
971 end
972 %Get the raw data Index
973 rawdataindex=fread(fid,1,'uint32');
974
975 if rawdataindex==0
976 if segCnt==0
977 e=errordlg(sprintf('Seqment %.0f within "%s" has "rawdataindex" value of 0 (%s.m).',segCnt,...
978 TDMSFileNameShort,mfilename),'Incorrect "rawdataindex"');
979 uiwait(e)
980 end
981 if kTocRawData
982 if (kTocNewObjectList)
983 ccnt=index.(obname).rawdatacount+1;
984 else
985 ccnt=index.(obname).rawdatacount;
986 end
987 index.(obname).rawdatacount=ccnt;
988 index.(obname).datastartindex(ccnt)=SegInfo.DataStartPosn(segCnt);
989 index.(obname).arrayDim(ccnt)=index.(obname).arrayDim(ccnt-1);
990 index.(obname).nValues(ccnt)=index.(obname).nValues(ccnt-1);
991 index.(obname).byteSize(ccnt)=index.(obname).byteSize(ccnt-1);
992 index.(obname).index(ccnt)=segCnt;
993
```

```
994 end
995 elseif rawdataindex+1==2^32
996 %Objects raw data index matches previous index - no changes. The root object will always have an
997 %'FFFFFFFF' entry
998 if strcmpi(index.(obname).long_name,'Root')
999 index.(obname).rawdataindex=0;
1000 else
1001 %Need to account for the case where an object (besides the 'root') is added that has no data but
reports
1002 %using previous.
1003 if newob
1004 index.(obname).rawdataindex=0;
1005 else
1006 if kTocRawData
1007 if (kTocNewObjectList)
1008 ccnt=index.(obname).rawdatacount+1;
1009 else
1010 ccnt=index.(obname).rawdatacount;
1011 end
1012 index.(obname).rawdatacount=ccnt;
1013 index.(obname).datastartindex(ccnt)=SegInfo.DataStartPosn(segCnt);
1014 index.(obname).arrayDim(ccnt)=index.(obname).arrayDim(ccnt-1);
1015 index.(obname).nValues(ccnt)=index.(obname).nValues(ccnt-1);
1016 index.(obname).byteSize(ccnt)=index.(obname).byteSize(ccnt-1);
1017 index.(obname).index(ccnt)=segCnt;
1018
1019 end
1020 end
1021 end
1022 else
1023 %Get new object information
1024 if (kTocNewObjectList)
1025 ccnt=index.(obname).rawdatacount+1;
1026 else
1027 ccnt=index.(obname).rawdatacount;
1028 if (ccnt==0)
1029 ccnt=1;
1030 end
1031 end
1032 index.(obname).rawdatacount=ccnt;
1033 index.(obname).datastartindex(ccnt)=SegInfo.DataStartPosn(segCnt);
1034 %index(end).lenOfIndexInfo=fread(fid,1,'uint32');
1035
1036 index.(obname).dataType=fread(fid,1,'uint32');
1037 if (index.(obname).dataType~=32)
1038 index.(obname).datasize=getDataSize(index.(obname).dataType);
1039 end
1040 index.(obname).arrayDim(ccnt)=fread(fid,1,'uint32');
1041 index.(obname).nValues(ccnt)=fread(fid,1,'uint64');
1042 index.(obname).index(ccnt)=segCnt;
1043 if index.(obname).dataType==32
1044 %Datatype is a string
1045 index.(obname).byteSize(ccnt)=fread(fid,1,'uint64');
1046 else
1047 index.(obname).byteSize(ccnt)=0;
1048 end
1049
1050 end
```

```
1051
1052 %Get the properties
1053 numProps=fread(fid,1,'uint32');
1054 if numProps>0
1055
1056 if isfield(index.(obname),'PropertyInfo')
1057 PropertyInfo=index.(obname).PropertyInfo;
1058 else
1059 clear PropertyInfo
1060 end
1061 for p=1:numProps
1062 propNameLength=fread(fid,1,'uint32');
1063 switch 1
1064 case 1
1065 PropName=fread(fid,propNameLength,'*uint8')';
1066 PropName=native2unicode(PropName,'UTF-8');
1067 case 2
1068 PropName=fread(fid,propNameLength,'uint8=>char')';
1069 otherwise
1070 end
1071 propsDataType=fread(fid,1,'uint32');
1072
1073 %Create property's name. Use a generic field name to avoid issues with strings that are too long
and/or
1074 %characters that cannot be used in MATLAB variable names. The actual property name is retained for
the
1075 %final output structure
1076 if exist('PropertyInfo','var')
1077 %Check to see if the property already exists for this object. Need to get the existing
'PropertyInfo'
1078 %structure for this object. The 'PropertyInfo' structure is not necessarily the same for every
1079 %object in the data file.
1080 PropIndex=find(strcmpi({PropertyInfo.Name},PropName));
1081 if isempty(PropIndex)
1082 %Is does not exist, so create the generic name field name
1083 propExists=false;
1084 PropIndex=numel(PropertyInfo)+1;
1085 propsName=sprintf('Property%.0f',PropIndex);
1086 PropertyInfo(PropIndex).Name=PropName;
1087 PropertyInfo(PropIndex).FieldName=propsName;
1088 else
1089 %Assign the generic field name
1090 propExists=true;
1091 propsName=PropertyInfo(PropIndex).FieldName;
1092 end
1093 else
1094 %No properties exist for this object, so create the first one using a generic name field name.
1095 propExists=false;
1096 PropIndex=p;
1097 propsName=sprintf('Property%.0f',PropIndex);
1098 PropertyInfo(PropIndex).Name=PropName;
1099 PropertyInfo(PropIndex).FieldName=propsName;
1100 end
1101 dataExists=isfield(index.(obname),'data');
1102
1103 if dataExists
1104 %Get number of data samples for the object in this segment
1105 nsamps=index.(obname).nsamples+1;
```

```
1106 else
1107 nsamps=0;
1108 end
1109
1110 if propsDataType==32
1111 %String data type
1112 PropertyInfo(PropIndex).DataType='String';
1113 propsValueLength=fread(fid,1,'uint32');
1114 propsValue=convertToText(fread(fid,propsValueLength,'uint8=>char'))';
1115 if propExists
1116 if isfield(index.(obname).(propsName),'cnt')
1117 cnt=index.(obname).(propsName).cnt+1;
1118 else
1119 cnt=1;
1120 end
1121 index.(obname).(propsName).cnt=cnt;
1122 index.(obname).(propsName).value{cnt}=propsValue;
1123 index.(obname).(propsName).samples(cnt)=nsamps;
1124 else
1125 if strcmp(index.(obname).long_name,'Root')
1126 %Header data
1127 index.(obname).(propsName).name=index.(obname).long_name;
1128 index.(obname).(propsName).value={propsValue};
1129 index.(obname).(propsName).cnt=1;
1130 else
1131 index.(obname).(propsName).name=PropertyInfo(PropIndex).Name;
1132 index.(obname).(propsName).datatype=PropertyInfo(PropIndex).DataType;
1133 index.(obname).(propsName).cnt=1;
1134 index.(obname).(propsName).value=cell(nsamps,1); %Pre-allocation
1135 index.(obname).(propsName).samples=zeros(nsamps,1); %Pre-allocation
1136 if iscell(propsValue)
1137 index.(obname).(propsName).value(1)=propsValue;
1138 else
1139 index.(obname).(propsName).value(1)={propsValue};
1140 end
1141 index.(obname).(propsName).samples(1)=nsamps;
1142 end
1143 end
1144 else
1145 %Numeric data type
1146 if propsDataType==68
1147 PropertyInfo(PropIndex).DataType='Time';
1148 %Timestamp data type
1149 tsec=fread(fid,1,'uint64')/2^64+fread(fid,1,'uint64'); %time since Jan-1-1904 in
seconds
1150 %R. Seltzer: Not sure why '5/24' (5 hours) is subtracted from the time value. That's how it
was
1151 %coded in the original function I downloaded from MATLAB Central. But I found it to be 1
hour too
1152 %much. So, I changed it to '4/24'.
1153 %propsValue=tsec/86400+695422-5/24; %/864000 convert to days; +695422 days from Jan-0-0000
to Jan-1-1904
1153
%propsValue=tsec/86400+695422-5/24; %/864000 convert to days; +695422 days from Jan-0-0000
to Jan-1-1904
1154 propsValue=tsec/86400+695422-4/24; %/864000 convert to days; +695422 days from Jan-0-0000
to Jan-1-1904
1155 else
```

```
1156 PropertyInfo(PropIndex).DataType='Numeric';
1157 matType=LV2MatlabDataType(propsDataType);
1158 if strcmp(matType,'Undefined')
1159 e=errordlg(sprintf('No MATLAB data type defined for a "Property Data Type" value of
"%.0f".',...
1160 propsDataType),'Undefined Property Data Type');
1161 uiwait(e)
1162 fclose(fid);
1163 return
1164 end
1165 if strcmp(matType,'uint8=>char')
1166 propsValue=convertToText(fread(fid,1,'uint8'));
1167 else
1168 propsValue=fread(fid,1,matType);
1169 end
1170 end
1171 if propExists
1172 cnt=index.(obname).(propsName).cnt+1;
1173 index.(obname).(propsName).cnt=cnt;
1174 index.(obname).(propsName).value(cnt)=propsValue;
1175 index.(obname).(propsName).samples(cnt)=nsamps;
1176 else
1177 index.(obname).(propsName).name=PropertyInfo(PropIndex).Name;
1178 index.(obname).(propsName).datatype=PropertyInfo(PropIndex).DataType;
1179 index.(obname).(propsName).cnt=1;
1180 index.(obname).(propsName).value=NaN(nsamps,1); %Pre-allocation
1181 index.(obname).(propsName).samples=zeros(nsamps,1); %Pre-allocation
1182 index.(obname).(propsName).value(1)=propsValue;
1183 index.(obname).(propsName).samples(1)=nsamps;
1184 end
1185 end
1186
1187 end %'end' for the 'Property' loop
1188 index.(obname).PropertyInfo=PropertyInfo;
1189
1190 end
1191
1192 end %'end' for the 'Objects' loop
1193 end
1194
1195 %Move the offset calculation to the end to account for added channels and other optimizations
1196 if (kTocRawData) %only do the check if there was raw data in the segment
1197 offset=0;
1198 for kk=1:numel(objOrderList)
1199 obname=objOrderList{kk};
1200 ccnt=index.(obname).rawdatacount;
1201 if (ccnt>0)
1202 index.(obname).rawdataoffset(ccnt)=offset;
1203 if index.(obname).dataType==32
1204 %Datatype is a string
1205 offset=offset+index.(obname).byteSize(ccnt);
1206 else
1207 offset=offset+index.(obname).nValues(ccnt)*index.(obname).datasize;
1208 end
1209 end
1210 end
1211
1212 %Don't know why but sometimes the 'nValues' parameter is sometimes incorrect. Either the documentation was wrong
```

```
or
%someone who wrote the drivers was lazy. Seems to happen with waveform files. Check to make sure that the final
%offset value matches the segment's size. If it doesn't, then check if the size is a multiple of the offset.
If
%it is, then multiply all appropriate parameters in the index structure. If not, then generate a warning.
if (offset~=SegInfo.DataLength(segCnt))
%if (mod(SegInfo.DataLength(segCnt),offset)==0)
multiplier=floor(SegInfo.DataLength(segCnt)/offset);
for kk=1:numel(objOrderList)
obname=objOrderList{kk};
ccnt=index.(obname).rawdatacount;
if (ccnt>0)&&(index.(obname).index(ccnt)==segCnt);
index.(obname).multiplier(ccnt)=multiplier;
if index.(obname).dataType==32
%Datatype is a string
index.(obname).skip(ccnt)=offset-index.(obname).byteSize(ccnt);
else
index.(obname).skip(ccnt)=offset-index.(obname).nValues(ccnt)*index.(obname).datasize;
end
end
end
% else
% warning('segment %d error: offset=%d, dataLength=%d\n',segCnt,offset,SegInfo.DataLength(segCnt));
% end
end
end

end
%clean up the index if it has to much data
fnm=fieldnames(index);
for kk=1:numel(fnm)
ccnt=index.(fnm{kk}).rawdatacount+1;

index.(fnm{kk}).datastartindex(ccnt:end)=[];
index.(fnm{kk}).arrayDim(ccnt:end)=[];
index.(fnm{kk}).nValues(ccnt:end)=[];
index.(fnm{kk}).byteSize(ccnt:end)=[];
index.(fnm{kk}).index(ccnt:end)=[];
index.(fnm{kk}).rawdataoffset(ccnt:end)=[];
index.(fnm{kk}).multiplier(ccnt:end)=[];
index.(fnm{kk}).skip(ccnt:end)=[];

end
end

function ob=getData(fid,index)
ob=[];
fnm=fieldnames(index);
for kk=1:length(fnm)
id=index.(fnm{kk});
nsamples=sum(id.nValues.*id.multiplier);
if id.rawdatacount>0
cname=id.name;
ob.(cname).nsamples=0;
if id.dataType==32
ob.(cname).data=cell(nsamples,1);
else
ob.(cname).data=zeros(nsamples,1);
```

```
1269 end
1270 for rr=1:id.rawdatacount
1271 %Loop through each of the groups/channels and read the raw data
1272 fseek(fid,id.datastartindex(rr)+id.rawdataoffset(rr),'bof');
1273
1274
1275 nvals=id.nValues(rr);
1276
1277 if nvals>0
1278
1279 switch id.dataType
1280
1281 case 32 %String
1282 %From the National Instruments web page (http://zone.ni.com/devzone/cda/tut/p/id/5696) under the
1283 %'Raw Data' description on page 4:
1284 %String type channels are preprocessed for fast random access. All strings are concatenated to a
1285 %contiguous piece of memory. The offset of the first character of each string in this contiguous
piece
1286 %of memory is stored to an array of unsigned 32-bit integers. This array of offset values is
stored
1287 %first, followed by the concatenated string values. This layout allows client applications to
access
1288 %any string value from anywhere in the file by repositioning the file pointer a maximum of three
times
1289 %and without reading any data that is not needed by the client.
1290 data=cell(1,nvals*id.multiplier(rr)); %Pre-allocation
1291 for mm=1:id.multiplier(rr)
1292 StrOffsetArray=fread(fid,nvals,'uint32');
1293 for dcnt=1:nvals
1294 if dcnt==1
1295 StrLength=StrOffsetArray(dcnt);
1296 else
1297 StrLength=StrOffsetArray(dcnt)-StrOffsetArray(dcnt-1);
1298 end
1299 data{1,dcnt+(mm-1)*nvals}=char(convertToText(fread(fid,StrLength,'uint8=>char'))');
1300 end
1301 if (id.multiplier(rr)>1)&&(id.skip(rr)>0)
1302 fseek(fid,id.skip(rr),'cof');
1303 end
1304 end
1305 cnt=nvals*id.multiplier(rr);
1306
1307 case 68 %Timestamp
1308 %data=NaN(1,nvals); %Pre-allocation
1309 data=NaN(1,nvals*id.multiplier(rr));
1310 for mm=1:id.multiplier(rr)
1311 dn=fread(fid,2*nvals,'uint64');
1312 tsec=dn(1:2:end)/2^64+dn(2:2:end);
1313 data((mm-1)*nvals+1:(mm)*nvals)=tsec/86400+695422-4/24;
1314 fseek(fid,id.skip(rr),'cof');
1315 end
1316 %{
1317 for dcnt=1:nvals
1318 tsec=fread(fid,1,'uint64')/2^64+fread(fid,1,'uint64'); %time
since Jan-1-1904 in seconds
1319 %R. Seltzer: Not sure why '5/24' (5 hours) is subtracted from
the time value. That's how it was
1320 %coded in the original function I downloaded from MATLAB
```

```
Central. But I found it to be 1 hour too
1321 %much. So, I changed it to '4/24'.
1322 data(1,dcnt)=tsec/86400+695422-5/24; %/864000 convert to
days; +695422 days from Jan-0-0000 to Jan-1-1904
1323 data(1,dcnt)=tsec/86400+695422-4/24; %/864000 convert to
days; +695422 days from Jan-0-0000 to Jan-1-1904
1324 end
1325 %}
1326 cnt=nvals*id.multiplier(rr);
1327
1328 otherwise %Numeric
1329 matType=LV2MatlabDataType(id.dataType);
1330 if strcmp(matType,'Undefined')
1331 e=errordlg(sprintf('No MATLAB data type defined for a "Raw Data Type" value of
"%.0f".',...
1332 id.dataType),'Undefined Raw Data Type');
1333 uiwait(e)
1334 fclose(fid);
1335 return
1336 end
1337 if (id.skip(rr)>0)
1338 ntype=sprintf('%d*%s',nvals,matType);
1339 if strcmp(matType,'uint8=>char')
1340 [data,cnt]=fread(fid,nvals*id.multiplier(rr),ntype,id.skip(rr));
1341 data=convertToText(data);
1342 else
1343 [data,cnt]=fread(fid,nvals*id.multiplier(rr),ntype,id.skip(rr));
1344 end
1345 else
1346 [data,cnt]=fread(fid,nvals*id.multiplier(rr),matType);
1347 end
1348 end
1349
1350 if isfield(ob.(cname),'nsamples')
1351 ssamples=ob.(cname).nsamples;
1352 else
1353 ssamples=0;
1354 end
1355 if (cnt>0)
1356 ob.(cname).data(ssamples+1:ssamples+cnt,1)=data;
1357 ob.(cname).nsamples=ssamples+cnt;
1358 end
1359 end
1360 end
1361
1362 end
1363
1364 end
1365 end
1366
1367 function [DataStructure,GroupNames]=postProcess(ob,index)
1368 %Re-organize the 'ob' structure into a more user friendly format for output.
1369
1370
1371 DataStructure.Root=[];
1372 DataStructure.MeasuredData.Name=[];
1373 DataStructure.MeasuredData.Data=[];
1374
```

```
1375 obFieldNames=fieldnames(index);
1376
1377 cntData=1;
1378
1379 for i=1:numel(obFieldNames)
1380
1381 cname=obFieldNames{i};
1382
1383 if strcmp(index.(cname).long_name,'Root')
1384
1385 DataStructure.Root.Name=index.(cname).long_name;
1386
1387 %Assign all the 'Property' values
1388 if isfield(index.(cname),'PropertyInfo')
1389 for p=1:numel(index.(cname).PropertyInfo)
1390 cfield=index.(cname).PropertyInfo(p).FieldName;
1391 if isfield(index.(cname).(cfield),'datatype')
1392 DataType=index.(cname).(cfield).datatype;
1393 else
1394 %ASSUME a 'string' data type
1395 DataType='String';
1396 end
1397 DataStructure.Root.Property(p).Name=index.(cname).PropertyInfo(p).Name;
1398
1399 switch DataType
1400 case 'String'
1401 if iscell(index.(cname).(cfield).value)
1402 Value=index.(cname).(cfield).value';
1403 else
1404 Value=cellstr(index.(cname).(cfield).value);
1405 end
1406
1407 case 'Time'
1408 clear Value
1409 if index.(cname).(cfield).cnt==1
1410 if iscell(index.(cname).(cfield).value)
1411 Value=datestr(cell2mat(index.(cname).(cfield).value),'dd-mmm-yyyy HH:MM:SS');
1412 else
1413 Value=datestr(index.(cname).(cfield).value,'dd-mmm-yyyy HH:MM:SS');
1414 end
1415 else
1416 Value=cell(index.(cname).(cfield).cnt,1);
1417 for c=1:index.(cname).(cfield).cnt
1418 if iscell(index.(cname).(cfield).value)
1419 Value(c)={datestr(cell2mat(index.(cname).(cfield).value),'dd-mmm-yyyy HH:MM:SS')};
1420 else
1421 Value(c)={datestr(index.(cname).(cfield).value,'dd-mmm-yyyy HH:MM:SS')};
1422 end
1423 end
1424 end
1425
1426 case 'Numeric'
1427 if isfield(index.(cname).(cfield),'cnt')
1428 Value=NaN(index.(cname).(cfield).cnt,1);
1429 else
1430 if iscell(index.(cname).(cfield).value)
1431 Value=NaN(numel(cell2mat(index.(cname).(cfield).value)),1);
1432 else
```

```
1433 Value=NaN(numel(index.(cname).(cfield).value),1);
1434 end
1435 end
1436 for c=1:numel(Value)
1437 if iscell(index.(cname).(cfield).value)
1438 Value(c)=index.(cname).(cfield).value{c};
1439 else
1440 Value(c)=index.(cname).(cfield).value(c);
1441 end
1442 end
1443 otherwise
1444 e=errordlg(sprintf(['No format defined for Data Type "%s" in the private function
"postProcess" '...
1445 'within %s.m.'],index.(cname).(cfield).datatype,mfilename),'Undefined Property Data Type');
1446 uiwait(e)
1447 return
1448 end
1449 if isempty(Value)
1450 DataStructure.Root.Property(p).Value=[];
1451 else
1452 DataStructure.Root.Property(p).Value=Value;
1453 end
1454 end
1455 end
1456
1457
1458 end
1459
1460 DataStructure.MeasuredData(cntData).Name=index.(cname).long_name;
1461 %Should only need the 'ShortName' for debugging the function
1462 %DataStructure.MeasuredData(cntData).ShortName=cname;
1463 if (isfield(ob,cname))
1464 if isfield(ob.(cname),'data')
1465 DataStructure.MeasuredData(cntData).Data=ob.(cname).data;
1466 %The following field is redundant because the information can be obtained from the size of the 'Data' field.
1467 DataStructure.MeasuredData(cntData).Total_Samples=ob.(cname).nsamples;
1468 else
1469 DataStructure.MeasuredData(cntData).Data=[];
1470 DataStructure.MeasuredData(cntData).Total_Samples=0;
1471 end
1472 else
1473 DataStructure.MeasuredData(cntData).Data=[];
1474 DataStructure.MeasuredData(cntData).Total_Samples=0;
1475 end
1476
1477 %Assign all the 'Property' values
1478 if isfield(index.(cname),'PropertyInfo')
1479 for p=1:numel(index.(cname).PropertyInfo)
1480 cfield=index.(cname).PropertyInfo(p).FieldName;
1481 DataStructure.MeasuredData(cntData).Property(p).Name=index.(cname).(cfield).name;
1482
1483 if strcmpi(DataStructure.MeasuredData(cntData).Property(p).Name,'Root')
1484 Value=index.(cname).(cfield).value;
1485 else
1486
1487 switch index.(cname).(cfield).datatype
1488 case 'String'
1489 clear Value
```

```
1490 if index.(cname).(cfield).cnt==1
1491 if iscell(index.(cname).(cfield).value)
1492 Value=char(index.(cname).(cfield).value);
1493 else
1494 Value=index.(cname).(cfield).value;
1495 end
1496 else
1497 Value=cell(index.(cname).(cfield).cnt,1);
1498 for c=1:index.(cname).(cfield).cnt
1499 if iscell(index.(cname).(cfield).value)
1500 Value(c)=index.(cname).(cfield).value;
1501 else
1502 Value(c)={index.(cname).(cfield).value};
1503 end
1504 end
1505 end
1506
1507 case 'Time'
1508 clear Value
1509 if index.(cname).(cfield).cnt==1
1510 if iscell(index.(cname).(cfield).value)
1511 Value=datestr(cell2mat(index.(cname).(cfield).value),'dd-mmm-yyyy HH:MM:SS');
1512 else
1513 Value=datestr(index.(cname).(cfield).value,'dd-mmm-yyyy HH:MM:SS');
1514 end
1515 else
1516 Value=cell(index.(cname).(cfield).cnt,1);
1517 for c=1:index.(cname).(cfield).cnt
1518 if iscell(index.(cname).(cfield).value)
1519 Value(c)={datestr(cell2mat(index.(cname).(cfield).value),'dd-mmm-yyyy HH:MM:SS')};
1520 else
1521 Value(c)={datestr(index.(cname).(cfield).value,'dd-mmm-yyyy HH:MM:SS')};
1522 end
1523 end
1524 end
1525
1526 case 'Numeric'
1527 if isfield(index.(cname).(cfield),'cnt')
1528 Value=NaN(index.(cname).(cfield).cnt,1);
1529 else
1530 if iscell(index.(cname).(cfield).value)
1531 Value=NaN(numel(cell2mat(index.(cname).(cfield).value)),1);
1532 else
1533 Value=NaN(numel(index.(cname).(cfield).value),1);
1534 end
1535 end
1536 for c=1:numel(Value)
1537 if iscell(index.(cname).(cfield).value)
1538 Value(c)=index.(cname).(cfield).value{c};
1539 else
1540 Value(c)=index.(cname).(cfield).value(c);
1541 end
1542 end
1543
1544 otherwise
1545 e=errordlg(sprintf(['No format defined for Data Type "%s" in the private function
"postProcess" '...
1546 'within %s.m.'],index.(cname).(cfield).datatype,mfilename),'Undefined Property Data Type');
```

```
1547 uiwait(e)
1548 return
1549 end
1550 end
1551 if isempty(Value)
1552 DataStructure.MeasuredData(cntData).Property(p).Value=[];
1553 else
1554 DataStructure.MeasuredData(cntData).Property(p).Value=Value;
1555 end
1556 end
1557 else
1558 DataStructure.MeasuredData(cntData).Property=[];
1559 end
1560
1561 cntData = cntData + 1;
1562 end %'end' for the 'groups/channels' loop
1563
1564 %Extract the Group names
1565 GroupIndices=false(numel(DataStructure.MeasuredData),1);
1566 for d=1:numel(DataStructure.MeasuredData)
1567
1568 if ~strcmpi(DataStructure.MeasuredData(d).Name,'Root')
1569 if (DataStructure.MeasuredData(d).Total_Samples==0)
1570 fs=strfind(DataStructure.MeasuredData(d).Name,'/');
1571 if (isempty(fs))
1572 GroupIndices(d)=true;
1573 end
1574 end
1575 end
1576
1577 end
1578 if any(GroupIndices)
1579 GroupNames=sort({DataStructure.MeasuredData(GroupIndices).Name})';
1580 else
1581 GroupNames=[];
1582 end
1583
1584 end
1585
1586 function sz=getDataSize(LVType)
1587 switch(LVType)
1588 case 0
1589 sz=0;
1590 case {1,5,33}
1591 sz=1;
1592 case 68
1593 sz=16;
1594 case {8,10}
1595 sz=8;
1596 case {3,7,9}
1597 sz=4;
1598 case {2,6}
1599 sz=2;
1600 case 32
1601 e=errordlg('Do not call the getDataSize function for strings. Their size is written in the data file','Error');
1602 uiwait(e)
1603 sz=NaN;
1604 case 11
```

```
1605 sz=10;
1606 end
1607 end
1608
1609 function matType=LV2MatlabDataType(LVType)
1610 %Cross Refernce Labview TDMS Data type to MATLAB
1611 switch LVType
1612 case 0 %tdsTypeVoid
1613 matType='';
1614 case 1 %tdsTypeI8
1615 matType='int8';
1616 case 2 %tdsTypeI16
1617 matType='int16';
1618 case 3 %tdsTypeI32
1619 matType='int32';
1620 case 4 %tdsTypeI64
1621 matType='int64';
1622 case 5 %tdsTypeU8
1623 matType='uint8';
1624 case 6 %tdsTypeU16
1625 matType='uint16';
1626 case 7 %tdsTypeU32
1627 matType='uint32';
1628 case 8 %tdsTypeU64
1629 matType='uint64';
1630 case 9 %tdsTypeSingleFloat
1631 matType='single';
1632 case 10 %tdsTypeDoubleFloat
1633 matType='double';
1634 case 11 %tdsTypeExtendedFloat
1635 matType='10*char';
1636 case 25 %tdsTypeSingleFloat with units
1637 matType='Undefined';
1638 case 26 %tdsTypeDoubleFloat with units
1639 matType='Undefined';
1640 case 27 %tdsTypeextendedFloat with units
1641 matType='Undefined';
1642 case 32 %tdsTypeString
1643 matType='uint8=>char';
1644 case 33 %tdsTypeBoolean
1645 matType='bit1';
1646 case 68 %tdsTypeTimeStamp
1647 matType='2*int64';
1648 otherwise
1649 matType='Undefined';
1650 end
1651
1652 end
1653
1654 function text=convertToText(bytes)
1655 %Convert numeric bytes to the character encoding localy set in MATLAB (TDMS uses UTF-8)
1656
1657 text=native2unicode(bytes,'UTF-8');
1658 end
1659
```

Appendix II: LabVIEW Modifications (Timing Synchronization)

In parallel to researching methods of outputting a time stamp or timing pulse from the MRI or MRI control computer, the following changes were made to the LabVIEW VI to display both the timestamp of the PPU signal generation and the time at which the HyPSTR encoder registered “0 mm”. Theoretically, these values should be identical, but there are limits to the precision of timing circuitry (especially when generating pulses through a serial RS-232 to fiber-optic cable converter.) A detailed description of timing issues and magnitudes is in Chapter 4: Assess MRI Timing Accuracy. The additional display on the LabVIEW panel allowed us to detect a large (problematic) vs. small (usually acceptable) synchronization offset between the HyPSTR and MRI acquisition before beginning the gated MRI procedure. Previously, this offset was only discoverable after the lengthy, and expensive, MRI test was completed. In the worst cases, the MRI data were completely unusable.

If the difference in timestamps on the LabVIEW panel was large (hundreds of ms), the HyPSTR and LabVIEW were stopped and restarted until it was small (<20 ms). A large offset, perhaps caused by limitations of the size of the optical patches within the encoder (used to track the actuator movement), was more rare than a small offset. The HyPSTR setup could be stopped and started in less than 2 min. Therefore, this workaround cost on the order of 5 to 10 min of time for each test (if multiple restarts were required), but sometimes as little as 0 min (if no reset was required.)

PPU timestamp output

The PPU emit time is datalogged inside of PPU_signal_generator_loop_2_sub.vi.[1] This VI is located at the top of the block diagram (Figure II.1), above all while loops and is triggered by an occurrence event that tells when the encoder has begun to move. Within this sub-VI, the timestamp is requested at “Note 9” and then passed inside of a while loop and a case structure. When the case is "Send Signal" another smaller case structure runs if the PPU voltage is high (during the 2 msec pulse, 5 volts). The PPU timestamp is recorded there (Figure II.2). To display this PPU on the main VI's front panel, pass a value by reference between VIs.[2] The indicator on the main VI still needs to be a timestamp type.

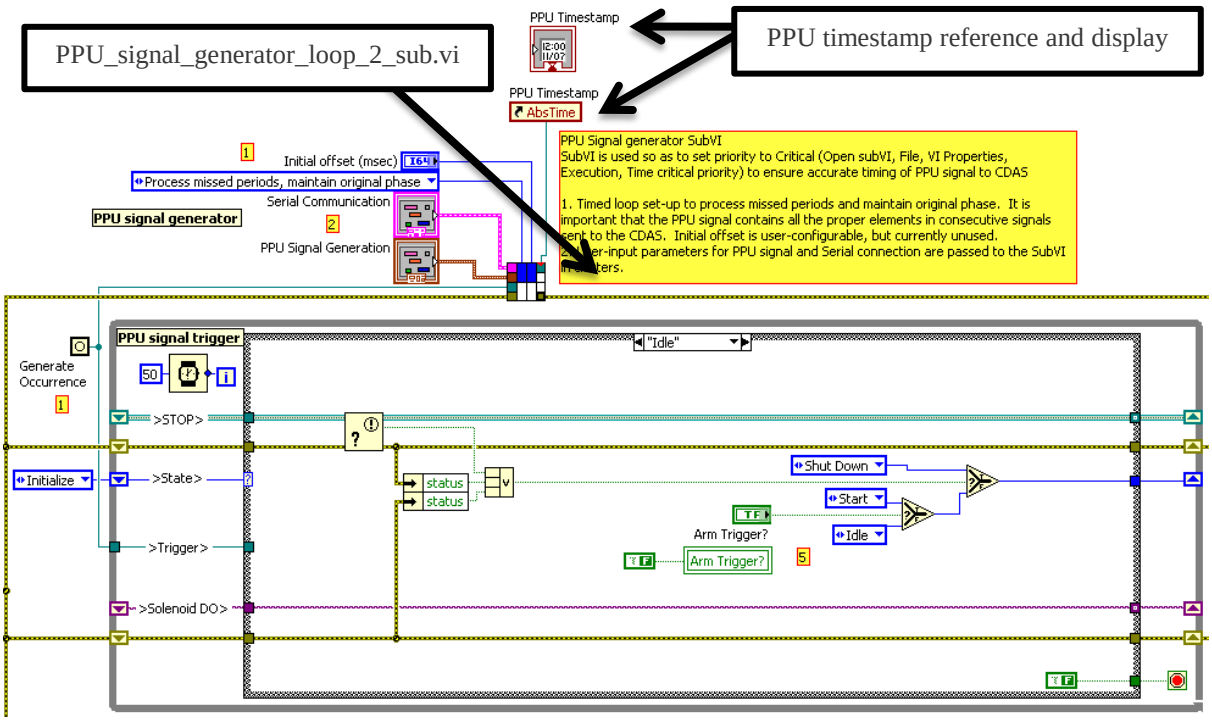


Figure II.1. Partial screenshot of the main VI, showing the location of the Signal Generation Loop VI and referenced timestamp variable for display on the front panel.

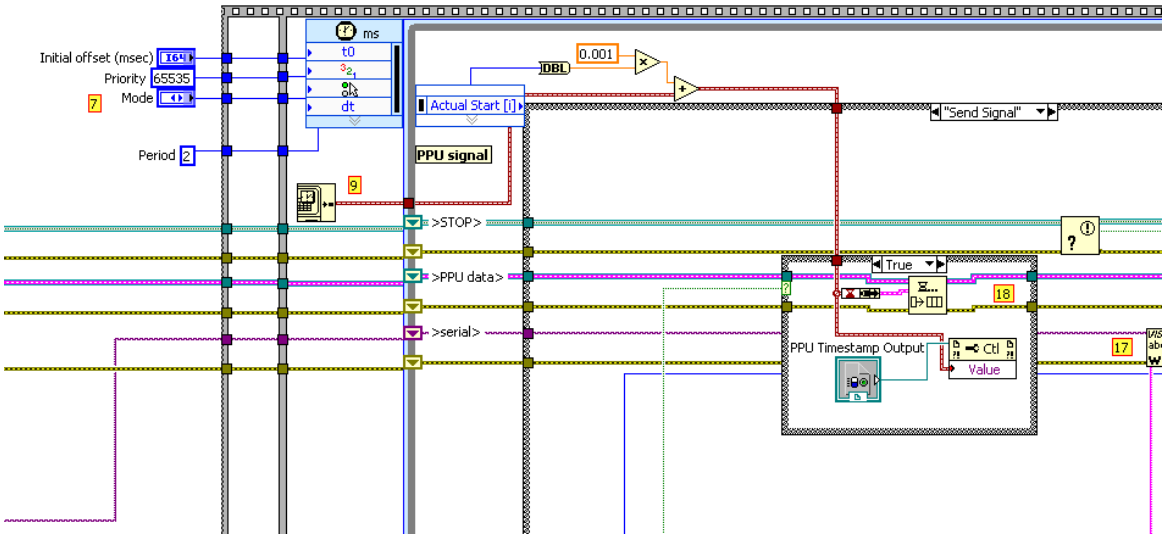


Figure II.2. Timestamp is generated at the note marker, yellow boxed "9". Follow the red line from the timestamp generation to the case structure that contains note marker "18". The "PPU Timestamp Output" control is the value that is referenced later.

Encoder timestamp output

I tried to display the most recent value in the encoder queue with the "Preview Queue Element" control, which should work well with a Waveform Datatype (Figure III.3). This function was already in place in the "Update Front Panel" while loop.[1] I wrote a condition that checks whether the encoder value is "0.000000," the encoder output at the bottom of its sine wave travel. The next step was to take a new timestamp at this point and output the position and time. This does not work because this timestamp refers to the time the queue was accessed for display, not when it was recorded. I tested it and it does not give accurate encoder zero times, even when placing the timestamp generation inside while loops, as close as possible to the DAQ board signal acquisition.

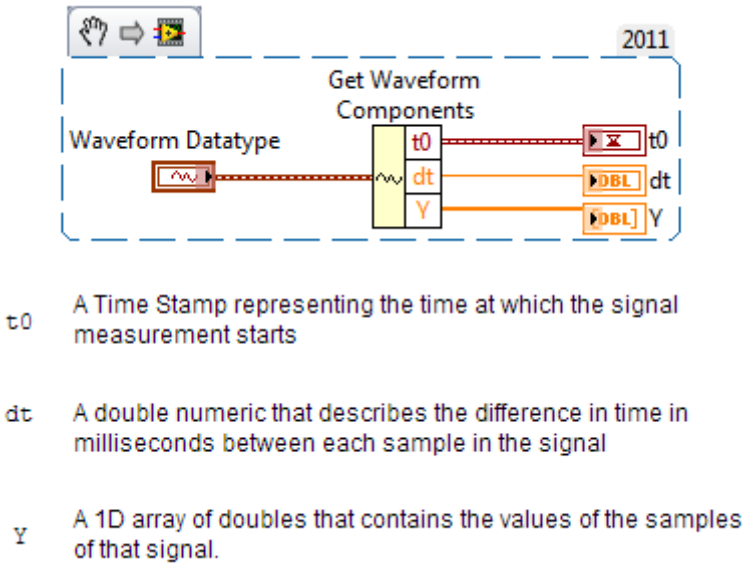


Figure II.3. Structure of LabVIEW's Waveform Datatype.[3]

It appears that the only way to get the time associated with a data value is to know the waveform index of the value, multiply this by the time step between acquisitions (dt) and add it to the timestamp of the first value (t0). I could not find a way to obtain the index position of a specific value while the VI loops were running.

The eventual solution was to use a plot display to show time-related information. LabVIEW's internal coding must keep track of the index, because one of the waveform graphical output options gives the option of showing the time stamp on the x-axis. This worked well, but the screen refreshes every 100 ms with a new encoder position, making it impossible to read the zero value accurately at the bottom of the sine wave. I moved the (already existing) encoder graphical output into a "less than 0.05 mm" loop and broadened the range on the x-axis, for better temporal resolution. This combination, with 100 ms data display updates (clock control = 100 ms for this while loop), worked well. A 200 ms interval and 0.1 mm test condition worked

also, but didn't show the low point of the sine wave as consistently. Every 100 ms, the VI reads the last 250 encoder data points. At 2500 Hz, this should be 100 ms of data.

It's obvious when things are late vs early, but I tested myself with two additional runs to see how close I could get within the first 5 cycles (25 seconds). From a graph, I was visually able to estimate the following lags:

1. Estimated 500 ms early...was actually 506 ms early
2. Estimated 28 ms late....was actually 26 ms late

This optical accuracy was considered sufficient in order to determine, at the time of data collection, whether the system was exhibiting sufficient lag to warrant further adjustment prior to MRI collection.

Other Useful Notes

Reading from the Encoder DAQMax (data acquisition board) signal is a slower process than reading from the top of the waveform (parallel timing) with Preview Queue Element. The Encoder signal comes into LabVIEW as an array of 250 doubles. The 250 doubles probably refers to the maximum data chunk size that the board can read from the raw encoder signal at the specified sampling rate, 2500 Hz.

References:

1. Stebbins MJ. Obtaining Material Properties of the Plantar Soft Tissue for a Patient-Specific Finite Element Model of the Foot. *University of Washington*. 2012; Mechanical Engineering Department.
2. Updating a Control or Indicator of a Top Level VI from a SubVI. National Instruments, 2013.

3. Anonymous. How To Measure Voltage. National Instruments, 2014, p. 6.

Appendix III: HyPSTR Preparation and Test Day Procedure

Operating the HyPSTR can be a challenging task, especially in the tight space of an MRI control room. Here is a set of instructions to help guide someone through the process of connecting the components of the instrumentation and follow the correct order of events on a test day. The test day actually begins a few days in advance, with bubble removal and displacement performance checks to ensure there are no major problems prior to moving equipment out of the lab space (and to the UW Bio-Molecular Imaging Center, in this case).

1. Reserve a van – This can be done either through the VA or the UW. It should ideally be a 15-passenger van with the rear seats removed. It is possible to fit the HyPSTR into the rear of a minivan, but many parts need to be removed in order to do so. Some 15-passenger vans are too tall for standard parking garages.

A. VA van reservation. Email James Felton (James.Felton@va.gov) and Peter Fiamalua (Peter.Fiamalua@va.gov) approximately one week prior to the test date. Ask for a "motor pool van", and any particular size requirements. Keys must be picked up between 6 AM and 10 AM on the reservation day and can be done in Building 105 (as of 11/2014).

B. UW van reservation. Follow instructions from the Fleet Services department regarding "Motorpool Reservations". http://www.washington.edu/facilities/transportation/fleetservices/ (as of 5/2015)

2. Remove dissolved gases from the hydraulic system – Removing dissolved gas from the hydraulic lines is the first of several bubble control measures. Larger, entrained and entrapped, bubbles will get re-introduced to the system when fittings are disconnected for transportation.

The most important step is to reduce the dissolved gas to 50% or lower so that bubbles do not come out of solution during agitation or the temporary vacuum created by the hydraulic draw stroke. The worst kind of bubble is the type that forms during a subject test. It adds a large amount of compressibility to the high pressure water column.

A. Connect the HyPSTR plumbing and electrical interfaces (see **Step 5**).

B. Turn the HyPSTR on (see **Step 6**). This is necessary in the bubble removal stage to energize the safety-release solenoid so that water does not simply pump out into the drain bucket.

C. Fill the silver pot, with a bulkhead fitting, from a clean water source. Lab tap water is acceptable. Place the pot above the HyPSTR, or have an assistant hold it, so that the feed line can overcome the 1 psi check valve restriction at point I (Figure III.1a and III.1b).

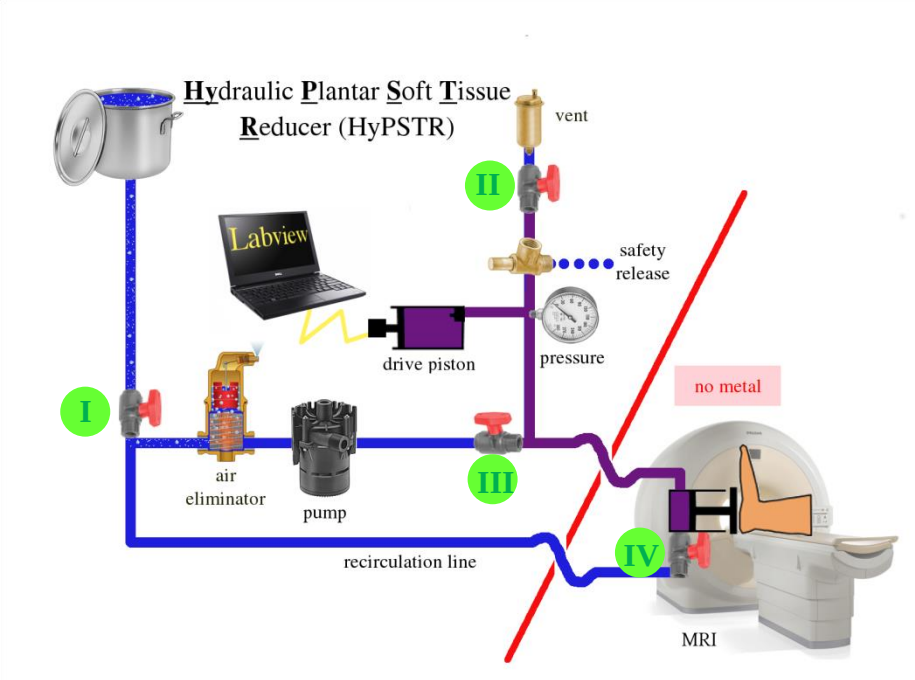


Figure III.1a – HyPSTR schematic. Also see the photograph for better spatial reference (Figure III.1b).

Figure III.1b – Photo of the main HyPSTR cart in the lab. Valves (marked with green circles) I, II, and III should be open when the recirculation pump is on.

D. Open plumbing valves (I, II, III, IV) (Figures III.1a-b and III.2) and place a red Velcro strap around the slave platen to hold it in its most retracted position (Figure III.2). The check valve at point I will be open automatically as long as the bucket is held high enough above the HyPSTR (7 feet from the ground is enough). It may be necessary to push the slave platen back manually. Otherwise, the recirculation pump pressure will extend the piston in the next step and bubbles could be trapped in the housing.

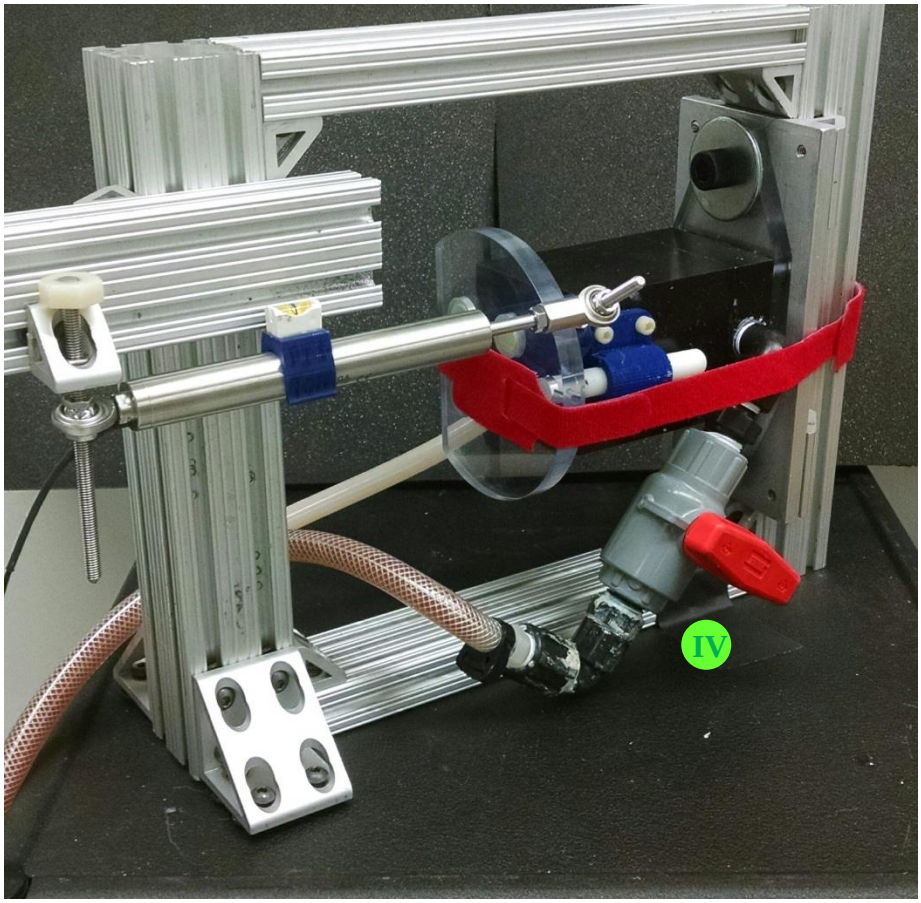


Figure III.2 – Testing frame built from 8020 aluminum bars. The red straps hold the platen from extending while the recirculation pump is running. The silver, horizontal cylinder is the LVDT, used for checking displacement performance. Valve IV is marked with a green circle.

E. Partially close (50% or 45^{o}) valve III (Figure III.1a-b) and turn on the recirculation pump. Adjust valve III until the analog pressure gauge reads 2 psi (Figure III.3). It is important to

start with a slow flow rate so that large entrained bubbles have time to rise to the vent or top of the air eliminator instead of rapidly passing these points and breaking down into smaller bubbles. Small bubbles are harder to remove and undesirably promote the formation of dissolved gases.

Figure III.3 – Pressure gauge for quickly monitoring the flow rate of the recirculation pump. The dissolved oxygen sample port is also visible in this picture, with the syringe body attached to the fitting. A small valve (shown in the off position) leads to this port.

F. After 10 minutes of slow flow rate, valve III can be opened slightly, but make sure to keep the pressure below 5 psi. Continue pumping until the dissolved gas percentage is 50% or lower. Depending on the state of the system prior to starting, this might take 20 minutes or 2 hours.

G. To check the dissolved gas, use the YSI ProODO dissolved oxygen (DO) meter. Pull samples from the syringe fitting next to the vent at the system high point (Figure III.3). Because the (semi) vertical pipe column leading to the vent is not directly in-line with the recirculation loop, changes in the dissolved gas are slower than in other parts of the system. It may be a good idea to recirculate the water for an hour, turn off the pump, and then check the DO percentage the next day. The presence of oxygen is easier to measure than other dissolved gases, but all gases are being removed. Thus, a 50% DO reading means the amount of dissolved "air" is also approximately 50% of normal equilibrium levels.

H. Turn off recirculation pump.

I. Close valves (I, II, III, IV) and remove the red Velcro restraining strap. The system is now ready for hydraulic testing.

3. Perform displacement check

A. Open Q-Programmer and LabVIEW (if not already open) from the shortcuts on the HyPSTR laptop desktop. These shortcuts should always be used, because they are set to run the programs in a priority mode that ensures optimal processing power (above the level of background operating system tasks). Once LabVIEW is open, choose the "edw_LELP_01.vi" file. This VI is set up to display and record LVDT, encoder, load cell, and pressure data.

B. Attach the LVDT to an 8020 Aluminum frame. There are many acceptable ways to do this (Figure III.2).

C. Open the Q-Programmer *program* titled "Sine_0.2hz_10.0mm.qpr" and run the code. Make sure you did not try to open a *segment*. If Q-Programmer is unable to communicate with the actuator, make sure the correct serial port is selected. Currently, and usually, that port number is "8".

D. In LabVIEW, click the  button to begin monitoring. Return to Q-Programmer and click the "Execute Program" button. Watch the minimum and maximum LVDT values (in LabVIEW) and confirm that their difference is above 9.55 mm. The ideal displacement range is 9.65 mm, because of a slightly smaller piston diameter (96.5% of the slave piston diameter) at the master piston side. If the displacement is significantly less than 9.55 mm, the HyPSTR might not be working efficiently or consistently enough to apply the necessary loads to the subject's foot later.

E. Click the 

button when done.

4. Transport the HyPSTR

A. Disconnect the recirculation and high pressure tubing water lines from the quick disconnect fittings at the main cart. This should completely separate the two halves of the HyPSTR. Disconnect all laptop-to-HyPSTR electrical/data interfaces.

B. Organize all of the tubing and, depending on the transportation vehicle, remove the vent from the top of the cart (plug the hole with a threaded end cap). The upper project box, containing primary signal processing electronics, might also need to be removed.

C. Take all tools used for HyPSTR deconstruction to the MRI facility for reconstruction.

D. Transport all components, with help from one or two assistants and reconstruct the HyPSTR at the MRI facility.

5. Connect the HyPSTR plumbing and electronics

A. Join the slave and master halves of the system at the quick disconnect fittings on the main cart. If at the MRI facility, route the hydraulic lines through the wall access port first.

B. Connect the laptop data connections as shown in the picture (Figure III.4).

Figure III.4. – Laptop data and electrical connections.

C. At the MRI facility, plug the RS232 serial-to-fiber optic signal converter into the laptop and route the fiber optic cable to the control and data acquisition system (CDAS) signal input box (Figure III.5).

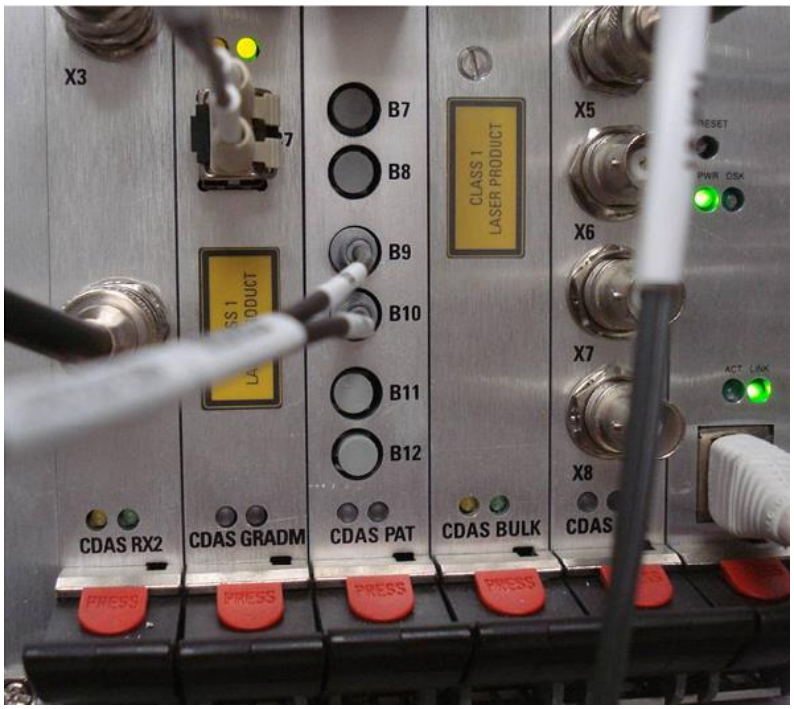


Figure III.5 – CDAS box for Philips 3T MRI systems. The fiber optic cables that carry the timing pulse (from the RS-232 serial converter) plug into the holes marked B9 and B10.

D. Open the bottom project box, containing power supplies and a separate fiber optic interface adapter (Figure III.6). Connect the E-Stop safety button fiber optic cables. If this is not connected correctly, the pressure release solenoid will not close and therefore the system cannot be used.

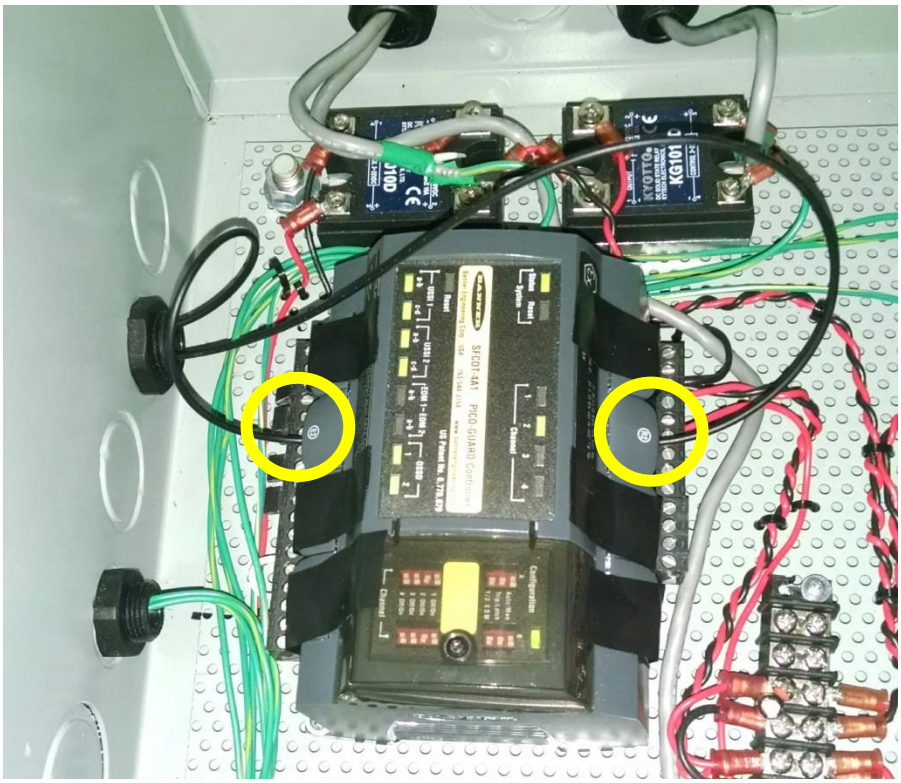


Figure III.6 – Fiber optic control box that manages the emergency stop safety button interface. One black cable is plugged into each of two corresponding sides to create a continuous light path. The connection points are circled in yellow. This set of cables is separate from the timing PPU that plugs into the CDAS.

6. Turn on the HyPSTR

A. Plug the HyPSTR power cable to a wall outlet. Turn on the main power switch to the HyPSTR, followed by the two switches (in order) to the right of the main power switch. The third switch to the right is only used after an emergency stop (physical button, not LabVIEW's stop button) event has occurred. In that case, after the button has been reset, flip the switch up for 0.25 to 2.0 s, and then back down.

B. Plug in the laptop power cable to the power strip or wall outlet.

C. Open Q-programmer and LabVIEW EP from the desktop shortcuts.

D. Once LabVIEW has started, open the "edw_EP_02.vi" file. This VI is set up to display and record encoder and pressure data during MRI tests.

E. Click the  (run) button. The safety release solenoid should make a solitary "click" noise to show that it is energized. There should also be a red LED that turns on at the fiber optic emergency stop button, on top of the uppermost project box, to signal that the pressure release valve is closed and ready for receiving normal system pressures. If it is not obvious that the pressure release valve is closed, toggle the red "Emergency Pressure Release" button on the LabVIEW front panel by clicking it with the mouse. The red LED light on the E-stop button should turn on and off, in addition to an audible "click" of the solenoid.

7. Remove entrained and entrapped bubbles

A. Remove any bubbles that were introduced to the system from the reinstallation at the quick disconnect connections. These bubbles should be large and easy to remove (5 min of recirculation at most).

B. Follow the instructions in **Steps 2C, 2D, and 2E.** Be sure to use the Velcro strap to restrain the slave platen before turning on the recirculation pump.

8. Secure subject in loading jig

A. Construct the loading jig as shown below (Figure III.7), but on the MRI patient table. Attach the appropriately-sized ankle foot orthotic (AFO) to the white plastic/foam collars that are secured to the two long rails. Make sure that all adapter parts for connecting the LVDT have been removed (they contain metal bolts).

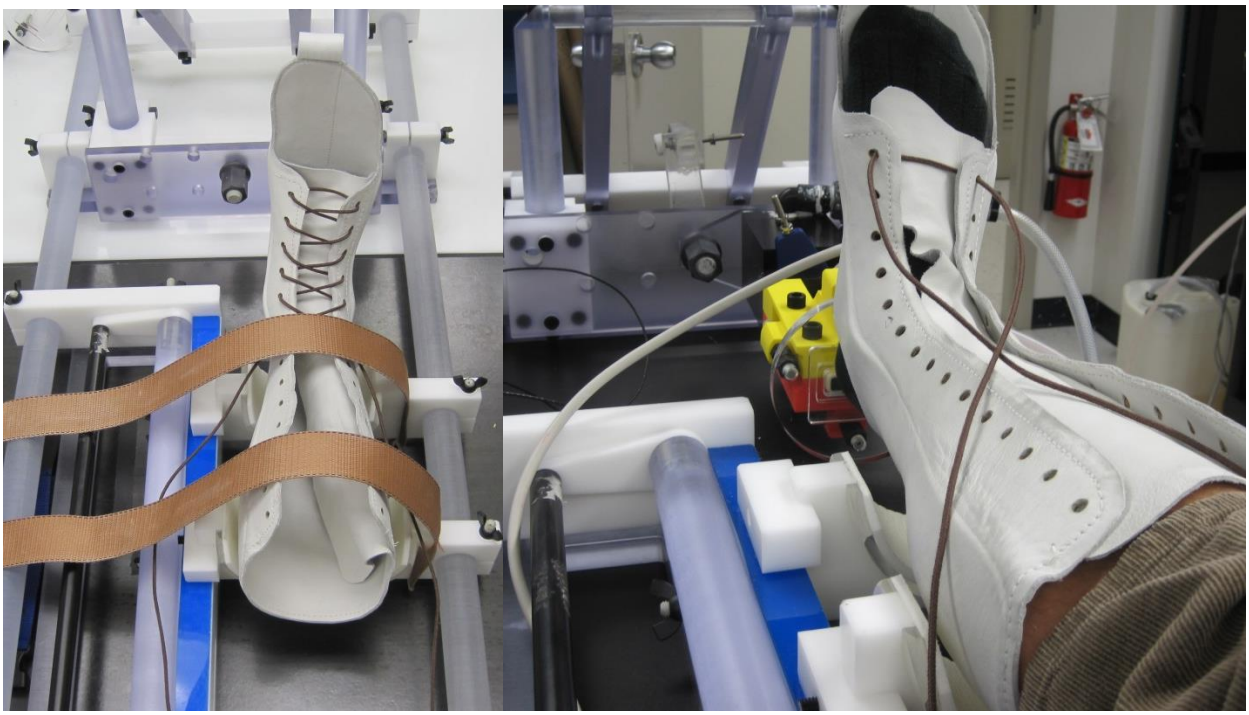


Figure III.7 – Two pictures showing the basic configuration of the loading jig. Notice the yellow and red ultrasound adapter in the right picture. The ultrasound adapter is not needed for the MRI tests.

B. Lace the AFO snuggly (but not painfully) until the lower leg skin cannot slide axially with respect to the loading jig. If there are gaps or locations of discomfort on the leg, use foam shims to make fine fit adjustments.

C. With the HyPSTR loading platen in place, use the wing nuts on the rails to slide the platen or the leg closer to each other, such that the platen is just barely touching the skin.

9. Test displacement range (this will also precondition the foot tissue, which is desirable)

A. Ensure that LabVIEW is displaying the data signals on the various screens and fields.

B. In Q-programmer, open the “Sine_0.2hz_5.0mm.qpr” program and set to cycle 10 times. Click “Execute Program”.

C. Watch the pressure value; it should maximize at close to 200 kPa.

D. Repeat **Step 9B and 9C** with increasing displacements (from 5.0 mm up to 23.0 mm) and stop when a particular test reaches 200 kPa. Until there is a better prediction method for load on a subject’s foot, 200 kPa (approximately 200 N, from the calibration curves) is as high as we are comfortable testing. There was concern that higher loads would be painful after 20+ min of testing. The effects of larger loads on insensate patients are also not known.

E. Once the final displacement has been chosen, either reference previous tests to determine the precise PPU period or repeat 10 cycles with the LabVIEW “record data” toggle switch turned on. The period is not exactly 5000 ms and the quality of the MRI data greatly depends on keeping the HyPSTR as close to synchronized as possible with the gated MRI protocol. The PPU period can be calculated by processing the LabVIEW “PPU” datalog file with the LabVIEW VI, “datalog2ascii.vi”. This can be done in the MRI control room, but it is better to have these values pre-determined from lab verification tests.

10. Adjust PPU signal parameter and confirm gated MRI protocol

A. If not already open, select the “edw_EP_02.vi” LabVIEW VI. At the bottom of the screen, choose a unique file name for saving the test data. Make sure the “Record Data?” switch is toggled to the right (green color).

B. Scroll the VI panel to the left and find the “PPU Signal Generation” field (Figure III.8). Enter the precise number of milliseconds for the PPU (and therefore also encoder) period that was determined in **Step 9**.

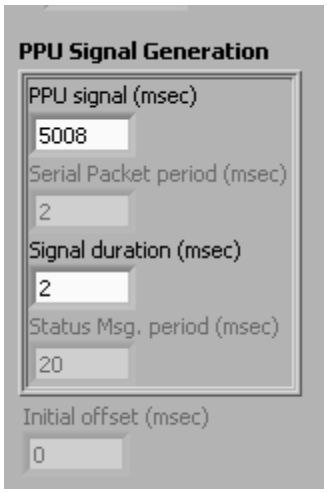


Figure III.8 – Visual reference for where to enter the PPU period. Here, it is shown as 5008 ms.

C. Confirm with the radiologist that the gated MRI protocol is ready, but don’t start the MRI yet.

11. Start the test (may require restarting several times if timing is not synchronized)

A. Click “Encoder reset”

B. Click “Time reset”

C. Click “Arm Trigger?”

D. Click  (run).

E. Switch to the Q-Programmer window and click “Execute Program”. Make sure you have the correct program loaded and enough cycles to last longer than the time of the MRI scan.

F. Switch back to LabVIEW and monitor the first 10 cycles on the encoder display (Figure III.9).

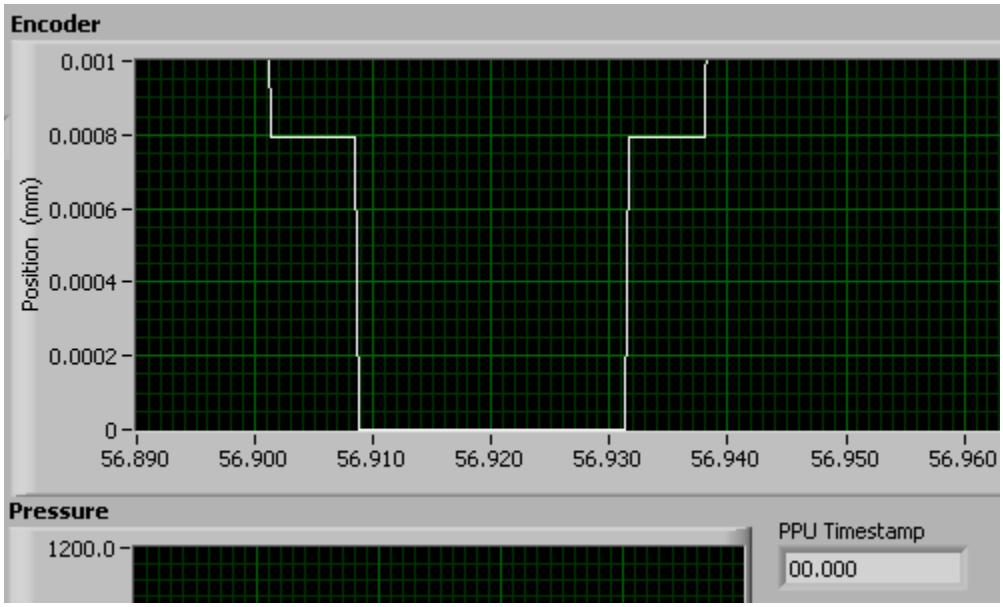


Figure III.9 – Encoder displacement (very magnified zoom) showing the time of the "zero" position. Here, it is approximately 56.920 ms. During an actual MRI test, the "PPU Timestamp" field will be non-zero, and should display a value very close to 56.920 ms.

G. If the encoder "zero" time on the plot is within 20 ms of the PPU Timestamp, then wait until 10 cycles are complete and then tell the radiologist to start the gated MRI acquisition. The MRI control computer should show a visual confirmation that it is receiving the PPU.

H. If the encoder "zero" time on the plot is greater than 20 ms (usually 100+ ms if this is the case), then click the Stop button, choose a new filename, and start again at **Step 11A**. Repeat this until the encoder zero, as shown graphically, is within 20 ms of the PPU timestamp.

12. Stop the HyPSTR

A. Allow the gated MRI scanning to finish completely.

B. Let the HyPSTR cycle approximately 5 more times in order to be sure that it has recorded data through the entire MRI acquisition process.

C. Click the 

button.

13. Remove the subject from the loading jig

A. Unlace the AFO and remove the subject's leg from the loading jig.

B. Disassemble all HyPSTR components necessary for transportation back to the lab.

14. Backup the LabVIEW data (pressure and displacement) and ask for MRI DVD.

A. Back up the LabVIEW data file on a USB drive.

B. When the MRI post-processing and gated image construction is complete, ask the radiologist for a copy of the data on DVD.

C. Also ask for the "exam card" that corresponds to the MRI data. It will contain information like the following:

<parameter name="Scan percentage (%)" value="87.09677"/>
<parameter name="TFE shots" value="270"/>
<parameter name="TFE dur. shot / acq (ms)" value="190.5 / 152.4"/>
<parameter name="Trigger delay max. / act. (ms)" value="4500.0 / 152.0"/>
<parameter name="Phase interval (ms)" value="278.99"/>

15. Return the van and return the HyPSTR back to the lab.

16. Go for a jog and have beer, preferably from the Flying Lion.