Declaration Page

Task Topic: Biomechanics Analysis of Self-Induced Mechanical Shocks

Task Subject: Biomechanics

No of Words: 8000

Task Instructions: The present project will set out to utilise the MATLAB and an open-source OpenSim package to visualise the trajectories of the human body (kinematics) and ground reaction forces and moments (kinetics) from a dynamic sitting experiment. The aim of the project is to quantify the dynamic forces and moments during the sitting motion of the human subject.

Objectives:

-Use Matlab for dynamic data and dynamic movement simulation.

-Prepare the data for using Matlab to integrate with OpenSim simulation.

-Conduct inverse kinematics and dynamics analysis in OpenSim

Dissertation Help Services

ABSTRACT

Dynamical simulations play an important role in tackling tough engineering challenges and can influence the decision-making and treatment approaches. In this post, we are creating a new interface between the two software tools to combine the MATLAB/fast simulator models, controller systems and powerful digital capabilities with OpenSim simultaneously and human motion dynamics. The MATLAB S-function technique is used for OpenSim to integrate with Simulink, and both open loop and lock loop systems display the interface. (D. A. Winter, 2009.)

The initial research focuses on developing static equalization equations and models for the calculation of forces and pressure on the body, implementing these models in the MATLAB software to be easily integrated into current software for biomechanical simulation and developing a user interface that will allow clinicians to report relevant data. To far, the MATLAB code has been developed to represent the human body as rigid body segments, utilising average data from several individual investigations. The code is entered into to indicate a possible wheelchair user's orientation in the wheelchair with the height and weight of the potential user. The code is used. (D. A. Winter, 2009.)

When MATLAB/Simulink is used in the open-loop system as a distinct reproduction for the OpenSim Forward Dynamics Tool, the shut-off mechanism provides a unique functionality to OpenSim which is required for the majority of human movement simulations. In both open-loop and clos-loop instances, an example of arm model was employed effectively. This project is intended to undertake a basic biomechanical analysis of the human body when seated and to apply this analytic especially to disabled people. (D. A. Winter, 2009.)

The study is initiated to produce static equilibrium equations and models for the calculation of body forces and pressures. The implementation of these models in MATLAB software will allow them to easily be incorporated in the current bio-mechanical simulation software. Up to now, MATLAB codes were designed as hard parts of the human body utilising data averaged over numerous investigations. The code is entered into to indicate a possible wheelchair user's orientation in the wheelchair with the height and weight of the potential user. (K. Waugh and J. M. Bach, 2019)

The code gives an imagery of the user and the pointing forces connected with postural supports which operate on him or her. Future work involves removing the assumptions provided in the model and implementing the concept in OpenSim software. The given context offers a strong and versatile technique for effective musculoskeletal movement control simulations utilising OpenSim and MATLAB.. This could make predictive modeling more easily used by researchers as a means to treat clinical problems which restrict human movement. (Tilley, 2018)

Acknowledgement (OPTIONAL)

The College of Engineering Studies *********** has endorsed this study. I would want to express appreciation for the research and assistance that my advisers Dr. ***** ***** and Mr. ******* were able to carry out.

Dissertation Help Services

Table of Content

List of Figures

Dissertation Help Services

1. INTRODUCTION

Many neuromusculoskeletal components interact to allow coordinated movement. Human movement-fascinated scientists have carried out significant investigations to explain these components. As such, the mechanic, geometric link between muscles and bones and joint movements are distinguished by a multitude of data. A neuromuscular excitation patterns and the motion kinematics of thousands of patients, both before and during treatment operations, have been studied by clinicians who treat motion defects in people with brain paralysis, a stroke, arthritis and Parkinson's condition. However, it remains an essential task to synthesize thorough descriptions of neuromusculoskeletal system elements with measures of motion to build an integrated knowledge of normal movement and to establish a scientific foundation for rectifying anomalous movement. (Dan., 2017)

Yet, a key issue remains the synthesis of precise descriptions of the parts of the neuromusculoskeletal system and measures of movement to develop an integrated knowledge of normal motion and to provide scientific foundations for the correction of aberrant motion. This research is focused on the development of static equalization and models for strength calculations and pressure on the body. These models can easily be integrated in the current biomechanical simulation software by means of the MATLAB software and the development of an interface to allow clinicians to report data. The MATLAB code has been built as a rigid body sequence to depict the human body using average data from several individual studies.. This code shows a possible wheelchair user orientation in the wheelchair with the probable user's height and weight. You are using the code. (Dan., 2017)

Two basic limitations exist when using experiments alone to understand motion dynamics. Firstly, in experiments, key factors, such as muscular forces, are not typically quantifiable. Second, in complex dynamical systems only from experimental data, it is difficult to establish cause/effect correlations. As a result, it is not easy to clarify the muscle activities. (Ethan, 2017)

Electromyography (EMG) captured recordings for example may show when a muscle is active, but analysis of EMG not allows one to establish which bodily movements are caused by the activities of a muscle. It is difficult to determine how specific muscles contribute to the observable movements since a muscle may accelerate joints and not connect the body segments. In combination with studies, a theoretical framework is needed for discovering the principles that regulate the coordination of muscles during normal motions, determining the contribution of neuromuscular deficits to aberrant movement and the functional implications of therapy. The theoretical framework must disclose connections of cause-and-effect between neuromuscular arousal patterns, muscle forces and body movements to achieve these objectives. The research has begun to generate static balance equations and models for body forces and pressure calculations. These models are simply integrated in the present software of bio-mechanical simulation through the installation of the MATLAB program. MATLAB algorithms have previously been built as hard components of the human body that use averages data from various studies. The code is input for a prospective wheelchair user orientation with the height and weight of the potential user in the wheelchair (Kwak, 2019)

Such a framework is provided by a dynamic motion simulation that incorporates models that describe the anatomy and physiology of the parts of the neuromusculoskeletal and multi joint movement mechanics. Muscle-driven dynamic simulations supplement experimental techniques by evaluating crucial, experimentally challenging variables, such as muscle forces and joint forces. Simulations also identify causation-effect correlations and let 'what if?' experiments to be conducted which, for instance, can change the muscle's excitement pattern and watch the subsequent motion. Although it is well known the importance of dynamic movement simulations, the area is fragmented. Many labs build their own simulation software, therefore it is impossible to utilize or assess this simulation outside the lab, where it is developed. It is tough to evaluate this program. The failure to deliver results is a key constraint on the advancement of biomedical simulation science. (Noah, 2018)

Simulation technology has been elegantly supported by individual researchers, including the development of new methods for modeling muscles, simulating contact and showing muscular skeleton geometry, but it is difficult for others to use those new techniques because software is generally unavailable. Since software tools are not publicly available to support muscular skeletal dynamic simulation development, analysis and control, researchers usually need to invest much money developing and building tools to study each new simulation. (Riley., 2017)

The production of dynamic movement simulations is technically difficult and many movement science laboratories lack the funding or technical competence to produce their own simulations. These circumstances represent an important obstacle to the advancement of simulation technology and the scientific capacity of muscle skeletal simulations. Delp and Loan created in the early 1990s a musculoskeletal modeling system, dubbed the SIMM, enabling users to build, modify and assess many various architectures of musculoskeletal structures. Hundreds of biomechanics researchers currently utilize this software to develop computer models of musculoskeletal systems and to mimic motions like as walking, cycling, ride and escalation. SIMM was used to construct models of both the lower and upper ends to investigate biomechanical effects of surgery, including tendon surgery, osteotomy, and complete joint replacement. In order to quantify the muscle-tenderon lengths, speeds, arms momentum and acceleration during normal and pathologic walking, a lower extremity model was utilized. (Tilley, 2018)

Studies have been carried out to study the treatment of people with a spinal cord injury, the analysis of joint mechanics in pain patients, calculating knee forces in running and cutting and to explore the effect of foot placement and compliance with the joint on ankle strains. These investigations proved the usefulness of musculoskeletal models and dynamic simulations in the analysis of the causes and consequences of gastric anomalies. SIMM has assisted simulate researchers who built frog, tyrannosaur, cockroach and other animal computer models (Tilley, 2018)

.

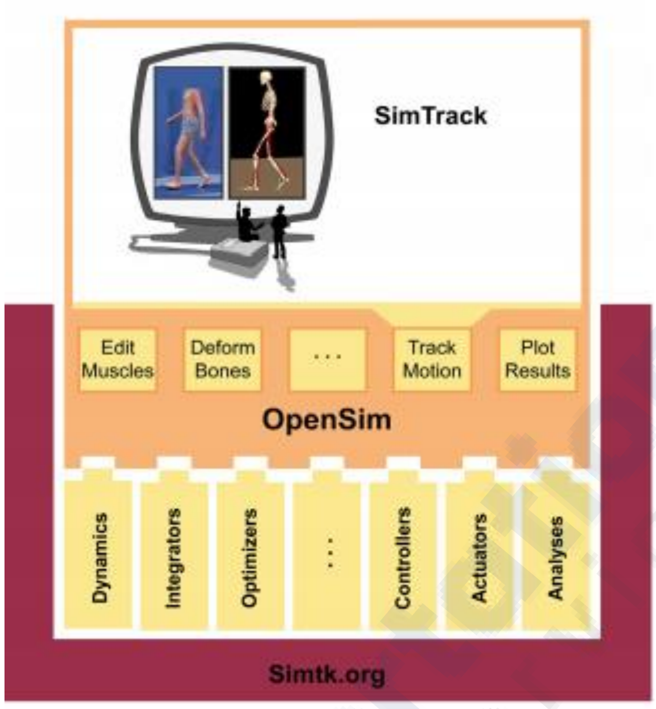


Figure 1 OpenSim Scheme, Software of Open Source

OpenSim schema, an open source software framework for neuromusculoskeletal systems modeling, simulations and analysis, OpenSim is constructed on the top of key computational components that allow movement equations to be derived, numerically integrated, and restricted non-linear problems to be solved. Additionally, OpenSim gives access to control methods, for example, computed muscle control, actuators such as muscle and contact models as well as analyses of accelerations caused by muscles, for example. OpenSim incorporates the components into a platform for modeling and simulation. By developing its own plug-ins for analysis or control or for representing neuromusculoskeletal components such as muscle models, users can enhance OpenSim. The user has access to a set of advanced tools to inspect models, to change muscles, plot results and other tasks on a graphical user interface. (Wade., 2019) SimTrack, one of the tools OpenSim, allows for precise muscle-driven simulations representing the dynamics of each subject. On Simtk.org, OpenSim is created and maintained; all software is provided free of charge. While SIMM helps to construct models of the musculoskeletal system and dynamic movement simulations, it does not help calculate the muscle stimuli that cause coordinated movement and has limited capabilities to analyze outcomes from dynamic simulations. In addition the complete access to source code for SIMM and other business packages, such Visual 3D C-Motion Inc., Anybody Technology or Adams MSC Software Corp., makes it difficult for biomechanics researchers to increase their capabilities. During the last 10 years, new methodologies for software engineering have developed, allowing the construction of more expandable software systems. This is a chance to build a simulation platform involving a larger range of biomechanics. In order to expedite the creation and sharing of simulation systems and to better integrate dynamic simulation in movement research. The development of open source software has become an excellent approach for both business and academic activities, e.g. the operating system Linux. Making accessible source code allows researchers, by making modifications and adapting code to fit their needs, to duplicate results produced by other laboratories. (Tilley, 2018)

Modern plug-in technology that we have implemented allows users to increase program capabilities and make it easier to share new solutions. We think that via an open-source initiative, the biomechanics community will benefit from an increase in collaboration. The first development workers need to offer the tools that others may use and expand to help build and test open-source software. OpenSim offers two of them. The first consists of a series of modeling and analysis instruments comparable to the SIMM tools. Second, SimTrack, allows researchers to produce dynamic movement simulations of motion collection data. (Ethan, 2017)

A brief summary of OpenSim is the first thing in this report. We focus on SimTrack and how simulations that describe each patient's dynamics might contribute to the treatment approach. A technique is presented and a case study is done using a dynamic simulation of a patient with stiff knee gait, through which the underlying reasons of the patient's aberrant movement as well as the implications of various therapies are understood. Finally, we look at the problems in the field. (Ethan, 2017)

2. Objectives and Aims

The research aims to measure the moment and dynamic forces of the human subject while sitting. And this initiative has further goals;

- Mat LAB stimulation of dynamic data and movement
- Mat LAB data development and OpenSim stimulation data integration
- Reverse film and dynamic analysis conducted in OpenSim (Dan., 2017)

3. LIERATURE REVIEW

3.1 OPEN SIM

OpenSim is an open source platform for neuromuscular skeletal system modeling, simulation and analysis. It contains low-level computer tools that a program invokes. A graphical user interface offers access to the essential features. A growing group of people is developing and maintaining OpenSim on Simtk.org. Simtk.org provides as a public resource for data, models and computational instruments for biological structure modeling based on physics. The program is ANSI C++ written and the graphical user interface has Java, which allows OpenSim to build and execute with common platforms. For some fundamental functionality, Open-Source third-party tools like Xerces Parser from the Apache Foundation for reading and writing XML files (xml.apache.org/xerces-c) and the Kitware Visualization Toolkit are utilized (www.vtk.org). Using the plug-in approach, computational components like dynamic engines, integrators and optimizers may be upgraded without major restructuring as necessary. For example, OpenSim is presently using SD Fast as the dynamic motor (Parametric Technology Corp.); but, future versions will let Simbody TM to also be utilized. Simbody TM is a developed, open-source dynamics motor at Simtk.org (Ethan, 2017)

.

Dissertation Help Services

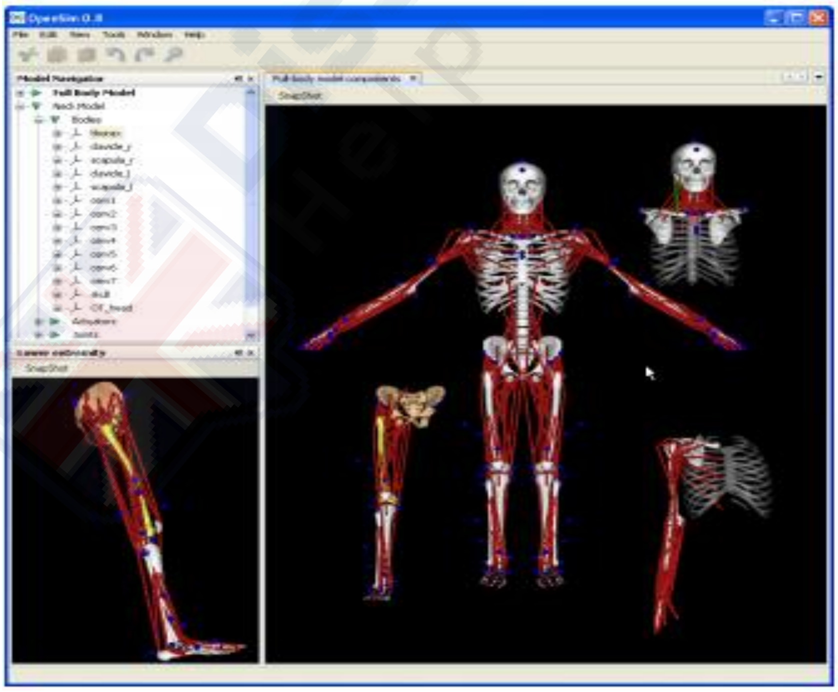


Figure 2. OpenSim display, the lower extremities, upper extremity and the neck may be loaded, examined and analyzed on models of various musculoskeletal structures. Muscles are displayed as red lines; blue spheres are depicted as the virtual mark

OpenSim's plug-in design enables users to expand their own muscle models, touch models, controllers and analytics. For example in OpenSim there are around one dozen analytical plug-ins written by different users. These tools compute joint forces, accelerations generated by muscles, muscle forces and other factors. These analyses are designed for several models of the musculoskeletal, but they may be used in general with any OpenSim model. The OpenSim plug-in design thus offers the biomechanical community a way to quickly distribute new capabilities. A new C ++ class (e.g., Induced Acceleration) from the suitable basic class (e.g., Analysis) must be added to the complement, a number of methods necessary must be implemented, and a class should be compiled into a dynamically connected library, the new plug-in may be utilized for simulations and shared with other users (e.g., Induced Acceleration Analysis). Plug-ins can also be built to improve the functions of the graphical user interface on their own. Almost all functionality from plug-ins comes through the user interface. The modules for viewing, charting and modifying of movements, for example, are all plug-ins. (Ethan, 2017)

Other users will have a user interface plugin example. The OpenSim graphical user interface offers toolboxes for the creation, generation and visualization of musculoskeletal models. A large part of Simm's functions, such as handling joints and editing the muscles and the variables of the plot, will become available in OpenSim. Furthermore, you may import SIMM joint (*.jnt) and muscle (*.msl) files. OpenSim offers non-SIMM simulation and control capability. In specifically, SimTrack is a tool able to quickly and correctly generate muscle-acted simulations of subject-specific movement as stated above. (Tilley, 2018)

3.2 SIMTRACK: AN OPENSIM DYNAMIS GENERATING TOOL

In order to create a muscular movement simulation, a dynamic model of the muscle skeletal system and its environmental interactions must first be created. Models comprising a series of differential equations describing muscles, muscles, skeleton and segmental dynamics are used for the constituents of the musculoskeletal system. These equations define the muscular-skeletal system's time-dependent conduct in response to neuromuscular excitement. The next stage is to identify a pattern of muscle excitations which produces a coordinated movement when a dynamic model of the musculoskeletal system is formed. (Pandy, 2018)

A solution to an optimization issue can be achieved by defining the purpose of a motor task (e.g., to leap as high as possible) or to drive a dynamic model to "follow" experimental move data. Simulations are usually assessed by how well they are in agreement with kinetics, kinetics and EMG patterns empirically recorded. Once a simulation is made and its correctness is validated, the contribution of a muscle to bodily movements and the impacts of a simulated therapy may be analyzed. The identification of a collection of muscle excitations that lead to coordinated movement is one of the biggest problems in dynamic simulations. Historically, the expense of producing co-ordinated movement simulations in muscles has been significant, with computer times taking days, weeks or months. Recent developments have substantially decreased the time required to produce such simulations in the application of robotic control systems for bio-mechanical simulation. (Byrd, 2018)

The computed muscle control technology for example determines muscle excitation that replaces observed pedaling dynamics in about ten minutes. This is quicker than traditional dynamic optimization methods than two orders of magnitude. This method, developed by

Thelen and Anderson, led to a 22-degree freedom 91 muscle model to follow the experimental data of ten individuals in health. Their approach included calculating muscle excitation patterns. In around 30 minutes, a simulation of half a gait cycle was created. The speed of this approach enables a broad range of motions to be generated with precise subject-specific simulations. (D. A. Winter, 2009.)

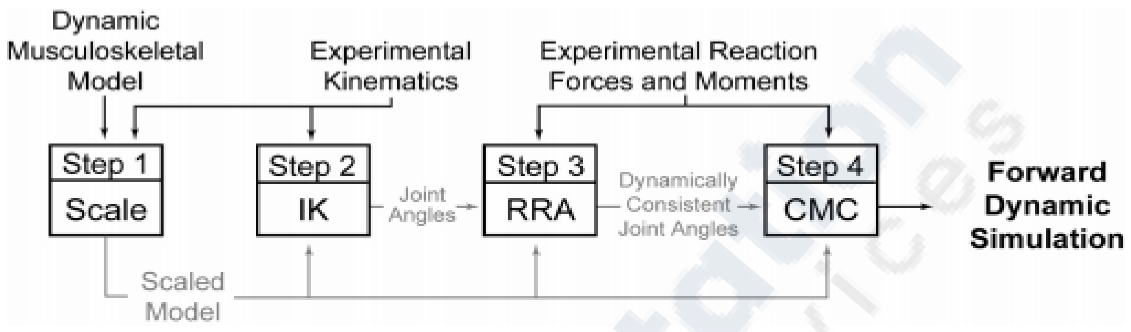

Dynamic Musculoskeletal Model
Experimental Kinematics
Experimental Reaction Forces and Moments
Step 1
Scale
Step 2
IK
Joint Angles
Step 3
RRA
Dynamically Consistent Joint Angles
Step 4
CMC
Forward Dynamic Simulation
Scaled Model

Figure 3Methods to construct a musculature movement simulation with SimTrack. The inputs include a dynamic muscular skeleton model, experimental kinematics and the experimental reaction force and moment of the subject. x-y-z marker data, joint centers, and joint

- Experimental kinetics is utilized in step 1 to measure the musculoskeletal model in order to match the topic dimension.
- In step 2, the problem of reverse movies (IK) is resolved to determine the joint angles of the model best for experimental filmmaking.
- A RRA (residual reduction algorithm) is employed in Step 3 to modify the model kinetics so that it conforms to the experiential response forces and moments more dynamically.
- In Step 4, a CMC algorithm serves to detect a collection of muscle excitations that produce a forward dynamic simulation that closely matches the subject's movements.

Dynamic simulation may be created in four phases via SimTrack As an input, SimTrack uses a dynamic model of the musculoskeletal system and testing of movie and reaction forces and moments. Although this is a generic method, we discuss it as part of gait simulations, as this is one of the more demanding applications. Step 1 sets up a model of dynamic muscular skeleton (such as a model of SIMM) to fit an individual subject's anthropometry. (Dan., 2017)

The dimension of each segment of the model in the body is scaled depending on relative distance from a motion capture system between pairs of markers and the corresponding virtual marker positions in the model. The mass characteristics of the bodily segments are proportionately reduced to replicate the total mass measured by the individual. Muscle fiber lengths and tendon lengths are scaled to remain the same proportion of the whole actuator duration. (Dan., 2017)

Step 2 solves an issue using reverse kinematics (IK) to establish the model of generalized co-ordinates (joint angles and translations) that reproduces best the raw marker data derived from movement capture. Step 2 is framed as a minor problem which minimizes discrepancies, subject to joint restrictions, between the measured marker sites and the model's virtual marker sites. If a set of joint centers or joint angles created by motion capture software are incorporated into experimental kinetics, these may also be included in the formulation. The reverse kinematics challenge is therefore to minimize the weighted squared error on each frame in experimental kinematics: (Kwak, 2019)

$$Squared\ Error = \sum_{i=1}^{\text{markers}} w_i(\vec{x}_i^{subject} - \vec{x}_i^{model})^2 + \sum_{j=1}^{\text{joint angles}} \omega_j(\theta_j^{subject} - \theta_j^{model})^2$$

As a result of experimental errors and modeling assumptions, the observed ground response forces and moments frequently differ dynamically from model movies. In step 3, a residual reduction algorithms (RRAs) are performed in order to make the models more dynamically compatible with their measurement of ground reaction forces and moments in generalized co-ordinates (joint angles and translations). The following equation in Newton's second law covers the determined ground response strength and gravitational acceleration with body segment acceleration (Kwak, 2019)

$$\vec{F}_{external} = \sum_{i=1}^{segments} m_i \vec{a}_i - \vec{F}_{residual}$$

A comparable equation covers the instant of the soil interaction with the cinematic model and the rest. The residual force should be zero in the absence of experimental and modeling error. This is never the case in practice. With the combination of tiny controlled disturbances of the motion path and small modifications of the model's mass parameters, the residual forces and times for dynamic consistency may be reduced. The residues are computed and averaged across the period of the motion in order to decrease the remaining forces and moments. For example, the calculated muscle control technology determines a muscular excitement which takes around ten minutes to replace the observed pedaling dynamics. This is faster than two magnitude orders than standard dynamic optimization approaches. This technique, created by Thelen and Anderson, resulted in a ninety muscle model thirty two degrees freedom, which follows the experimental data of twenty healthy persons. Their technique involved calculating the patterns of muscular arousal. A simulation of a half-stroke cycle was developed in around twenty five minutes. The speed of this method allows for accurate object-specific simulations to produce a large number of movements (D. A. Winter, 2009.)

The algorithm proposes modifications in the model mass parameters such as the position of the center of the trunk mass, which lower the average residual value over the time of movement, based on such averages. After all mass parameters have been changed; a control problem with all degrees of freedom in the model is solved. The joints are controlled by idealized joint times in particular and, in addition, a chosen model segment is controlled by

Dissertation Help Services

the six degrees of freedom between the model and the ground by three residual force and three residual time (i.e., three translations and three rotations). If the residues are not limited, the movie industry can be traced by little or no mistake. At the user's option, however, the magnitude of the residue can be set at upper bounds, whilst the model movement is adjusted, resulting in a new set of movies that dynamically conform to the limited residues. To spread tracking faults via joint angles a performance test is employed (Noah, 2018)

$$\text{Squared Error} = \sum_{j=1}^{joints} \Omega_j \cdot (\ddot{q}_j^{desired} - \ddot{q}_j^{model})^2$$

The results of the residual reduction technique are utilized for the model levels of freedom and mass characteristics in Step 4. Step 4 uses CMC to generate a sequence of muscular excitations that form a co-ordinated simulation of the movement of the subject, which is driven by muscle. A static optimization criterion is used to distribute forces through synergistic muscles and proportional derivative control in order to provide a forward dynamic simulation that closely monitors the film industry produced in step 3. Although the entire status equations defining muscle activation and contraction dynamics are utilized as a static performance criterion, they are integrated in the forward dynamic simulation. Activation dynamics are modeled on the temporal rate of change in muscular strength and excitement (Coleman, 2020)

$$\dot{a} = \begin{cases} (u-a)\cdot[u/\tau_{act} + (u+1)/\tau_{deact}] & u > a \\ (u-a)/\tau_{deact} & u < a \end{cases}$$

A lumped parameter model describes the contraction dynamics in the musculotendon, which account for the muscle's strength-length-velocity properties and the tendon elastic characteristics. The rate of change of muscle length is especially linked to muscle length, length of the actuator of muscular tendon and activation of the muscle. (Tilley, 2018)

$$\dot{l}_m = f_v^{-1}(l_m, l_{mt}, a)$$

Our current technique does not simulate the strength between the foot and the ground but instead directly applies the observation of the ground reactions forces and moments. The spring damping elements are given between foot and ground in the analysis to allow the reaction forces to adapt to disturbances, as explained in the case study below. (Pandy, 2018)

4. METHODOLOGY/ EXPERIMENTS / DESIGN

Biomechanical study of the human body was performed in different seating postures by creating free body diagrams. This enabled the strengths of the supports to be implemented and computed using the following equations:

$$\sum Fx = 0$$

$$\sum Fy = 0$$

$$\sum TA = 0$$

Where

Fx= force in x direction

Fy= force in Y direction

TA= torque of an arbitrary point in the model

4.1 Model of Body Segment

In order to depict the complete corps, the body must be divided into rigid body parts and drawn as direct lines. The segments of the corps were: the skull, the chest, two uppers, two forearms, two hands, two thighs, two shanks and two feet. The length, position, weight and COG placement of each body section were its own. The length and position of each body segment were estimated as a share of the human body's overall height entered in the model. Thus, the body segment lengths and positions of a fifty percent man were divided by eight percentages by a fifty percentile male in total (Ethan, 2017)

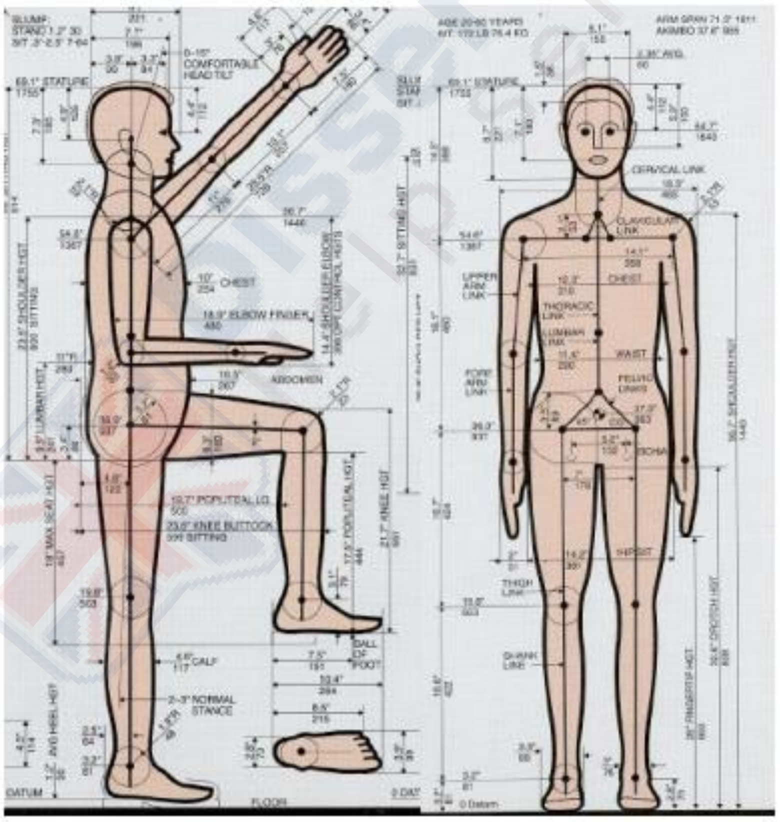
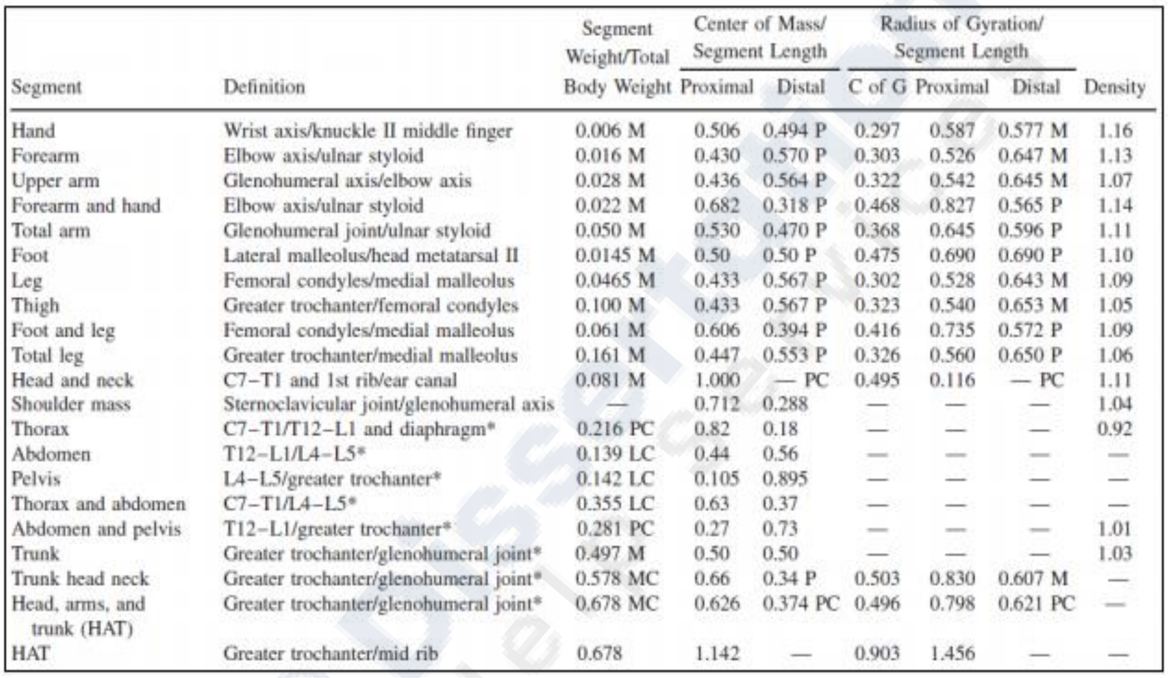


Figure 4(a) 50th male percentile side view, (b) male 50th percentile front view

The information from the widely regarded book The Measure of Man and Woman: Human Factors in Design, for the lengths and places of the body of a fifty percent male was derived. (Kwak, 2019)Figures four (a) and four (b) provide a schematic of the fiftieth percentile man in this book. The weight of each segment has been computed as a percentage of the body's entire weight, whereas the COG position has been calculated as a percentage of every segment's length from the proximal end. The body must be split into stiff sections of the body and represented as direct lines in order to represent the entire body. The body was divided in sections: the cranium, the thighs, two uppers and two forearms, two hands, two thighs, two shanks. Each body part has its own length, location, weight and COG placement. As a part of the total human body height input in the model, the length, location of each section of the body was calculated. Thus, the lengths and locations of the body segment of a thirty percent male were split by nine percent by a forty percent male. (Pandy, 2018)



		Segment Weight/Total	Center of Mass/ Segment Length		Radius of Gyration/ Segment Length			
Segment	Definition	Body Weight	Proximal	Distal	C of G	Proximal	Distal	Density
Hand	Wrist axis/knuckle II middle finger	0.006 M	0.506	0.494 P	0.297	0.587	0.577 M	1.16
Forearm	Elbow axis/ulnar styloid	0.016 M	0.430	0.570 P	0.303	0.526	0.647 M	1.13
Upper arm	Glenohumeral axis/elbow axis	0.028 M	0.436	0.564 P	0.322	0.542	0.645 M	1.07
Forearm and hand	Elbow axis/ulnar styloid	0.022 M	0.682	0.318 P	0.468	0.827	0.565 P	1.14
Total arm	Glenohumeral joint/ulnar styloid	0.050 M	0.530	0.470 P	0.368	0.645	0.596 P	1.11
Foot	Lateral malleolus/head metatarsal II	0.0145 M	0.50	0.50 P	0.475	0.690	0.690 P	1.10
Leg	Femoral condyles/medial malleolus	0.0465 M	0.433	0.567 P	0.302	0.528	0.643 M	1.09
Thigh	Greater trochanter/femoral condyles	0.100 M	0.433	0.567 P	0.323	0.540	0.653 M	1.05
Foot and leg	Femoral condyles/medial malleolus	0.061 M	0.606	0.394 P	0.416	0.735	0.572 P	1.09
Total leg	Greater trochanter/medial malleolus	0.161 M	0.447	0.553 P	0.326	0.560	0.650 P	1.06
Head and neck	C7–T1 and 1st rib/ear canal	0.081 M	1.000	— PC	0.495	0.116	— PC	1.11
Shoulder mass	Sternoclavicular joint/glenohumeral axis	—	0.712	0.288	—	—	—	1.04
Thorax	C7–T1/T12–L1 and diaphragm*	0.216 PC	0.82	0.18	—	—	—	0.92
Abdomen	T12–L1/L4–L5*	0.139 LC	0.44	0.56	—	—	—	—
Pelvis	L4–L5/greater trochanter*	0.142 LC	0.105	0.895	—	—	—	—
Thorax and abdomen	C7–T1/L4–L5*	0.355 LC	0.63	0.37	—	—	—	—
Abdomen and pelvis	T12–L1/greater trochanter*	0.281 PC	0.27	0.73	—	—	—	1.01
Trunk	Greater trochanter/glenohumeral joint*	0.497 M	0.50	0.50	—	—	—	1.03
Trunk head neck	Greater trochanter/glenohumeral joint*	0.578 MC	0.66	0.34 P	0.503	0.830	0.607 M	—
Head, arms, and trunk (HAT)	Greater trochanter/glenohumeral joint*	0.678 MC	0.626	0.374 PC	0.496	0.798	0.621 PC	—
HAT	Greater trochanter/mid rib	0.678	1.142	—	0.903	1.456	—	—

Figure 5Relative anthropometric data on body segments and COG locations

4.2 Modeling Software (MATLAB)

This biomechanical analysis was carried out using a number of software tools. OpenSim was a student of both academics and Bio pathology students at the University of Ohio who initiated the study on a simple biomechanical 3D software program. This software has caused problems, however, because OpenSim is best used for modeling human body dynamics to model and forecast joint torques and muscle activation in the body, rather than to detect static external forces on the body as a result of other body pressures. (Noah, 2018)

MATLAB was also studied because of its significant use by engineering studies and academics at The Ohio State University. MATLAB is an easy to use programming language, which enables numerous operations to be carried out fast and efficiently. On the MATLAB website, Math Works, the developer, says: "MATLAB is tailored for addressing engineering and scientific issues, Anthropometric data on relative masses and COG sites in the body segments. MATLAB is the most obvious means of expressing computer mathematics worldwide. Integrated graphs allow the visualization of data and the obtaining of insight." It was chosen to utilize MATLAB for the modeling of the human body, as it had the straightest interface and all the instruments required for external force calculation. (Dan., 2017)

4.3 Description of MATLAB Code

For this study, the final code of the MATLAB, as listed in Annex A, was divided into two views of the body: the lateral view which modeled the coil, backrest and base and the front view allowing the coil, footrests and support for the sides of the trunk to be modeled. Both scripts have been executed with inputs of the overall person's height and weight and particular joint angles that the program user wanted to see the body. The angle at the pelvis, knees, ankle, shoulder, elbow and wrist were part of these joint angles. Inputs to both the sagittal angles of the plane (side view) and to the frontal angles (front view) were incorporated in the pelvis and on the shoulders. (D. A. Winter, 2009.)

4.4 OpenSim-MATLAB interface

With the OpenSim API, OpenSim was interfaced with MATLAB. MATLAB serves to address issues with optimization and OpenSim represents musculoskeletal system dynamics. For multi body dynamics and other numerical processes OpenSim itself relies on the Simbody dynamics engine. On the block in Fig. 6 called the "State Derivatives," which match with Euler discretionary scheme given by Eq, is the crucial link between OpenSim and the MATLAB, OpenSim for a specific collection of discreet states and controls by evaluating Eq. The fi+1 vector term values in Eq. may be determined. If the value of the target function requires the magnitude of any amounts which are implicit functions and controls, for example contact forces, muscle powers, they may also be acquired from MATLAB by using the corresponding OpenSim methods. (Wade., 2019)

.

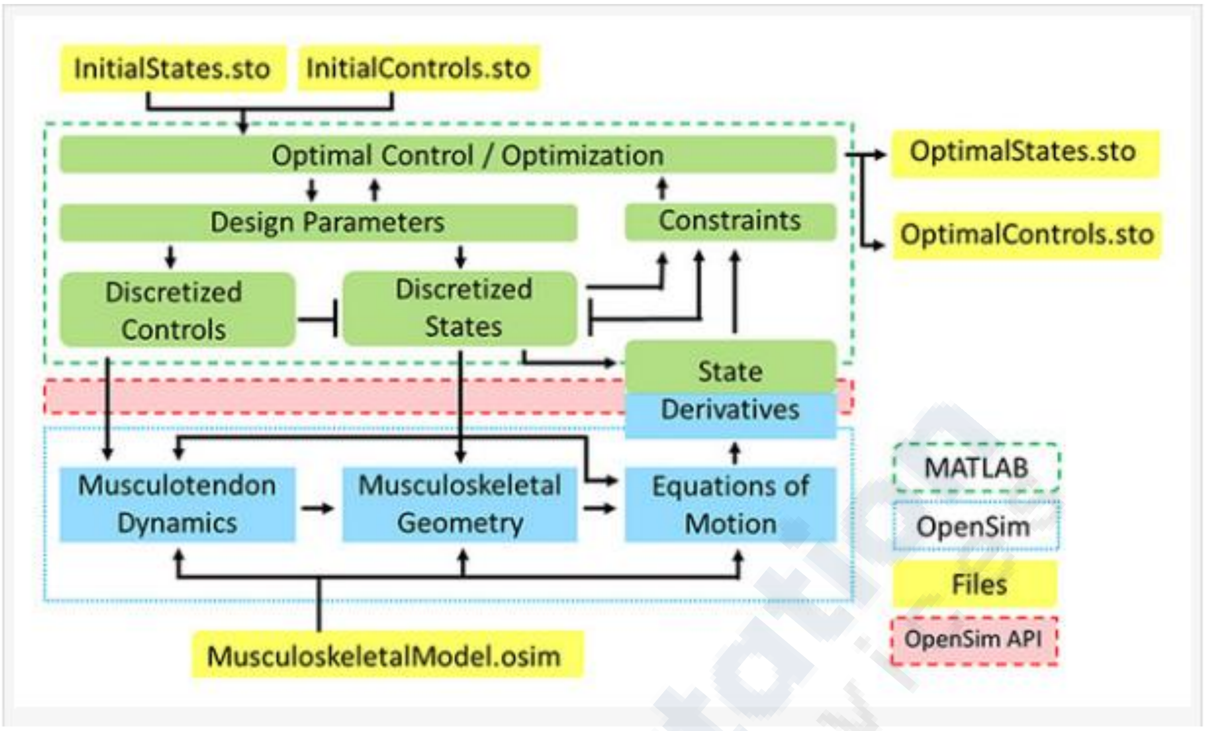


Figure 6 OpenSim, MATLAB interface with direct collocation for the solution of optimum check issues.

MATLAB and OpenSim are communicated using the OpenSim API. The green boxes reflect MATLAB's optimization process. The blue boxes indicate OpenSim's computer processes. Files input and output are shown in the yellow boxes. The "state Derivatives" green-blue box symbolizes the discretion in the approach to direct communication. In the OpenSim graphical user interfaces the original guess and the optimum outcome may be viewed. (Kwak, 2019)

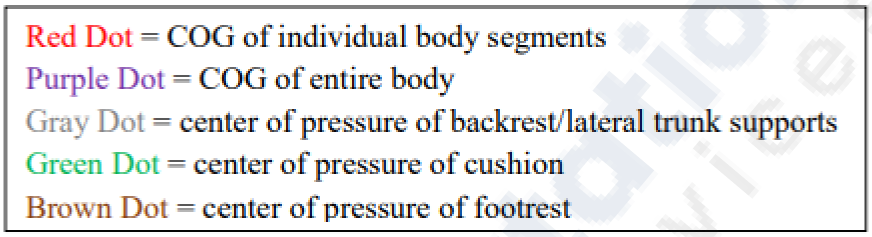
The first guess for optimization settings is found in two files called Initial Status.sto and Initial Controls.sto in from OpenSim storage (.sto). The optimization findings are written in into two comparable files, Optimal Status.sto and Optimal Controls.sto. In the OpenSim graphical user interface, beginning devices and end outcomes may therefore be easily seen (GUI). Optionally, intermediate outcome files can be produced to enable an intermediate motion in OpenSim GUI as optimization continues. The storage files with final findings make it easy to do the forward dynamics simulations using the OpenSim Forward Dynamics tool based on the DC results. Further simulations in the Optimal States are produced utilising the states from the previous time. This file from every time point in the Optimal Controls as beginning conditions and the muscle excitations. As controls, store files. (Ethan, 2017)

5. Results / Findings

Test of weight the scale beneath the underside was 124.8 pounders when the participant's arms were completely extended, while the scale below the participant's feet was 45.4 pounds. The scale below the underside was 135.4 pounds, whereas the scale below the participant read 28.2 pounds, when he had his arms hung by his sides. The COGs of the hands were positioned under the feet over a scale while the participant's arms were completely extended. These outcomes permitted: Test configuration with fully extended arms and scale underneath the subject. Determined which supports weight of the specific body support in the position of the COG of a segment of the body not in direct touch with a support. The computation of the strength from footstool will include the weight of the hands if, for example, the COGs on the hands of the modeling body are situated above the footstool. (Pandy, 2018)

5.1 MATLAB Program Output

By entering the value at the command window of MATLAB for easy reading the output of MATLAB codes gives the user of the program the numerical values for points connected with each support. Furthermore, on a plot with grid lines, a graphical representation for the body's free diagram at the input position represents the special co-ordinates of each body segment. Figure 8 marks the following caption, which defines each symbol in the output plots: (Kwak, 2019)



Red Dot = COG of individual body segments
Purple Dot = COG of entire body
Gray Dot = center of pressure of backrest/lateral trunk supports
Green Dot = center of pressure of cushion
Brown Dot = center of pressure of footrest

Figure 7Legend for MATLAB free body diagrams.

5.2 Case Study

We utilized a basic first degree-freedom (DOF) model consisting of a block acting on by two muscles to show the technique utilising the OpenSim-MATLAB interface, resulting in a model with six states and two controllers Fig. 2A. This model may translate along the medio lateral Z axis of Fig. 2A because the two muscles, the one pushing the other in a negative direction, act on them. From the Tug of War.osim example supplied with OpenSim was changed the simple pattern. The block has 6 DOF and 5 restrictions to generate uniaxial sliding in the original example model. In order to alleviate the limitations and decrease the state space, we have replaced the 6 DOF free joint with a first DOF slider joint. We have also adjusted the slack lengths of the tendon from the example such that the muscles function closer to the strength-length curve plateau for the simulated movement task. The muscles had a maximum isometric strength of 1000 N for this investigation, an optimum fiber length of 0.251 m, 0.051 m slack tendon longitudes and 0° cleavage angles. (Tilley, 2018)

.

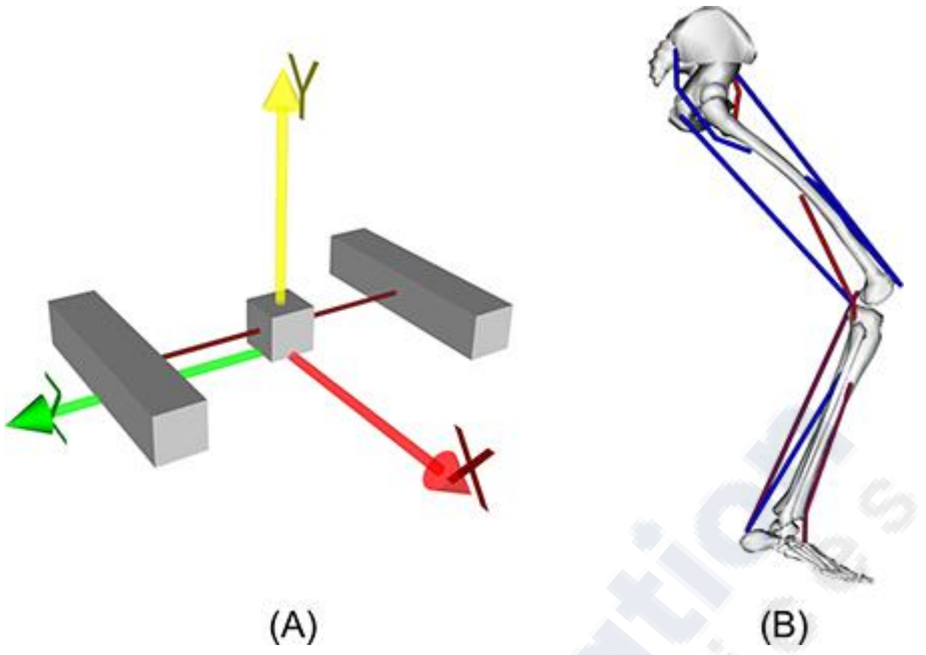


Figure 8this project uses OpenSim models.

Predictive simulations were created in which the goal motion for the block should begin at rest from −0,08m in the mediolateral axis, translate to a position of 0,08 m in half of the period of movement, and return to its original condition over a total time of 1,0s. Apart from these task restrictions at the first time, midpoints and the end time, the actual movement was not specified. Other limitations were imposed in order to ensure that states and controls were in line at the original time. The aim was to minimize the sum of the integrals of square muscle activation (Blake, 2019)

$$J = \frac{1}{T}\sum_{i=1}^{m}\int_0^T a_i^2(t)dt$$

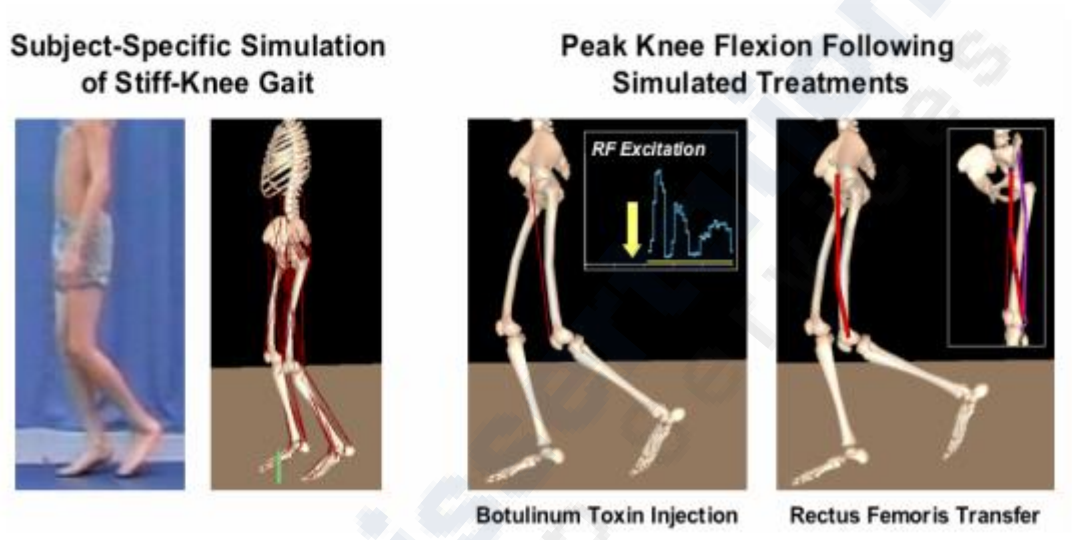
The number of muscles is where **ai** is immediate activation of the ith muscle and m. In a variety of grid densities, 25–501 nodes addressed the NLP issue (25, 51, 101, 151, 201, 301, 401 and 501 nodes). Solutions for all grid densities and for fmincon (interior-point method) up to 201 nodes were achieved for IPOPT. In order to deal with such a tiny question, the calculated duration of fmincon on the denser grids was too long (>1 day). (D. A. Winter, 2009.)

An initial estimate was created when the model started statically in the original 0.0 m location with the 1.0 s forward simulations and did not move, because the muscle controls were both set to zero. This is called the "static" initial assumption. With IPOPT, two alternative approaches had been addressed for the NLP problem: one utilising a grid refinement technique using a static starting device for all grid densities. For fmincon, the NLP problem was handled only with grid refining, as convergence with the static initial estimate was too slow. The original estimate for a given grid density was that from the next lower (that is to say, coarser) grid density a solution was achieved, with the exception of the 25 node grid where there was no lower grid density. For example, with the static starting estimate and with an ideal outcome for the 51 node grids, the 101 node case was resolved. We've always utilized an unusual amount of nodes for this specific movement job because of the mid time limitation. For the 25 nodes, 200 unknowns and 156 restrictions were present, but for the 501 node example, 4008 unknowns and 3012 restrictions were present. The solutions were evaluated by a comparison of results from the different grid densities and by comparing results with future simulations at each grid density based on optimum controls and ideal beginning conditions derived from the DC optimizations. (Riley., 2017)

In order to assess the causes of their aberrant walking pattern and model therapy possibilities, we have developed dynamic simulations of individual patients with abnormal gait utilising calculated muscle control. This case study shows how simulations may give us a sense of the factors that generate steep knee gait, which significantly diminishes swinging-phase knee flexion. (Riley., 2017)

Reduced knee bending is typically due to increased excitement of the femoris rectus during the swinging phase. However, variables such as excessive strength in vasti or rectal femoris or reduced power in iliopsoa or gastrocnemius that restrict knee bending speed just before swinging can also help to minimize knee flexion during swing. It is hard to establish which of the variables restrict the knee bending of a person as present techniques of diagnosis cannot assess how pressures created by rectum femoris or another muscle impact knee movements of swing-phase. The therapy of steep knee gait has various possibilities. Theoretically, injecting botulinum toxin reduces the moment of hip and knee caused by rectus femoris. Determined to sustain the weight of the particular body support in the COG position of a segment of the body not directly related to the support, For example, when the COGs on the modeling body are located on top of the footstool, the calculation of the strength from footstool should include the weight of hands. (Dave., 2018) Weight test when the participant's arms were fully extended, the scale below the underground was 124.8 pounders, with the participant's feet below 45.4 pounds. The bottom scale was PL 135.4 while the bottom scale read PL 28.2 when the subject had his arms hung at his sides. The hand COGs were positioned over a size below the feet while the arms of the subject were stretched fully. These results allowed for: Test setup and scale below the person with fully extended arms. A second alternative, the transfer of rectum femoris, reduces theoretically the moment of muscle extension while maintaining its moment of hip bending intact. At present the processes that enhance the swing-phase knee bending of patients are not fully known following these therapies. In this case study, the biomechanical reason of his declining knee flexion and the possible implications of alternative treatment options were created and assessed using dynamic simulations of the stiff-knee gait subject. (Dan., 2017)

A 12-year-old male with spastic paralysis was diagnosed. The topic, during swing and aberrant activity of the femoris (preswing & swinging) and the vasti the lower left leg showed restricted knee flexion (preswing). The musculoskeletal system of the individual was represented by a scaled, 21° freedom connection, which was operated by 92 muscles, and we produced a dynamic forward simulation. The measured knee flexion angle of the individual was recreated at 2° using a simulated joint angle. By changing the muscle excitations in the simulation and calculating the resultant changes in peak bending, we assessed rectal femoris, vastial, and other muscles' contributions to knee flexion. Analysis of this dynamic simulation showed that the main cause of the stiff-knee-gout was excessive knee extensor activation in preswing, reducing rectus femora’s or vasti stimulation during preswing, significantly increasing peak knee bending. Reducing rectal femoral excitement in the early swing had a minimal influence on peak knee bending (Coleman, 2020)


Subject-Specific Simulation of Stiff-Knee Gait
Peak Knee Flexion Following Simulated Treatments
RF Excitation
Botulinum Toxin Injection
Rectus Femoris Transfer

Normal walking simulation has enabled researchers to discover the activities of muscles that exceed knowledge acquired via experimental approaches by means of a degree of precision and confidence. Aberrant walking simulations have the same potential, but in part they are difficult to build since they need to determine muscles that cause the abnormal dynamic of motion that people with movement disorders experience. The calculated muscle control technique gives a computer efficient way to produce these simulations and is currently available worldwide for academics. (Wade., 2019)

6. Discussion

Researchers have used the MATLAB interface to build a novel framework to resolve optimum muscular control challenges with the OpenSim API. The fmincon solver also solved the basic model optimum control issue, but the calculation time was too long to be generally useful. We seek to facilitate the use of predictive biomechanical models to tackle clinically relevant human mobility problems. MATLAB is effectively coupled with OpenSim musculoskeletal Modeling, Simulation and Analyze tools for the high programming, design and management capacity. With this in mind, the direct collocation approach has solved two forecast problems: a recurrent movement problem with a fundamental musculoskeletal model and a discrete motion problem based on the lower limb model that is more realistic. Both problems with acceptable computing requirements were resolved using an IPOPT solver. The fmincon solver could also solve the basic model optimum control issue, but the calculation durations were too lengthy for widespread application. Our objective is to allow use of predictive biomechanical simulation to solve challenges of human mobility that are clinically relevant. (K. Waugh and J. M. Bach, 2019)

6.1 Biomechanics Simulation Opportunities and Challenges

We think simulations will improve the science of movement through exchanges between modelers and experimenters. Experimentalists are required by modelers to obtain parameters used in simulations and to check the correctness of simulation findings. Experimentalists need to build models in order to evaluate experimental observations and acquire perspectives from the abundance of biomechanical experimental data. (Noah, 2018)

Biomechanical researchers are now able to build a quantitative, causative-effect relationship between the neuromuscular excitation patterns, muscle forces, external reaction forces and movements of the body that are observed in the lab by accessing open-source software to develop and analyses muscle-driven simulation. In combination with high-quality experimental observations, simulations can assist to understand the interactions between parts of the Neuromusculoskeletal System and the results of therapy for people with motion problem. (D. A. Winter, 2009.)

A number of software programs have been used to develop and analyses models for the lower extrem, higher extrem, cervical spine, lower back and other components of the musculoskeletal. Although these models are implemented in various modeling software, the model parameters are comparable. One difficulty is to create modeling standards and encourage the exchange of modeling packages. (D. A. Winter, 2009.)

Another difficulty is to show that using simulations may enhance therapy results for people with mobility problems. The possibility to better understand the reasons of movement aberrations and the treatment choices using subject-specific simulations is intriguing, and the above case study offers unique and important insights about rigid knee gait for one patient. Future investigations in order to evaluate if the principles for treatment planning can be explained through insights acquired from analysis simulations are required in order to carry out simulations for a number of individuals. Studies that compare predictions of subject-specific simulations retrospectively with the actual results of individuals are also needed to assess if current muscular-skeletal models are accurate enough and to determine criteria for the application of the results of simulations. The open-source simulation platform we have developed allows such extensive investigations, however more development is necessary in order to streamline the simulation process of impaired patients. Future clinical studies are ultimately necessary to evaluate if simulations may improve therapy results. (Dave., 2018)

New research opportunities are being offered by the ability to quickly build synchronized muscle driven simulations. Many earlier simulation researches have one single simulation findings. With SimTrack, 3D simulations of multiple individuals may be generated and analyzed, and standards can be set to describe muscle function for subjects of various sizes, strengths and patterns of movement. Sensitivity studies are also practical to discover if the findings from the simulation are responsive to modifications in model parameters. This is of particular use if a direct comparison is not practicable with experimental data (e.g. trajectories of muscle force). (D. A. Winter, 2009.)

Simulation-base tests may therefore be replicated and tested outside the lab for the first time. Such stringent testing are necessary in order to make biomechanical modeling more a science and less an art. It is a major problem to build a digital human (a computational model of the human neuromuscular skeletal system with a man-like complexity). If there are wide and complete models, then users can decide how they might reduce the model to solve a specific scientific issue. This problem and some of its key success advantages may be outlined in the Physiome Project. (Pandy, 2018)

As demonstrated by the case study, it is also feasible to analyse how impairments such as aberrant muscular excitement might lead to atypical movements in particular patients and how therapies can have functional repercussions. The precision of the simulation depends on the fidelity of the neuromusculoskeletal system's underlying mathematical model. In developing musculoskeletal models several assumptions are made and some of the assumptions are based on little experimental data. (Coleman, 2020)

In order to increase accuracy of muscular skeletal models, more in vivo muscle geometry and joint videos are needed. This enables us to learn how variations in size, age, deformation or surgery affect the model's prediction and how general model simulations are suited for each patient. Experiments that characterize the impacts of illness and surgery to generate muscle strength are necessary to check assumptions in musculoskeletal models and to assess their influence on movement.. In order to create simulations including sensory-motor control representations, neuroscience developments are essential. (Tilley, 2018)

Since simulations contain assumptions and approximations, it is essential that the constraints of each simulation be evaluated. As more researchers utilize muscular skeleton dynamic simulations, in the context of their unique scientific investigation it is vital that each scientist evaluate the correctness of its models. OpenSim offers additional cooperation and peer review options. A multi-institutional cooperation is used to test, evaluate and enhance the code that comprises OpenSim. Users are invited to alter the code accordingly and share their modifications with others. (Kwak, 2019)

Simulation-base tests may therefore be replicated and tested outside the lab for the first time. Such stringent testing are necessary in order to make biomechanical modeling more a science and less an art. It is a major problem to build a digital human (a computational model of the human neuromuscular skeletal system with a man-like complexity). If there are wide and complete models, then users can decide how they might reduce the model to solve a specific scientific issue. This problem and some of its key success advantages may be outlined in the Physiome Project. (Byrd, 2018)

The development of movement simulations emphasizes the limits of current movement capture data and shows the need for enhanced testing methods. A plethora of data is generated by muscle-based simulations. Simulations must be used to clarify the principles which control muscle coordination and produce superior clinical results so that insights from

Dissertation Help Services

these data may be revealed. The development and spreading of tools for analysing and visualizing which offer fresh insights offers a significant challenge to biomechanical simulation progress. Our objective is to provide a platform for biomechanics to develop tools for the detection of principles governing human movement and for the development of improved therapy for people with physical impairments. (Riley., 2017)

7. Conclusion Project

The OpenSim-MATLAB interface offers a robust and flexible technique for efficient simulations of musculoskeletal movements using DC.. It might enable optimal control to be applied in developing therapeutic illness therapies that restrict human mobility. This study used the new MATLAB interface for the OpenSim API. All the programming was done in the high level MATLAB environment, but for musculoskeletal and associate mathematical calculations the robust and efficient OpenSim C++ library was utilized. OpenSim itself is based on the engine Simbody dynamics, built on cutting-edge mathematical algorithms like LAPACK. Our usage of MATLAB in OpenSim interface differs from the technique previously documented when MATLAB has been connected via an S-function with OpenSim. The S-Function API was utilized for OpenSim in a Simulink block and subsequently for running both open and closed loop forward MATLAB/Simulink simulations. This technique is, in fact, complimentary and is just for different reasons. During MATLAB use, OpenSim API, much like the IPOPT solver, is also available via Python. (Ethan, 2017)

Python has numerous numerical and scientific computing capabilities as an open source high level programmed language. Therefore, the technique provided here should be replicated using either MATLAB or Python by other researchers. (Blake, 2019)

We employ DC to solve the optimum control issue in this project, although we have taken various different solutions for generating simulations of a number of human movements. The previous technique was that controls were only discrete with low or high dimensions. Then, dynamic equations were integrated forward to assess the goal function and constraints. Some new techniques include muscle reflex modeling and global muscle parameterization using the Fourier series. These alternative techniques may also be implemented using OpenSim and MATLAB and would be susceptible to many of the limitations and strengths discussed in this document. For researchers using OpenSim and MATLAB, the example code supplied by this paper might be a good starting-point. (Dan., 2017)

8. Forces of support

The MATLAB forces reflect the point forces linked to each of the supports. The point force serves as the core of the pressure distribution over the entire base; yet in actual fact it is not just on one place of the body but the strength is divided between the support and the body across the contact region. The pressure distribution is now represented on a point force to simplify the governing equation that the sum of the torques is zero. However, this proves to be problematic as each individual is different in the contact area and exceedingly difficult to quantify for a clinician. The average pressure associate with each holder may be determined by breaking point strength between the contact areas. (Dan., 2017)

9. Assumptions of Model

Assumptions of Model In the creation of the model there have been many assumptions because of limits in existing research on the human body and because of the unique characteristics of every human person. Because of the fact that all data have been obtained from a male of the 50th percentile, this model is not currently applied to people close to population extremes, such as 5th and 95th percentiles, or to people whose body shape or body weight distribution varies considerably from those of a male of the 500th. In addition, it cannot be used to people with postural abnormalities since, as oppose the curved nature of a healthy or malformed spine, the body is depicted as inflexible straight-lines. The model has certain broad assumptions. (D. A. Winter, 2009.) The footrest is without frictional strength but rather functions as two separate footrests with just regular strength, where there is one footrest behind the shank and the second footrest below the feet. The feet are always 90° to the shank, which allows for the perpendicularity of the two modelled footrests. There is no friction component of the backrest and so a frictional force must be provided in the cushion to ensure that the user does not slide off from the wheelchair front. Lateral trunk supports forces do not include friction components; thus, a friction force needs to be provided to ensure the wheelchair user does not slide off the wheelchair's side. All these assumptions had to be established in order to maintain static determination of the free body diagram of the body segments that implies that the free body diagram has the same number of equations as unknown variables, so that each variable, the forces and positions in this instance may be resolved. (D. A. Winter, 2009.)

10.Research Future

This research is still in its early phases, and new paths for future research in the same field were being developed as work was finished. Currently, two distinct MATLAB scripts are run independently of each other on both the side view and the front view of the model. Certain assumptions on the orientation of the body on the other aircraft have to be made inside each code. The model and plane angles can be combined to form a single code so that any orientation can be accurately represented, irrespective of the impossibility of a posture. (Coleman, 2020)This research is still in its early phases, and new paths for future research in the same field were being developed as work was finished. Currently, two distinct MATLAB scripts are run independently of each other on both the side view and the front view of the model. Certain assumptions on the orientation of the body on the other aircraft have to be made inside each code. The model and plane angles can be combined to form a single code so that any orientation can be accurately represented, irrespective of the impossibility of a posture. Rather, a static coefficient may be applied in a model to build a link between the force of friction and the normal force of support between the person in the wheelchair and the support in the corresponding case. Using this connection, the model should stay static while forecasting support forces more precisely. Finally, in a future endeavour, a 3D biomechanical software application called OpenSim can entail modelling the human body. OpenSim enables the determination of internal joint torques and muscle activations to expand this study to disabled persons. OpenSim may also display the user of the software the software user might also learn how the body posture changes as the muscles start to relax over time. (Byrd, 2018)

Dissertation Help Services

References

Blake. (2019). *Lifemod. San Clemente: LifeModeler Inc.*

Byrd, R. J. (2018). *"A Trust Region Method Based on Interior PointTechniques for Nonlinear Programming,"* . Mathematical Programming.

Coleman, T. a. (2020). *A Reflective Newton Method for Minimizing a Quadratic Function Subject to Bounds on Some of the Variables* . SIAM Journal on Optimization, Vol. 6,.

D. A. Winter. (2009.). *Biomechanics and Motor Control of Human Movement, Hoboken:* . John Wiley & Sons, Inc.,.

Dan. (2017, 05 19). *assistivedevices*. Retrieved 08 03, 2021, from disabled world: http://www.disabled world.com/assistivedevices/mobility/wheelchairs/.

Dave. (2018, 12 16). *products solutions*. Retrieved 08 03, 2021, from tekscan: https://www.tekscan.com/products solutions/systems/matscan

Ethan. (2017). *Stanford California: National Institute of Health Centre for Biomedical Computation.*

K. Waugh and J. M. Bach. (2019). *"Biomechanics and Its Application to Seating,"* .

Kwak, B. C. (2019). *‘Determination of muscle and joint forces: A new technique to solve the indeterminate problem.’.* Transactions of the American Society of Mechanical Engineering.

Noah. (2018). *U.S. National Library of Medicine.*

Pandy, M. (2018). *'Computer Modeling and Simulation of Human Movement.'* . Ann RevBiomed .

Riley. (2017). *Disabled World.*

Tilley, A. R. (2018). *The Measure of Man and Woman: Human Factors in Design, New York: Whitney Library of Design, .*

Wade. (2019, 08 14). *medlineplus*. Retrieved 08 03, 2021, from nlm.nih: https://www.nlm.nih.gov/medlineplus/ency/article/007071.htm.

Dissertation Help Services

Appendix A: Matlab Code

```
"function vect3d(p0,p1)

% The function plot 3D vector with arrow from point p0 to point p1.
% Example:
%       p0 = [1 2 3];    % Coordinate of the first point p0
%       p1 = [4 5 6];    % Coordinate of the second point p1
%       vect3d(p0,p1)
% YH 2017-10-04

x0 = p0(1) ; y0 = p0(2) ; z0 = p0(3) ; % Read coordinate of the first point
(tail)
x1 = p1(1) ; y1 = p1(2) ; z1 = p1(3) ; % Read coordinate of the second
point (head)
plot3([x0;x1],[y0;y1],[z0;z1],'b-','linewidth',1.5) ;     % Draw a line
between p0 and p1

p = p1 - p0 ;
% p = abs(p1 - p0) ;
alpha = 0.5 ;   % Size of arrow head relative to the length of the vector
```

Dissertation Help Services

```
beta = 0.8 ;     % Width of the base of the arrow head relative to the
length

hu = [x1-alpha*(p(1)+beta*(p(2)+eps)) ; x1 ; x1-alpha*(p(1)-
beta*(p(2)+eps))] ;
hv = [y1-alpha*(p(2)-beta*(p(1)+eps)) ; y1 ; y1-
alpha*(p(2)+beta*(p(1)+eps))] ;
hw = [z1-alpha*p(3) ; z1 ; z1-alpha*p(3)] ;

hold on ;
plot3(hu(:),hv(:),hw(:),'b-','linewidth',1.5) ;  % Plot arrow head
grid on ;

% xlabel('x') ; ylabel('y') ; zlabel('z') ;
% hold off ;

?
function vect3d(p0,p1)

% The function plot 3D vector with arrow from point p0 to point p1.
% Example:
%       p0 = [1 2 3];    % Coordinate of the first point p0
%       p1 = [4 5 6];    % Coordinate of the second point p1
%       vect3d(p0,p1)
% YH 2017-10-04

x0 = p0(1) ; y0 = p0(2) ; z0 = p0(3) ; % Read coordinate of the first point
(tail)
x1 = p1(1) ; y1 = p1(2) ; z1 = p1(3) ; % Read coordinate of the second
point (head)
plot3([x0;x1],[y0;y1],[z0;z1],'r-','linewidth',1.5) ;     % Draw a line
between p0 and p1

p = p1 - p0 ;
% p = abs(p1 - p0) ;
alpha = 0.5 ;    % Size of arrow head relative to the length of the vector
beta = 0.8 ;     % Width of the base of the arrow head relative to the
length

hu = [x1-alpha*(p(1)+beta*(p(2)+eps)) ; x1 ; x1-alpha*(p(1)-
beta*(p(2)+eps))] ;
hv = [y1-alpha*(p(2)-beta*(p(1)+eps)) ; y1 ; y1-
alpha*(p(2)+beta*(p(1)+eps))] ;
hw = [z1-alpha*p(3) ; z1 ; z1-alpha*p(3)] ;

hold on ;
plot3(hu(:),hv(:),hw(:),'r-','linewidth',1.5) ;  % Plot arrow head
grid on ;

% xlabel('x') ; ylabel('y') ; zlabel('z') ;
% hold off ;

%% Read data from mat file containing force plate and camera trajectory
data from Qualysis export
% 2019-07-01 YH modified for MSc project OpenSim preparation

close all; clc; clear;

%% Prepare and load for trc data (camera data - trajectory)
```

```
load s1m1t1.mat ; % 10 s of data
% load s1static.mat ; % 1 s of data
tName = whos ;
eval ( strcat ( ' myData = ' , tName.name , ' ; ' ) ) ; % change the
variable/structure name to 'myData'

start = 0.4 ; % s, start time of display
stop = 0.8 ; % s, stop time of display
gap = 10 ; % no. of smaples to skip in display

nf = myData.Frames ; % total number of frames captured by camera
fs_fp = myData.Force(1).Frequency ; % Hz, sampling rate of force plate 1 =
1000 (should be the same for plate 2 and 3; 3 is the seat)
fs_cam = myData.FrameRate ; % Hz, sampling rate of cameras = 200 Hz
usually, this is the sampling rate for the system
t = 0:1/fs_cam:(myData.Frames/fs_cam-1/fs_cam) ; % s, e.g. 0-10 s sampled
at 200 Hz = 0-2000 data points

mass = 77 ; height = 1870 ; % kg; mm; s1 RNLI

trj = myData.Trajectories.Labeled.Data ; % mm, original trajectories matrix
to be transformed to model coord
nmkr = myData.Trajectories.Labeled.Count ; % total no. of markers
markers = myData.Trajectories.Labeled.Labels ; % marker names

%% Lab coordinate to OpenSim model coordinate transformation - check the
comment out part before use!
% for mk = 1 : nmkr % Coordinate rotation
%     trj(mk,1,:) = myData.Trajectories.Labeled.Data(mk,1,:) ; % model
cooridnate +x from lab coord +x
%     trj(mk,2,:) = myData.Trajectories.Labeled.Data(mk,3,:) ; % model
cooridnate +y from lab coord +z
%     trj(mk,3,:) = -myData.Trajectories.Labeled.Data(mk,2,:) ; % model
cooridnate +z from lab coord -y
% end

% Data(67,4,2000), 67 markers, 4 channels (x,y,z,tol), 2000 data points
(200 Hz)
%     tr() = myData.Trajectories.Labeled.Data(1,1,:) ;
% Index for each marker

iLFHD = find(contains(markers,'LFHD')); % Head 4
iRFHD = find(contains(markers,'RFHD'));
iLBHD = find(contains(markers,'LBHD'));
iRBHD = find(contains(markers,'RBHD'));

iLACR = find(contains(markers,'LACR')); % Trunk 11~
iRACR = find(contains(markers,'RACR'));
iCLAV = find(contains(markers,'CLAV'));
iRBAC = find(contains(markers,'RBAC'));
iSTER = find(contains(markers,'STER'));
iC7 = find(contains(markers,'C7'));
iT10 = find(contains(markers,'T10'));
iLASIS = find(contains(markers,'LASIS'));
iRASIS = find(contains(markers,'RASIS'));
iLPSIS = find(contains(markers,'LPSIS'));
iRPSIS = find(contains(markers,'RPSIS'));
```

```
iLUPA = find(contains(markers,'LUPA')); % Arms 22~
iRUPA = find(contains(markers,'RUPA'));
iLUPP = find(contains(markers,'LUPP'));
iRUPP = find(contains(markers,'RUPP'));
iLUD = find(contains(markers,'LUD'));
iRUD = find(contains(markers,'RUD'));
iLLELB = find(contains(markers,'LLELB'));
iRLELB = find(contains(markers,'RLELB'));
iLMELB = find(contains(markers,'LMELB'));
iRMELB = find(contains(markers,'RMELB'));
iLFPA = find(contains(markers,'LFPA'));
iRFPA = find(contains(markers,'RFPA'));
iLFPP = find(contains(markers,'LFPP'));
iRFPP = find(contains(markers,'RFPP'));
iLFD = find(contains(markers,'LFD'));
iRFD = find(contains(markers,'RFD'));
iLWRA = find(contains(markers,'LWRA'));
iRWRA = find(contains(markers,'RWRA'));
iLWRB = find(contains(markers,'LWRB'));
iRWRB = find(contains(markers,'RWRB'));
iLMCP3 = find(contains(markers,'LMCP3'));
iRMCP3 = find(contains(markers,'RMCP3'));

iLMCP2 = find(contains(markers,'LMCP2'));
iRMCP2 = find(contains(markers,'RMCP2'));
iLMCP5 = find(contains(markers,'LMCP5'));
iRMCP5 = find(contains(markers,'RMCP5'));

iLTHIPA = find(contains(markers,'LTHIPA')); % Legs 30
iRTHIPA = find(contains(markers,'RTHIPA'));
iLTHIPP = find(contains(markers,'LTHIPP'));
iRTHIPP = find(contains(markers,'RTHIPP'));
iLTHIDA = find(contains(markers,'LTHIDA'));
iRTHIDA = find(contains(markers,'RTHIDA'));
iLTHIDP = find(contains(markers,'LTHIDP'));
iRTHIDP = find(contains(markers,'RTHIDP'));
iLLKN = find(contains(markers,'LLKN'));
iRLKN = find(contains(markers,'RLKN'));
iLMKN = find(contains(markers,'LMKN'));
iRMKN = find(contains(markers,'RMKN'));
iLSHAPA = find(contains(markers,'LSHAPA'));
iRSHAPA = find(contains(markers,'RSHAPA'));
iLSHAPP = find(contains(markers,'LSHAPP'));
iRSHAPP = find(contains(markers,'RSHAPP'));
iLSHADA = find(contains(markers,'LSHADA'));
iRSHADA = find(contains(markers,'RSHADA'));
iLSHADP = find(contains(markers,'LSHADP'));
iRSHADP = find(contains(markers,'RSHADP'));
iLLANK = find(contains(markers,'LLANK'));
iRLANK = find(contains(markers,'RLANK'));
iLMANK = find(contains(markers,'LMANK'));
iRMANK = find(contains(markers,'RMANK'));
iLHE = find(contains(markers,'LHE'));
iRHE = find(contains(markers,'RHE'));
iLMTP1 = find(contains(markers,'LMTP1'));
iRMTP1 = find(contains(markers,'RMTP1'));
iLMTP5 = find(contains(markers,'LMTP5'));
iRMTP5 = find(contains(markers,'RMTP5'));

for frames = 1:nf % Coordinate origin translation to between two feet
    om = [ (trj(iLMANK,1,frames)+trj(iRMANK,1,frames))/2 ,...
```

```
        min(trj(iLMTP1,2,frames),trj(iRMTP1,2,frames)) ,...
        (trj(iLMANK,3,frames)+trj(iRMANK,3,frames))/2 ] ; % origin of model
coordinate
end

%% Kinematic (trajectory) data processing
% Compute angular parameters of trunk
theta_ASIS = permute( atan2d((trj(iLASIS,3,:)-trj(iRASIS,3,:)) ,
(trj(iLASIS,1,:)-trj(iRASIS,1,:))) , [3 2 1] ) ; % deg, ASIS line angle in
the x-z plane
theta_CS = permute( atan2d((trj(iCLAV,3,:)-trj(iSTER,3,:)) ,
(trj(iCLAV,1,:)-trj(iSTER,1,:))) , [3 2 1] ) ; % deg, CLAV-STER line angle
in the x-z plane
theta_FT = abs(theta_CS - theta_ASIS) ; % deg, front trunk roll angle in
the frontal plane

theta_PSIS = permute( atan2d((trj(iLPSIS,3,:)-trj(iRPSIS,3,:)) ,
(trj(iLPSIS,1,:)-trj(iRPSIS,1,:))) , [3 2 1] ) ; % deg, PSIS line angle in
the x-z plane
theta_C7T10 = permute( atan2d((trj(iC7,3,:)-trj(iT10,3,:)) , (trj(iC7,1,:)-
trj(iT10,1,:))) , [3 2 1] ) ; % deg, C7-T10 line angle in the x-z plane
theta_BT = abs(theta_C7T10 - theta_PSIS) ; % deg, back trunk roll angle in
the frontal plane

% Compute velocity, acceleration from displacement

dis_z_LASIS = permute( trj(iLASIS,3,:) , [3 2 1] )./1000 ; % displacement
from mm to m
dis_z_RASIS = permute( trj(iRASIS,3,:) , [3 2 1] )./1000 ;
dis_z_LPSIS = permute( trj(iLPSIS,3,:) , [3 2 1] )./1000 ;
dis_z_RPSIS = permute( trj(iRPSIS,3,:) , [3 2 1] )./1000 ;

lpf = 15 ; % Hz, Lowpass filter
[b,a]=butter(4,lpf./(fs_cam./2),'low');

vel_z_LASIS = gradient( dis_z_LASIS , 1/fs_cam ) ;
velf_z_LASIS = filtfilt(b,a,vel_z_LASIS); % with filtfilt() 4-pole is 8-
pole
acc_z_LASIS = gradient( velf_z_LASIS , 1/fs_cam ) ; % peak acc from force
estimates = 17730/(654.5/9.81) = 26.53 m/s2

vel_z_RASIS = gradient( dis_z_RASIS , 1/fs_cam ) ;
velf_z_RASIS = filtfilt(b,a,vel_z_RASIS); % with filtfilt() 4-pole is 8-
pole
acc_z_RASIS = gradient( velf_z_RASIS , 1/fs_cam ) ;

vel_z_LPSIS = gradient( dis_z_LPSIS , 1/fs_cam ) ;
velf_z_LPSIS = filtfilt(b,a,vel_z_LPSIS); % with filtfilt() 4-pole is 8-
pole
acc_z_LPSIS = gradient( velf_z_LPSIS , 1/fs_cam ) ;

vel_z_RPSIS = gradient( dis_z_RPSIS , 1/fs_cam ) ;
velf_z_RPSIS = filtfilt(b,a,vel_z_RPSIS); % with filtfilt() 4-pole is 8-
pole
acc_z_RPSIS = gradient( velf_z_RPSIS , 1/fs_cam ) ;

figure (15)
subplot(3,1,1)
```

```
plot(t,dis_z_LASIS,'k-',t,dis_z_RASIS,'k-.',t,dis_z_LPSIS,'b--
',t,dis_z_RPSIS,'r:');
ylabel('Displacement ( m )'); xlabel('Time ( s )');
subplot(3,1,2)
plot(t,velf_z_LASIS,'k-',t,velf_z_RASIS,'k-.',t,velf_z_LPSIS,'b--
',t,velf_z_RPSIS,'r:');
ylabel('Velocity ( m / s )'); xlabel('Time ( s )');
subplot(3,1,3)
plot(t,acc_z_LASIS,'k-',t,acc_z_RASIS,'k-.',t,acc_z_LPSIS,'b--
',t,acc_z_RPSIS,'r:');
ylabel('Acceleration ( m / s^2 )'); xlabel('Time ( s )');
legend('z-LASIS','z-RASIS','z-LPSIS','z-RPSIS');

%% Force plate (FP) data - 3 force plates

f1 = myData.Force(1).Force()' ; % N, force plate data: 0-10000 data points
at 1000 Hz = 0-10 s
f2 = myData.Force(2).Force()' ;
f3 = myData.Force(3).Force()' ;

f1 = downsample(f1,fs_fp/fs_cam) ; % N, force down-sampled to match
trajectory at 200 Hz
f1 = fillnan(f1) ; % fillnan() to fill all nan in each column
f2 = downsample(f2,fs_fp/fs_cam) ; f2 = fillnan(f2) ;
f3 = downsample(f3,fs_fp/fs_cam) ; f3 = fillnan(f3) ;

m1 = myData.Force(1).Moment()' ; % Nm, moment
m2 = myData.Force(2).Moment()' ;
m3 = myData.Force(3).Moment()' ;

m1 = downsample(m1,fs_fp/fs_cam) ; % Nm, moment down-sampled to match
trajectory at 200 Hz
m1 = fillnan(m1) ;
m2 = downsample(m2,fs_fp/fs_cam) ; m2 = fillnan(m2) ;
m3 = downsample(m3,fs_fp/fs_cam) ; m3 = fillnan(m3) ;

cop1 = myData.Force(1).COP()' ; % mm, centre of pressure
cop2 = myData.Force(2).COP()' ;
cop3 = myData.Force(3).COP()' ;

cop1 = downsample(cop1,fs_fp/fs_cam) ; % mm, cop down-sampled to match
trajectory at 200 Hz
cop1 = fillnan(cop1) ;
cop2 = downsample(cop2,fs_fp/fs_cam) ; cop2 = fillnan(cop2) ;
cop3 = downsample(cop3,fs_fp/fs_cam) ; cop3 = fillnan(cop3) ;

%% FP Step 1 check pose of all 3 force plates (FP): if needed inverse sign
of X and Y axes FP data - Force, Moment, CoP
% r_a = [-1 0 0 ; 0 -1 0 ; 0 0 1] ; % rotation matrix to rectify Qualysis
% measurement from FP (flip signs of x and y) 2018-07-18
% r_a = [1 0 0 ; 0 1 0 ; 0 0 1] ; % no rotation as 2018-07-01
r_a = [-1 0 0 ; 0 1 0 ; 0 0 1] ; % flip x-axis only for force and moment
but not COP:

f1r = (r_a * f1')' ; % transformation using inner product with rotation
matrix
m1r = (r_a * m1')' ;
cop1r = cop1 ;
% cop1r = (r_a * cop1')' ;
```

```
f2r = (r_a * f2')' ; % transformation using inner product with rotation
matrix
m2r = (r_a * m2')' ;
cop2r = cop2 ;
% cop2r = (r_a * cop2')' ;

f3r = (r_a * f3')' ; % transformation using inner product with rotation
matrix
m3r = (r_a * m3')' ;
cop3r = cop3 ;
% cop3r = (1 * cop3')' ;

%% FP Step 2 rectify FP3 beneath rigid seat due to seat height

az0 = 0.58 ; % m, thickness of padding above sensor z- this is seat height
My = m3r(:,2) ; Mx = m3r(:,1) ;
Fx = f3r(:,1) ; Fy = f3r(:,2) ; Fz = f3r(:,3) ;

% COPx = -(Myr/Fz)
Myr = ( My - Fx * az0 ) ; % Nm, thickness rectified moment y-
COPx = - ( Myr ./ Fz )*1000 ; % mm, thickness rectified COP x-

% COPy = (Mxr/Fz)
Mxr = ( Mx + Fy * az0 ) ; % thickness rectified moment x-
COPy = ( Mxr ./ Fz ) * 1000 ; % mm, thickness rectified COP y-

m3r(:,1) = Mxr ; % Nm, rectified with thickness
m3r(:,2) = Myr ; % Nm
cop3r = [COPx COPy zeros(length(m3r),1)] ; % mm

%% FP Step 3 rotation from lab coordinates to OpenSim model coordinates
% After two steps: f1r, f2r, f3r, m1r, m2r, m3r, cop1r, cop2r, cop3r
(number indicates FP)
r_b = [1 0 0 ; 0 0 -1 ; 0 1 0] ; % rotation to OpenSim model coordinates
2018-07-06
% f1r = (r_b * f1r')' ; f2r = (r_b * f2r')' ; f3r = (r_b * f3r')' ;
% m1r = (r_b * m1r')' ; m2r = (r_b * m2r')' ; m3r = (r_b * m3r')' ;
% cop1r = (r_b * cop1r')' ; cop2r = (r_b * cop2r')' ; cop3r = (r_b *
cop3r')' ;

sw = mean(f3r(1:fs_cam*1,3)) ; % sitting weight on FP3 = average of first 1
s of z-axis force
% f3r(:,3) = f3r(:,3) - sw ; % remove static sitting weight

%% FP Rectification display

figure(11) % FP1
subplot(2,3,1);plot(t,f1(:,1),'r--',t,f1(:,2),'b-',t,f1(:,3),'k-');
% subplot(2,3,1),plot(t(1:end-3),f1(1:end-2,1),'r--',t(1:end-3),f1(1:end-
2,2),'b-',t(1:end-3),f1(1:end-2,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )'); title('Force VS Time (FP1) ')
legend({'f_x','f_y','f_z'}); box on; grid on;
subplot(2,3,4);plot(t,f1r(:,1),'r--',t,f1r(:,2),'b-',t,f1r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )');
legend({'f_x (rec)','f_y (rec)','f_z'}); box on; grid on;

subplot(2,3,2);plot(t,m1(:,1),'r--',t,m1(:,2),'b-',t,m1(:,3),'k-');
```

```
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('Moment VS Time
(FP1)')
legend({'m_x','m_y','m_z'});  box on; grid on;
subplot(2,3,5);plot(t,m1r(:,1),'r--',t,m1r(:,2),'b-',t,m1r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )');
legend({'m_x (rec)','m_y (rec)','m_z'}); box on; grid on;

subplot(2,3,3);plot(t,cop1(:,1),'r--',t,cop1(:,2),'b-',t,cop1(:,3),'k-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('CoP VS Time (FP1)')
legend({'cop_x','cop_y','cop_z'});  box on; grid on;
subplot(2,3,6);plot(t,cop1r(:,1),'r--',t,cop1r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )');
legend({'cop_x (rec)','cop_y (rec)'}); box on; grid on;

figure(12) % FP2
subplot(2,3,1);plot(t,f2(:,1),'r--',t,f2(:,2),'b-',t,f2(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )'); title('Force VS Time (FP2)  ')
legend({'f_x','f_y','f_z'}); box on; grid on;
subplot(2,3,4);plot(t,f2r(:,1),'r--',t,f2r(:,2),'b-',t,f2r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )');
legend({'f_x (rec)','f_y (rec)','f_z'}); box on; grid on;

subplot(2,3,2);plot(t,m2(:,1),'r--',t,m2(:,2),'b-',t,m2(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('Moment VS Time
(FP2)')
legend({'m_x','m_y','m_z'});  box on; grid on;
subplot(2,3,5);plot(t,m2r(:,1),'r--',t,m2r(:,2),'b-',t,m2r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )');
legend({'m_x (rec)','m_y (rec)','m_z'}); box on; grid on;

subplot(2,3,3);plot(t,cop2(:,1),'r--',t,cop2(:,2),'b-',t,cop2(:,3),'k-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('CoP VS Time (FP2)')
legend({'cop_x','cop_y','cop_z'});  box on; grid on;
subplot(2,3,6);plot(t,cop2r(:,1),'r--',t,cop2r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )');
legend({'cop_x (rec)','cop_y (rec)'}); box on; grid on;

figure(13) % FP3
subplot(2,3,1);plot(t,f3(:,1),'r--',t,f3(:,2),'b-',t,f3(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )'); title('Force VS Time (FP3)  ')
legend({'f_x','f_y','f_z'}); box on; grid on;
subplot(2,3,4);plot(t,f3r(:,1),'r--',t,f3r(:,2),'b-',t,f3r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )');
legend({'f_x (rec)','f_y (rec)','f_z'},'location','best'); box on; grid on;

subplot(2,3,2);plot(t,m3(:,1),'r--',t,m3(:,2),'b-',t,m3(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('Moment VS Time
(FP3)')
legend({'m_x','m_y','m_z'});  box on; grid on;
subplot(2,3,5);plot(t,m3r(:,1),'r--',t,m3r(:,2),'b-',t,m3r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )');
legend({'m_x (rec+thick)','m_y (rec+thick)','m_z'},'location','best'); box
on; grid on;

subplot(2,3,3);plot(t,cop3(:,1),'r--',t,cop3(:,2),'b-',t,cop3(:,3),'k-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('CoP VS Time (FP3)')
legend({'cop_x','cop_y','cop_z'});  box on; grid on;
subplot(2,3,6);plot(t,cop3r(:,1),'r--',t,cop3r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )');
```

```
legend({'cop_x (rec+thick)','cop_y (rec+thick)'},'location','best'); box
on; grid on;

%% Three FPs - (1) Left foot; (2) Right foot; (3) Seat pan

figure ( 21 )
set(gcf,'position',[10 5 700 800]);

subplot(3,3,1);plot(t,f1r(:,1),'r--',t,f1r(:,2),'b-',t,f1r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )'); title('FP1');
legend({'f_x (rec)','f_y (rec)','f_z'}); box on; grid on;
subplot(3,3,2);plot(t,m1r(:,1),'r--',t,m1r(:,2),'b-',t,m1r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('FP1');
legend({'m_x (rec)','m_y (rec)','m_z'}); box on; grid on;
subplot(3,3,3),plot(t,cop1r(:,1),'r--',t,cop1r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('FP1');
legend({'cop_x (rec)','cop_y (rec)'}); box on; grid on;

subplot(3,3,4);plot(t,f2r(:,1),'r--',t,f2r(:,2),'b-',t,f2r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )'); title('FP2');
legend({'f_x (rec)','f_y (rec)','f_z'}); box on; grid on;
subplot(3,3,5);plot(t,m2r(:,1),'r--',t,m2r(:,2),'b-',t,m2r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('FP2');
legend({'m_x (rec)','m_y (rec)','m_z'}); box on; grid on;
subplot(3,3,6);plot(t,cop2r(:,1),'r--',t,cop2r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('FP2');
legend({'cop_x (rec)','cop_y (rec)'}); box on; grid on;

subplot(3,3,7);plot(t,f3r(:,1),'r--',t,f3r(:,2),'b-',t,f3r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )'); title('FP3');
legend({'f_x (rec)','f_y (rec)','f_z'},'location','best'); box on; grid on;
subplot(3,3,8);plot(t,m3r(:,1),'r--',t,m3r(:,2),'b-',t,m3r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('FP3');
legend({'m_x (rec+thick)','m_y (rec+thick)','m_z'},'location','best'); box
on; grid on;
subplot(3,3,9);plot(t,cop3r(:,1),'r--',t,cop3r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('FP3');
legend({'cop_x (rec+thick)','cop_y (rec+thick)'},'location','best'); box
on; grid on;

%% Display 3D force (x-y-z) and COP (x-y)
% p0 = COP coordinate x-y-z(0s) in mm, p1 = force x-y-z in N but same scale
as mm COP

fp1 = (1 * myData.Force(1).ForcePlateLocation')' ; % Force-plate 1 rotation
to model coord : 1 = no rotation, r_b = model coord rotation
fp1x = fp1(:,1) ; fp1y = fp1(:,2) ; fp1z = fp1(:,3) ;
fp2 = (1 * myData.Force(2).ForcePlateLocation')' ; % Force-plate 2 rotation
to model coord
fp2x = fp2(:,1) ; fp2y = fp2(:,2) ; fp2z = fp2(:,3) ;
fp3 = (1 * myData.Force(3).ForcePlateLocation')' ; % Force-plate 2 rotation
to model coord
fp3x = fp3(:,1) ; fp3y = fp3(:,2) ; fp3z = fp3(:,3) ;

for n = start/(1/fs_cam):gap:stop/(1/fs_cam) % frame range to be plotted
Figure 31

figure ( 31 )
set(gcf,'position',[750 5 550 800]); % position of figure window on screen
```

```
% fp3_p0 = [cop3r(n,1)+mean(fp3x) cop3r(n,2)+mean(fp3y)
cop3r(n,3)+mean(fp3z)] ; % Coordinate (x y z) of the first point p0
fp3_p0 = [cop3r(n,1)+mean(fp3x) cop3r(n,2)+mean(fp3y)
cop3r(n,3)+mean(fp3z)+1000*az0] ; % Coordinate (x y z) of the first point
p0 raised az0 for FP3
fp3_p1 = fp3_p0 + f3r(n,:) ;     % Coordinate (x y z) of the second point p1
vect3d(fp3_p0,fp3_p1) ; hold on ;

fp2_p0 = [cop2r(n,1)+mean(fp2x) cop2r(n,2)+mean(fp2y)
cop2r(n,3)+mean(fp2z)] ; % Coordinate (x y z) of the first point p0
fp2_p1 = fp2_p0 + f2r(n,:) ;     % Coordinate (x y z) of the second point p1
vect3d(fp2_p0,fp2_p1) ; hold on ;

fp1_p0 = [cop1r(n,1)+mean(fp1x) cop1r(n,2)+mean(fp1y)
cop1r(n,3)+mean(fp1z)] ; % Coordinate (x y z) of the first point p0
fp1_p1 = fp1_p0 + f1r(n,:) ;     % Coordinate (x y z) of the second point p1
vect3d(fp1_p0,fp1_p1) ; hold on ;

% Trajectory data:
% myData.Trajectories.Labeled.Data(1,1,:) ;
% Data(67,4,2000), 67 markers, 4 channels (x,y,z,tol), 2000 data points
(200 Hz)

% Find upper body centroid:
upper_x = mean([trj(iLFHD,1,n) trj(iRFHD,1,n)  ... % Head
                trj(iLBHD,1,n) trj(iRBHD,1,n)  ...
                trj(iLACR,1,n) trj(iRACR,1,n)  ... % Trunk
                trj(iRBAC,1,n) trj(iSTER,1,n)  ...
                trj(iC7,1,n) trj(iT10,1,n)  ...
                trj(iLPSIS,1,n) trj(iRPSIS,1,n)  ...
                trj(iLASIS,1,n) trj(iRASIS,1,n)  ...
                trj(iLUPA,1,n) trj(iRUPA,1,n)  ... % Arms
                trj(iLUPP,1,n) trj(iRUPP,1,n)  ...
                trj(iLUD,1,n) trj(iRUD,1,n)  ...
                trj(iLLELB,1,n) trj(iRLELB,1,n)  ...
                trj(iLMELB,1,n) trj(iRMELB,1,n)  ...
                trj(iLFPA,1,n) trj(iRFPA,1,n)  ...
                trj(iLFPP,1,n) trj(iRFPP,1,n)  ...
                trj(iLFD,1,n) trj(iRFD,1,n)  ...
                trj(iLWRA,1,n) trj(iRWRA,1,n)  ...
                trj(iLWRB,1,n) trj(iRWRB,1,n)  ...
                trj(iLMCP3,1,n) trj(iRMCP3,1,n) ]) ;

upper_y = mean([trj(iLFHD,2,n) trj(iRFHD,2,n)  ... % Head
                trj(iLBHD,2,n) trj(iRBHD,2,n)  ...
                trj(iLACR,2,n) trj(iRACR,2,n)  ... % Trunk
                trj(iRBAC,2,n) trj(iSTER,2,n)  ...
                trj(iC7,2,n) trj(iT10,2,n)  ...
                trj(iLPSIS,2,n) trj(iRPSIS,2,n)  ...
                trj(iLASIS,2,n) trj(iRASIS,2,n)  ...
                trj(iLUPA,2,n) trj(iRUPA,2,n)  ... % Arms
                trj(iLUPP,2,n) trj(iRUPP,2,n)  ...
                trj(iLUD,2,n) trj(iRUD,2,n)  ...
                trj(iLLELB,2,n) trj(iRLELB,2,n)  ...
                trj(iLMELB,2,n) trj(iRMELB,2,n)  ...
                trj(iLFPA,2,n) trj(iRFPA,2,n)  ...
                trj(iLFPP,2,n) trj(iRFPP,2,n)  ...
                trj(iLFD,2,n) trj(iRFD,2,n)  ...
                trj(iLWRA,2,n) trj(iRWRA,2,n)  ...
                trj(iLWRB,2,n) trj(iRWRB,2,n) ...
                trj(iLMCP3,2,n) trj(iRMCP3,2,n) ]) ;

upper_z = mean([trj(iLFHD,3,n) trj(iRFHD,3,n) ... % Head
                trj(iLBHD,3,n) trj(iRBHD,3,n) ...
                trj(iLACR,3,n) trj(iRACR,3,n) ... % Trunk
                trj(iRBAC,3,n) trj(iSTER,3,n) ...
                trj(iC7,3,n) trj(iT10,3,n) ...
                trj(iLPSIS,3,n) trj(iRPSIS,3,n) ...
                trj(iLASIS,3,n) trj(iRASIS,3,n) ...
                trj(iLUPA,3,n) trj(iRUPA,3,n) ... % Arms
                trj(iLUPP,3,n) trj(iRUPP,3,n) ...
                trj(iLUD,3,n) trj(iRUD,3,n) ...
                trj(iLLELB,3,n) trj(iRLELB,3,n) ...
                trj(iLMELB,3,n) trj(iRMELB,3,n) ...
                trj(iLFPA,3,n) trj(iRFPA,3,n) ...
                trj(iLFPP,3,n) trj(iRFPP,3,n) ...
                trj(iLFD,3,n) trj(iRFD,3,n) ...
                trj(iLWRA,3,n) trj(iRWRA,3,n) ...
                trj(iLWRB,3,n) trj(iRWRB,3,n) ...
                trj(iLMCP3,3,n) trj(iRMCP3,3,n) ]) ;

for mk = 1:nmkr % s1static.mat only contains 66 markers
    scatter3(trj(mk,1,n),trj(mk,2,n),trj(mk,3,n),100,...
        'MarkerEdgeColor','k','MarkerFaceColor','g') ; hold on

scatter3(upper_x,upper_y,upper_z,150,'MarkerEdgeColor','k','MarkerFaceColor
','r') ; hold on
end

ax = gca; ax.Projection = 'perspective'; % Foreshortening to perceive depth
in 2D representations of 3D objects
% 'orthographic' as defualt to maintain correct relative dimensions of
graphic objects
view(0,10); %
axis([-50 1000 -50 1500 -50 1500]) ; % frontal view
ax = gca; ax.XTick = [ 0 500 1000 1500]; ax.YTick = [0 500 1000 1500];
ax.ZTick = [0 500 1000 1500];
title(['start - stop / total time = ' num2str(start) ' - ' num2str(stop) '
/ ' num2str(t(end)) '  at ' num2str(fs_cam) ' Hz' ]) ;

fill3( fp1x , fp1y , fp1z , [0.85 0.85 0.85] ) ; hold on ; % FP1
text((fp1x(2)+fp1x(1))/2+100,(fp1y(4)+fp1y(1))/2-250,10,'FP1','Color',[0 0
1]) ; % FP1
line([(fp1x(1)+fp1x(2))/2 (fp1x(3)+fp1x(4))/2] , [(fp1y(1)+fp1y(2))/2
(fp1y(3)+fp1y(4))/2] ,...
    [(fp1z(1)+fp1z(2))/2 (fp1z(3)+fp1z(4))/2] , 'color' , [0 0 0] ,
'linewidth' , 1 ) ; hold on ;
line([(fp1x(1)+fp1x(4))/2 (fp1x(2)+fp1x(3))/2] , [(fp1y(1)+fp1y(4))/2
(fp1y(2)+fp1y(3))/2] ,...
    [(fp1z(1)+fp1z(4))/2 (fp1z(2)+fp1z(3))/2] , 'color' , [0 0 0] ,
'linewidth' , 1 ) ; hold on ;

fill3( fp2x , fp2y , fp2z , [0.85 0.85 0.85] ) ; hold on ; % FP2
text((fp2x(2)+fp2x(1))/2+100,(fp2y(4)+fp2y(1))/2-250,10,'FP2','Color',[0 0
1]) ; % FP2
line([(fp2x(1)+fp2x(2))/2 (fp2x(3)+fp2x(4))/2] , [(fp2y(1)+fp2y(2))/2
(fp2y(3)+fp2y(4))/2] ,...
    [(fp2z(1)+fp2z(2))/2 (fp2z(3)+fp2z(4))/2] , 'color' , [0 0 0] ,
'linewidth' , 1 ) ; hold on ;
```

```
line([(fp2x(1)+fp2x(4))/2 (fp2x(2)+fp2x(3))/2] , [(fp2y(1)+fp2y(4))/2
(fp2y(2)+fp2y(3))/2] ,...
    [(fp2z(1)+fp2z(4))/2 (fp2z(2)+fp2z(3))/2] , 'color' , [0 0 0] ,
'linewidth' , 1 ) ; hold on ;

% FP3 raised by az0
fill3( fp3x , fp3y , fp3z+1000*az0 , [0.85 0.85 0.85] ) ; hold on ; % FP3
text((fp3x(2)+fp3x(1))/2+50,(fp3y(4)+fp3y(1))/2-
250,30+1000*az0,'rFP3','Color',[0 0 1]) ; % FP3
line([(fp3x(1)+fp3x(2))/2 (fp3x(3)+fp3x(4))/2] , [(fp3y(1)+fp3y(2))/2
(fp3y(3)+fp3y(4))/2] ,...
    [(fp3z(1)+fp3z(2))/2+1000*az0 (fp3z(3)+fp3z(4))/2+1000*az0] , 'color' ,
[0 0 0] , 'linewidth' , 1 ) ; hold on ;
line([(fp3x(1)+fp3x(4))/2 (fp3x(2)+fp3x(3))/2] , [(fp3y(1)+fp3y(4))/2
(fp3y(2)+fp3y(3))/2] ,...
    [(fp3z(1)+fp3z(4))/2+1000*az0 (fp3z(2)+fp3z(3))/2+1000*az0] , 'color' ,
[0 0 0] , 'linewidth' , 1 ) ; hold on ;

% Upper body centroid vertical projection
line([upper_x upper_x] , [upper_y upper_y] , [0 upper_z] , 'color' , [1 0
0] , 'linewidth' , 1 , 'linestyle' , '--') ;  hold on ;
w_up = mass*0.75*9.81 ;                    % N, upper body weight = 75% of
body weight
upc_p0 = [upper_x upper_y upper_z] ;       % Coordinate (x y z) of the
first point p0
upc_p1 = [upper_x upper_y upper_z-w_up] ;  % Coordinate (x y z) of the
second point p1
vect3d_red(upc_p0,upc_p1) ; hold on ;

% Head
fill3( [trj(iLBHD,1,n) trj(iLFHD,1,n) trj(iRFHD,1,n) trj(iRBHD,1,n)],...  %
head x
        [trj(iLBHD,2,n) trj(iLFHD,2,n) trj(iRFHD,2,n) trj(iRBHD,2,n)],... %
y
        [trj(iLBHD,3,n) trj(iLFHD,3,n) trj(iRFHD,3,n) trj(iRBHD,3,n)],... %
z
       [0.5 0.5 0.5] ) ; hold on ;

% Trunk
line( [trj(iLACR,1,n) trj(iCLAV,1,n) trj(iSTER,1,n)],...
      [trj(iLACR,2,n) trj(iCLAV,2,n) trj(iSTER,2,n)],...
      [trj(iLACR,3,n) trj(iCLAV,3,n) trj(iSTER,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ; hold on ;
line( [trj(iCLAV,1,n) trj(iRACR,1,n) trj(iRBAC,1,n)],...
      [trj(iCLAV,2,n) trj(iRACR,2,n) trj(iRBAC,2,n)],...
      [trj(iCLAV,3,n) trj(iRACR,3,n) trj(iRBAC,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ; hold on ;
line( [trj(iLACR,1,n) trj(iC7,1,n) trj(iRACR,1,n)],...
      [trj(iLACR,2,n) trj(iC7,2,n) trj(iRACR,2,n)],...
      [trj(iLACR,3,n) trj(iC7,3,n) trj(iRACR,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ; hold on ;
fill3( [trj(iLPSIS,1,n) trj(iLASIS,1,n) trj(iRASIS,1,n)
trj(iRPSIS,1,n)],...
       [trj(iLPSIS,2,n) trj(iLASIS,2,n) trj(iRASIS,2,n)
trj(iRPSIS,2,n)],...
       [trj(iLPSIS,3,n) trj(iLASIS,3,n) trj(iRASIS,3,n) trj(iRPSIS,3,n)],
[0.5 0.5 0.5] ) ; hold on ;

line([(trj(iLPSIS,1,n)+trj(iRPSIS,1,n))/2 trj(iT10,1,n)],...
     [(trj(iLPSIS,2,n)+trj(iRPSIS,2,n))/2 trj(iT10,2,n)],...
     [(trj(iLPSIS,3,n)+trj(iRPSIS,3,n))/2 trj(iT10,3,n)], 'color' , [0 0 0]
, 'linewidth' , 2 ) ;  hold on ;
line( [trj(iT10,1,n) trj(iSTER,1,n)],...
      [trj(iT10,2,n) trj(iSTER,2,n)],...
      [trj(iT10,3,n) trj(iSTER,3,n)], 'color' , [0 0 0] , 'linewidth' , 2 )
; hold on ;

% Arm left
fill3( [trj(iLUPA,1,n) trj(iLUPP,1,n) trj(iLUD,1,n)],...
       [trj(iLUPA,2,n) trj(iLUPP,2,n) trj(iLUD,2,n)],...
       [trj(iLUPA,3,n) trj(iLUPP,3,n) trj(iLUD,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iLFPA,1,n) trj(iLFPP,1,n) trj(iLFD,1,n)],...
       [trj(iLFPA,2,n) trj(iLFPP,2,n) trj(iLFD,2,n)],...
       [trj(iLFPA,3,n) trj(iLFPP,3,n) trj(iLFD,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iLWRA,1,n) trj(iLWRB,1,n) trj(iLMCP3,1,n)],...
       [trj(iLWRA,2,n) trj(iLWRB,2,n) trj(iLMCP3,2,n)],...
       [trj(iLWRA,3,n) trj(iLWRB,3,n) trj(iLMCP3,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
line([trj(iLMELB,1,n) trj(iLLELB,1,n) trj(iLACR,1,n)],...
     [trj(iLMELB,2,n) trj(iLLELB,2,n) trj(iLACR,2,n)],...
     [trj(iLMELB,3,n) trj(iLLELB,3,n) trj(iLACR,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ;  hold on ;
line([trj(iLWRB,1,n) trj(iLLELB,1,n)],...
     [trj(iLWRB,2,n) trj(iLLELB,2,n)],...
     [trj(iLWRB,3,n) trj(iLLELB,3,n)], 'color' , [0 0 0] , 'linewidth' , 2
) ;  hold on ;

% Arm right
fill3( [trj(iRUPA,1,n) trj(iRUPP,1,n) trj(iRUD,1,n)],...
       [trj(iRUPA,2,n) trj(iRUPP,2,n) trj(iRUD,2,n)],...
       [trj(iRUPA,3,n) trj(iRUPP,3,n) trj(iRUD,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iRFPA,1,n) trj(iRFPP,1,n) trj(iRFD,1,n)],...
       [trj(iRFPA,2,n) trj(iRFPP,2,n) trj(iRFD,2,n)],...
       [trj(iRFPA,3,n) trj(iRFPP,3,n) trj(iRFD,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iRWRA,1,n) trj(iRWRB,1,n) trj(iRMCP3,1,n)],...
       [trj(iRWRA,2,n) trj(iRWRB,2,n) trj(iRMCP3,2,n)],...
       [trj(iRWRA,3,n) trj(iRWRB,3,n) trj(iRMCP3,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
line([trj(iRMELB,1,n) trj(iRLELB,1,n) trj(iRACR,1,n)],...
     [trj(iRMELB,2,n) trj(iRLELB,2,n) trj(iRACR,2,n)],...
     [trj(iRMELB,3,n) trj(iRLELB,3,n) trj(iRACR,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ;  hold on ;
line([trj(iRWRB,1,n) trj(iRLELB,1,n)],...
     [trj(iRWRB,2,n) trj(iRLELB,2,n)],...
     [trj(iRWRB,3,n) trj(iRLELB,3,n)], 'color' , [0 0 0] , 'linewidth' , 2
) ;   hold on ;

% Leg left
fill3( [trj(iLTHIPA,1,n) trj(iLTHIPP,1,n) trj(iLTHIDP,1,n)
trj(iLTHIDA,1,n)],...
       [trj(iLTHIPA,2,n) trj(iLTHIPP,2,n) trj(iLTHIDP,2,n)
trj(iLTHIDA,2,n)],...
       [trj(iLTHIPA,3,n) trj(iLTHIPP,3,n) trj(iLTHIDP,3,n)
trj(iLTHIDA,3,n)], [0.5 0.5 0.5] ) ; hold on ;
fill3( [trj(iLSHAPA,1,n) trj(iLSHAPP,1,n) trj(iLSHADP,1,n)
trj(iLSHADA,1,n)],...
```

```
        [trj(iLSHAPA,2,n) trj(iLSHAPP,2,n) trj(iLSHADP,2,n)
trj(iLSHADA,2,n)],...
        [trj(iLSHAPA,3,n) trj(iLSHAPP,3,n) trj(iLSHADP,3,n)
trj(iLSHADA,3,n)], [0.5 0.5 0.5] ) ; hold on ;
fill3( [trj(iLHE,1,n) trj(iLMTP1,1,n) trj(iLMTP5,1,n)],...
        [trj(iLHE,2,n) trj(iLMTP1,2,n) trj(iLMTP5,2,n)],...
        [trj(iLHE,3,n) trj(iLMTP1,3,n) trj(iLMTP5,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iLHE,1,n) trj(iLMTP1,1,n) trj(iLMANK,1,n)],...
        [trj(iLHE,2,n) trj(iLMTP1,2,n) trj(iLMANK,2,n)],...
        [trj(iLHE,3,n) trj(iLMTP1,3,n) trj(iLMANK,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iLHE,1,n) trj(iLMTP5,1,n) trj(iLLANK,1,n)],...
        [trj(iLHE,2,n) trj(iLMTP5,2,n) trj(iLLANK,2,n)],...
        [trj(iLHE,3,n) trj(iLMTP5,3,n) trj(iLLANK,3,n)], [0.5 0.5 0.5] ) ;
hold on ;

line([trj(iLLANK,1,n) trj(iLLKN,1,n) trj(iLMKN,1,n)],...
      [trj(iLLANK,2,n) trj(iLLKN,2,n) trj(iLMKN,2,n)],...
      [trj(iLLANK,3,n) trj(iLLKN,3,n) trj(iLMKN,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ; hold on ;
line([trj(iLLKN,1,n) trj(iLASIS,1,n)],...
      [trj(iLLKN,2,n) trj(iLASIS,2,n)],...
      [trj(iLLKN,3,n) trj(iLASIS,3,n)], 'color' , [0 0 0] , 'linewidth' , 2
) ;  hold on ;

% Leg right
fill3( [trj(iRTHIPA,1,n) trj(iRTHIPP,1,n) trj(iRTHIDP,1,n)
trj(iRTHIDA,1,n)],...
        [trj(iRTHIPA,2,n) trj(iRTHIPP,2,n) trj(iRTHIDP,2,n)
trj(iRTHIDA,2,n)],...
        [trj(iRTHIPA,3,n) trj(iRTHIPP,3,n) trj(iRTHIDP,3,n)
trj(iRTHIDA,3,n)], [0.5 0.5 0.5] ) ; hold on ;
fill3( [trj(iRSHAPA,1,n) trj(iRSHAPP,1,n) trj(iRSHADP,1,n)
trj(iRSHADA,1,n)],...
        [trj(iRSHAPA,2,n) trj(iRSHAPP,2,n) trj(iRSHADP,2,n)
trj(iRSHADA,2,n)],...
        [trj(iRSHAPA,3,n) trj(iRSHAPP,3,n) trj(iRSHADP,3,n)
trj(iRSHADA,3,n)], [0.5 0.5 0.5] ) ; hold on ;
fill3( [trj(iRHE,1,n) trj(iRMTP1,1,n) trj(iRMTP5,1,n)],...
        [trj(iRHE,2,n) trj(iRMTP1,2,n) trj(iRMTP5,2,n)],...
        [trj(iRHE,3,n) trj(iRMTP1,3,n) trj(iRMTP5,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iRHE,1,n) trj(iRMTP1,1,n) trj(iRMANK,1,n)],...
        [trj(iRHE,2,n) trj(iRMTP1,2,n) trj(iRMANK,2,n)],...
        [trj(iRHE,3,n) trj(iRMTP1,3,n) trj(iRMANK,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iRHE,1,n) trj(iRMTP5,1,n) trj(iRLANK,1,n)],...
        [trj(iRHE,2,n) trj(iRMTP5,2,n) trj(iRLANK,2,n)],...
        [trj(iRHE,3,n) trj(iRMTP5,3,n) trj(iRLANK,3,n)], [0.5 0.5 0.5] ) ;
hold on ;

line([trj(iRLANK,1,n) trj(iRLKN,1,n) trj(iRMKN,1,n)],...
      [trj(iRLANK,2,n) trj(iRLKN,2,n) trj(iRMKN,2,n)],...
      [trj(iRLANK,3,n) trj(iRLKN,3,n) trj(iRMKN,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ; hold on ;
line([trj(iRLKN,1,n) trj(iRASIS,1,n)],...
      [trj(iRLKN,2,n) trj(iRASIS,2,n)],...
      [trj(iRLKN,3,n) trj(iRASIS,3,n)], 'color' , [0 0 0] , 'linewidth' , 2
) ;  hold on ;
end
```

```
%% Display FP3 and 8 trj

figure ( 52 ) % Force and moment with loading rate 2019-02-08
set(gcf,'position',[10 200 700 550]);

subplot(3,1,1);
h(1) = plot(t,f3r(:,1),'k-','linewidth',0.5); hold on
h(2) = plot(t,f3r(:,2),'k:','linewidth',0.5);
h(3) = plot(t,f3r(:,3),'k-','linewidth',1.5);
ylabel('Force ( N )'); % xlabel('Time ( s )');
legend(h(1:3),{'\itF_x','\itF_y','\itF_z'},'location','northeast'); box on;
grid off;
axis([0 10 -250 1750]) ;
ax = gca; ax.YTick = [-250 0 250 500 750 1000 1250 1500 1750]; % ax.XTick =
[0 500 1000 1500];
title(['FP3     ' tName.name]);

subplot(3,1,2);
h(4) = plot(t,m3r(:,1),'k:','linewidth',0.5); hold on;
h(5) = plot(t,m3r(:,2),'k-','linewidth',1.5);
ylabel('Moment ( Nm )'); % xlabel('Time ( s )');
legend(h(4:5),{'\itM_x','\itM_y'},'location','northeast'); box on; grid
off;
axis([0 10 -100 50]) ;
ax = gca; ax.YTick = [-100 -50 0 50];

subplot(3,1,3);
h(6) = plot(t,theta_PSIS,'k-','linewidth',0.5); hold on
h(7) = plot(t,theta_C7T10,'k:','linewidth',0.5); hold on
h(8) = plot(t,theta_BT,'k-','linewidth',1.5);
xlabel('Time ( s )'); ylabel('Angle ( degree )');
legend(h(6:8),{'\it\theta_P_S_I_S','\it\theta_C_7_T_1_0','\it\theta_B_T'},'
location','northeast'); box on; grid off;
axis([0 10 -30 120]) ;
ax = gca; ax.YTick = [-30 0 30 60 90 120]; % ax.XTick = [0 500 1000 1500];

figure ( 61 ) % Angular data
set(gcf,'position',[750 5 700 800]);

subplot(2,1,1);
plot(t,theta_ASIS,'k-','linewidth',0.5); hold on
plot(t,theta_CS,'k:','linewidth',0.5); hold on
plot(t,theta_FT,'k-','linewidth',1.5);
xlabel('Time ( s )'); ylabel('Angle ( degree )');
legend({'\it\theta_A_S_I_S','\it\theta_C_S','\it\theta_F_T'},'location','no
rtheast'); box on; grid off;
axis([0 10 -20 120]) ;
ax = gca; ax.YTick = [-20 0 20 40 60 80 100 120]; % ax.XTick = [0 500 1000
1500];
% title('FP3');

subplot(2,1,2);
plot(t,theta_PSIS,'k-','linewidth',0.5); hold on
plot(t,theta_C7T10,'k:','linewidth',0.5); hold on
plot(t,theta_BT,'k-','linewidth',1.5);
xlabel('Time ( s )'); ylabel('Angle ( degree )');
legend({'\it\theta_P_S_I_S','\it\theta_C_7_T_1_0','\it\theta_B_T'},'locatio
n','northeast'); box on; grid off;
axis([0 10 -20 120]) ;
```

```
ax = gca; ax.YTick = [-20 0 20 40 60 80 100 120]; % ax.XTick = [0 500 1000
1500];

% csvwrite('s1m1t1_t_force_angle.csv',[t' f3r m3r theta_ASIS theta_CS
theta_FT theta_PSIS theta_C7T10 theta_BT]); % check data structure

function [out] = fillnan(in) ;
for i = 1:3
    out(:,i) = fillmissing(in(:,i),'nearest');
end
end
function vect3d(p0,p1)

% The function plot 3D vector with arrow from point p0 to point p1.
% Example:
%       p0 = [1 2 3];    % Coordinate of the first point p0
%       p1 = [4 5 6];    % Coordinate of the second point p1
%       vect3d(p0,p1)
% YH 2017-10-04

x0 = p0(1) ; y0 = p0(2) ; z0 = p0(3) ; % Read coordinate of the first point
(tail)
x1 = p1(1) ; y1 = p1(2) ; z1 = p1(3) ; % Read coordinate of the second
point (head)
plot3([x0;x1],[y0;y1],[z0;z1],'b-','linewidth',1.5) ;     % Draw a line
between p0 and p1

p = p1 - p0 ;
% p = abs(p1 - p0) ;
alpha = 0.5 ;    % Size of arrow head relative to the length of the vector
beta = 0.8 ;     % Width of the base of the arrow head relative to the
length

hu = [x1-alpha*(p(1)+beta*(p(2)+eps)) ; x1 ; x1-alpha*(p(1)-
beta*(p(2)+eps))] ;
hv = [y1-alpha*(p(2)-beta*(p(1)+eps)) ; y1 ; y1-
alpha*(p(2)+beta*(p(1)+eps))] ;
hw = [z1-alpha*p(3) ; z1 ; z1-alpha*p(3)] ;

hold on ;
plot3(hu(:),hv(:),hw(:),'b-','linewidth',1.5) ;  % Plot arrow head
grid on ;

% xlabel('x') ; ylabel('y') ; zlabel('z') ;
% hold off ;

?
function vect3d(p0,p1)

% The function plot 3D vector with arrow from point p0 to point p1.
% Example:
%       p0 = [1 2 3];    % Coordinate of the first point p0
%       p1 = [4 5 6];    % Coordinate of the second point p1
%       vect3d(p0,p1)
% YH 2017-10-04

x0 = p0(1) ; y0 = p0(2) ; z0 = p0(3) ; % Read coordinate of the first point
(tail)
```

Dissertation Help Services

```
x1 = p1(1) ; y1 = p1(2) ; z1 = p1(3) ; % Read coordinate of the second
point (head)
plot3([x0;x1],[y0;y1],[z0;z1],'r-','linewidth',1.5) ;     % Draw a line
between p0 and p1

p = p1 - p0 ;
% p = abs(p1 - p0) ;
alpha = 0.5 ;    % Size of arrow head relative to the length of the vector
beta = 0.8 ;     % Width of the base of the arrow head relative to the
length

hu = [x1-alpha*(p(1)+beta*(p(2)+eps)) ; x1 ; x1-alpha*(p(1)-
beta*(p(2)+eps))] ;
hv = [y1-alpha*(p(2)-beta*(p(1)+eps)) ; y1 ; y1-
alpha*(p(2)+beta*(p(1)+eps))] ;
hw = [z1-alpha*p(3) ; z1 ; z1-alpha*p(3)] ;

hold on ;
plot3(hu(:),hv(:),hw(:),'r-','linewidth',1.5) ;  % Plot arrow head
grid on ;

% xlabel('x') ; ylabel('y') ; zlabel('z') ;
% hold off ;

?
%% Read data from mat file containing force plate and camera trajectory
data from Qualysis export
% 2019-07-01 YH modified for MSc project OpenSim preparation

close all; clc; clear;

%% Prepare and load for trc data (camera data - trajectory)

load s1m1t1.mat ; % 10 s of data
% load s1static.mat ; % 1 s of data
tName = whos ;
eval ( strcat ( ' myData = ' , tName.name , ' ; ' ) ) ; % change the
variable/structure name to 'myData'

start = 0.4 ; % s, start time of display
stop = 0.8 ; % s, stop time of display
gap = 10 ; % no. of smaples to skip in display

nf = myData.Frames ; % total number of frames captured by camera
fs_fp = myData.Force(1).Frequency ; % Hz, sampling rate of force plate 1 =
1000 (should be the same for plate 2 and 3; 3 is the seat)
fs_cam = myData.FrameRate ; % Hz, sampling rate of cameras = 200 Hz
usually, this is the sampling rate for the system
t = 0:1/fs_cam:(myData.Frames/fs_cam-1/fs_cam) ; % s, e.g. 0-10 s sampled
at 200 Hz = 0-2000 data points

mass = 77 ; height = 1870 ; % kg; mm; s1 RNLI

trj = myData.Trajectories.Labeled.Data ; % mm, original trajectories matrix
to be transformed to model coord
nmkr = myData.Trajectories.Labeled.Count ; % total no. of markers
markers = myData.Trajectories.Labeled.Labels ; % marker names
```

```
%% Lab coordinate to OpenSim model coordinate transformation - check the
comment out part before use!
% for mk = 1 : nmkr % Coordinate rotation
%     trj(mk,1,:) = myData.Trajectories.Labeled.Data(mk,1,:) ; % model
cooridnate +x from lab coord +x
%     trj(mk,2,:) = myData.Trajectories.Labeled.Data(mk,3,:) ; % model
cooridnate +y from lab coord +z
%     trj(mk,3,:) = -myData.Trajectories.Labeled.Data(mk,2,:) ; % model
cooridnate +z from lab coord -y
% end

% Data(67,4,2000), 67 markers, 4 channels (x,y,z,tol), 2000 data points
(200 Hz)
%     tr() = myData.Trajectories.Labeled.Data(1,1,:) ;
% Index for each marker

iLFHD = find(contains(markers,'LFHD')); % Head 4
iRFHD = find(contains(markers,'RFHD'));
iLBHD = find(contains(markers,'LBHD'));
iRBHD = find(contains(markers,'RBHD'));

iLACR = find(contains(markers,'LACR')); % Trunk 11~
iRACR = find(contains(markers,'RACR'));
iCLAV = find(contains(markers,'CLAV'));
iRBAC = find(contains(markers,'RBAC'));
iSTER = find(contains(markers,'STER'));
iC7 = find(contains(markers,'C7'));
iT10 = find(contains(markers,'T10'));
iLASIS = find(contains(markers,'LASIS'));
iRASIS = find(contains(markers,'RASIS'));
iLPSIS = find(contains(markers,'LPSIS'));
iRPSIS = find(contains(markers,'RPSIS'));

iLUPA = find(contains(markers,'LUPA')); % Arms 22~
iRUPA = find(contains(markers,'RUPA'));
iLUPP = find(contains(markers,'LUPP'));
iRUPP = find(contains(markers,'RUPP'));
iLUD = find(contains(markers,'LUD'));
iRUD = find(contains(markers,'RUD'));
iLLELB = find(contains(markers,'LLELB'));
iRLELB = find(contains(markers,'RLELB'));
iLMELB = find(contains(markers,'LMELB'));
iRMELB = find(contains(markers,'RMELB'));
iLFPA = find(contains(markers,'LFPA'));
iRFPA = find(contains(markers,'RFPA'));
iLFPP = find(contains(markers,'LFPP'));
iRFPP = find(contains(markers,'RFPP'));
iLFD = find(contains(markers,'LFD'));
iRFD = find(contains(markers,'RFD'));
iLWRA = find(contains(markers,'LWRA'));
iRWRA = find(contains(markers,'RWRA'));
iLWRB = find(contains(markers,'LWRB'));
iRWRB = find(contains(markers,'RWRB'));
iLMCP3 = find(contains(markers,'LMCP3'));
iRMCP3 = find(contains(markers,'RMCP3'));

iLMCP2 = find(contains(markers,'LMCP2'));
iRMCP2 = find(contains(markers,'RMCP2'));
iLMCP5 = find(contains(markers,'LMCP5'));
iRMCP5 = find(contains(markers,'RMCP5'));
```

```
iLTHIPA = find(contains(markers,'LTHIPA')); % Legs 30
iRTHIPA = find(contains(markers,'RTHIPA'));
iLTHIPP = find(contains(markers,'LTHIPP'));
iRTHIPP = find(contains(markers,'RTHIPP'));
iLTHIDA = find(contains(markers,'LTHIDA'));
iRTHIDA = find(contains(markers,'RTHIDA'));
iLTHIDP = find(contains(markers,'LTHIDP'));
iRTHIDP = find(contains(markers,'RTHIDP'));
iLLKN = find(contains(markers,'LLKN'));
iRLKN = find(contains(markers,'RLKN'));
iLMKN = find(contains(markers,'LMKN'));
iRMKN = find(contains(markers,'RMKN'));
iLSHAPA = find(contains(markers,'LSHAPA'));
iRSHAPA = find(contains(markers,'RSHAPA'));
iLSHAPP = find(contains(markers,'LSHAPP'));
iRSHAPP = find(contains(markers,'RSHAPP'));
iLSHADA = find(contains(markers,'LSHADA'));
iRSHADA = find(contains(markers,'RSHADA'));
iLSHADP = find(contains(markers,'LSHADP'));
iRSHADP = find(contains(markers,'RSHADP'));
iLLANK = find(contains(markers,'LLANK'));
iRLANK = find(contains(markers,'RLANK'));
iLMANK = find(contains(markers,'LMANK'));
iRMANK = find(contains(markers,'RMANK'));
iLHE = find(contains(markers,'LHE'));
iRHE = find(contains(markers,'RHE'));
iLMTP1 = find(contains(markers,'LMTP1'));
iRMTP1 = find(contains(markers,'RMTP1'));
iLMTP5 = find(contains(markers,'LMTP5'));
iRMTP5 = find(contains(markers,'RMTP5'));

for frames = 1:nf % Coordinate origin translation to between two feet
    om = [ (trj(iLMANK,1,frames)+trj(iRMANK,1,frames))/2 ,...
        min(trj(iLMTP1,2,frames),trj(iRMTP1,2,frames)) ,...
        (trj(iLMANK,3,frames)+trj(iRMANK,3,frames))/2 ] ; % origin of model
coordinate
end

%% Kinematic (trajectory) data processing
% Compute angular parameters of trunk
theta_ASIS = permute( atan2d((trj(iLASIS,3,:)-trj(iRASIS,3,:)) ,
(trj(iLASIS,1,:)-trj(iRASIS,1,:))) , [3 2 1] ) ; % deg, ASIS line angle in
the x-z plane
theta_CS = permute( atan2d((trj(iCLAV,3,:)-trj(iSTER,3,:)) ,
(trj(iCLAV,1,:)-trj(iSTER,1,:))) , [3 2 1] ) ; % deg, CLAV-STER line angle
in the x-z plane
theta_FT = abs(theta_CS - theta_ASIS) ; % deg, front trunk roll angle in
the frontal plane

theta_PSIS = permute( atan2d((trj(iLPSIS,3,:)-trj(iRPSIS,3,:)) ,
(trj(iLPSIS,1,:)-trj(iRPSIS,1,:))) , [3 2 1] ) ; % deg, PSIS line angle in
the x-z plane
theta_C7T10 = permute( atan2d((trj(iC7,3,:)-trj(iT10,3,:)) , (trj(iC7,1,:)-
trj(iT10,1,:))) , [3 2 1] ) ; % deg, C7-T10 line angle in the x-z plane
theta_BT = abs(theta_C7T10 - theta_PSIS) ; % deg, back trunk roll angle in
the frontal plane

% Compute velocity, acceleration from displacement
```

```
dis_z_LASIS = permute( trj(iLASIS,3,:) , [3 2 1] )./1000 ; % displacement
from mm to m
dis_z_RASIS = permute( trj(iRASIS,3,:) , [3 2 1] )./1000 ;
dis_z_LPSIS = permute( trj(iLPSIS,3,:) , [3 2 1] )./1000 ;
dis_z_RPSIS = permute( trj(iRPSIS,3,:) , [3 2 1] )./1000 ;

lpf = 15 ; % Hz, Lowpass filter
[b,a]=butter(4,lpf./(fs_cam./2),'low');

vel_z_LASIS = gradient( dis_z_LASIS , 1/fs_cam ) ;
velf_z_LASIS = filtfilt(b,a,vel_z_LASIS); % with filtfilt() 4-pole is 8-
pole
acc_z_LASIS = gradient( velf_z_LASIS , 1/fs_cam ) ; % peak acc from force
estimates = 17730/(654.5/9.81) = 26.53 m/s2

vel_z_RASIS = gradient( dis_z_RASIS , 1/fs_cam ) ;
velf_z_RASIS = filtfilt(b,a,vel_z_RASIS); % with filtfilt() 4-pole is 8-
pole
acc_z_RASIS = gradient( velf_z_RASIS , 1/fs_cam ) ;

vel_z_LPSIS = gradient( dis_z_LPSIS , 1/fs_cam ) ;
velf_z_LPSIS = filtfilt(b,a,vel_z_LPSIS); % with filtfilt() 4-pole is 8-
pole
acc_z_LPSIS = gradient( velf_z_LPSIS , 1/fs_cam ) ;

vel_z_RPSIS = gradient( dis_z_RPSIS , 1/fs_cam ) ;
velf_z_RPSIS = filtfilt(b,a,vel_z_RPSIS); % with filtfilt() 4-pole is 8-
pole
acc_z_RPSIS = gradient( velf_z_RPSIS , 1/fs_cam ) ;

figure (15)
subplot(3,1,1)
plot(t,dis_z_LASIS,'k-',t,dis_z_RASIS,'k-.',t,dis_z_LPSIS,'b--
',t,dis_z_RPSIS,'r:');
ylabel('Displacement ( m )'); xlabel('Time ( s )');
subplot(3,1,2)
plot(t,velf_z_LASIS,'k-',t,velf_z_RASIS,'k-.',t,velf_z_LPSIS,'b--
',t,velf_z_RPSIS,'r:');
ylabel('Velocity ( m / s )'); xlabel('Time ( s )');
subplot(3,1,3)
plot(t,acc_z_LASIS,'k-',t,acc_z_RASIS,'k-.',t,acc_z_LPSIS,'b--
',t,acc_z_RPSIS,'r:');
ylabel('Acceleration ( m / s^2 )'); xlabel('Time ( s )');
legend('z-LASIS','z-RASIS','z-LPSIS','z-RPSIS');

%% Force plate (FP) data - 3 force plates

f1 = myData.Force(1).Force()' ; % N, force plate data: 0-10000 data points
at 1000 Hz = 0-10 s
f2 = myData.Force(2).Force()' ;
f3 = myData.Force(3).Force()' ;

f1 = downsample(f1,fs_fp/fs_cam) ; % N, force down-sampled to match
trajectory at 200 Hz
f1 = fillnan(f1) ; % fillnan() to fill all nan in each column
f2 = downsample(f2,fs_fp/fs_cam) ; f2 = fillnan(f2) ;
f3 = downsample(f3,fs_fp/fs_cam) ; f3 = fillnan(f3) ;

m1 = myData.Force(1).Moment()' ; % Nm, moment
```

```
m2 = myData.Force(2).Moment()' ;
m3 = myData.Force(3).Moment()' ;

m1 = downsample(m1,fs_fp/fs_cam) ; % Nm, moment down-sampled to match
trajectory at 200 Hz
m1 = fillnan(m1) ;
m2 = downsample(m2,fs_fp/fs_cam) ; m2 = fillnan(m2) ;
m3 = downsample(m3,fs_fp/fs_cam) ; m3 = fillnan(m3) ;

cop1 = myData.Force(1).COP()' ; % mm, centre of pressure
cop2 = myData.Force(2).COP()' ;
cop3 = myData.Force(3).COP()' ;

cop1 = downsample(cop1,fs_fp/fs_cam) ; % mm, cop down-sampled to match
trajectory at 200 Hz
cop1 = fillnan(cop1) ;
cop2 = downsample(cop2,fs_fp/fs_cam) ; cop2 = fillnan(cop2) ;
cop3 = downsample(cop3,fs_fp/fs_cam) ; cop3 = fillnan(cop3) ;

%% FP Step 1 check pose of all 3 force plates (FP): if needed inverse sign
of X and Y axes FP data - Force, Moment, CoP
% r_a = [-1 0 0 ; 0 -1 0 ; 0 0 1] ; % rotation matrix to rectify Qualysis
% measurement from FP (flip signs of x and y) 2018-07-18
% r_a = [1 0 0 ; 0 1 0 ; 0 0 1] ; % no rotation as 2018-07-01
r_a = [-1 0 0 ; 0 1 0 ; 0 0 1] ; % flip x-axis only for force and moment
but not COP:

f1r = (r_a * f1')' ; % transformation using inner product with rotation
matrix
m1r = (r_a * m1')' ;
cop1r = cop1 ;
% cop1r = (r_a * cop1')' ;

f2r = (r_a * f2')' ; % transformation using inner product with rotation
matrix
m2r = (r_a * m2')' ;
cop2r = cop2 ;
% cop2r = (r_a * cop2')' ;

f3r = (r_a * f3')' ; % transformation using inner product with rotation
matrix
m3r = (r_a * m3')' ;
cop3r = cop3 ;
% cop3r = (1 * cop3')' ;

%% FP Step 2 rectify FP3 beneath rigid seat due to seat height

az0 = 0.58 ; % m, thickness of padding above sensor z- this is seat height
My = m3r(:,2) ; Mx = m3r(:,1) ;
Fx = f3r(:,1) ; Fy = f3r(:,2) ; Fz = f3r(:,3) ;

% COPx = -(Myr/Fz)
Myr = ( My - Fx * az0 ) ; % Nm, thickness rectified moment y-
COPx = - ( Myr ./ Fz )*1000 ; % mm, thickness rectified COP x-

% COPy = (Mxr/Fz)
Mxr = ( Mx + Fy * az0 ) ; % thickness rectified moment x-
COPy = ( Mxr ./ Fz ) * 1000 ; % mm, thickness rectified COP y-
```

```
m3r(:,1) = Mxr ; % Nm, rectified with thickness
m3r(:,2) = Myr ; % Nm
cop3r = [COPx COPy zeros(length(m3r),1)] ; % mm

%% FP Step 3 rotation from lab coordinates to OpenSim model coordinates
% After two steps: f1r, f2r, f3r, m1r, m2r, m3r, cop1r, cop2r, cop3r
(number indicates FP)
r_b = [1 0 0 ; 0 0 -1 ; 0 1 0] ; % rotation to OpenSim model coordinates
2018-07-06
% f1r = (r_b * f1r')' ; f2r = (r_b * f2r')' ; f3r = (r_b * f3r')' ;
% m1r = (r_b * m1r')' ; m2r = (r_b * m2r')' ; m3r = (r_b * m3r')' ;
% cop1r = (r_b * cop1r')' ; cop2r = (r_b * cop2r')' ; cop3r = (r_b *
cop3r')' ;

sw = mean(f3r(1:fs_cam*1,3)) ; % sitting weight on FP3 = average of first 1
s of z-axis force
% f3r(:,3) = f3r(:,3) - sw ; % remove static sitting weight

%% FP Rectification display

figure(11) % FP1
subplot(2,3,1);plot(t,f1(:,1),'r--',t,f1(:,2),'b-',t,f1(:,3),'k-');
% subplot(2,3,1),plot(t(1:end-3),f1(1:end-2,1),'r--',t(1:end-3),f1(1:end-
2,2),'b-',t(1:end-3),f1(1:end-2,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )'); title('Force VS Time (FP1) ')
legend({'f_x','f_y','f_z'}); box on; grid on;
subplot(2,3,4);plot(t,f1r(:,1),'r--',t,f1r(:,2),'b-',t,f1r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )');
legend({'f_x (rec)','f_y (rec)','f_z'}); box on; grid on;

subplot(2,3,2);plot(t,m1(:,1),'r--',t,m1(:,2),'b-',t,m1(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('Moment VS Time
(FP1)')
legend({'m_x','m_y','m_z'});  box on; grid on;
subplot(2,3,5);plot(t,m1r(:,1),'r--',t,m1r(:,2),'b-',t,m1r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )');
legend({'m_x (rec)','m_y (rec)','m_z'}); box on; grid on;

subplot(2,3,3);plot(t,cop1(:,1),'r--',t,cop1(:,2),'b-',t,cop1(:,3),'k-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('CoP VS Time (FP1)')
legend({'cop_x','cop_y','cop_z'});  box on; grid on;
subplot(2,3,6);plot(t,cop1r(:,1),'r--',t,cop1r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )');
legend({'cop_x (rec)','cop_y (rec)'}); box on; grid on;

figure(12) % FP2
subplot(2,3,1);plot(t,f2(:,1),'r--',t,f2(:,2),'b-',t,f2(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )'); title('Force VS Time (FP2) ')
legend({'f_x','f_y','f_z'}); box on; grid on;
subplot(2,3,4);plot(t,f2r(:,1),'r--',t,f2r(:,2),'b-',t,f2r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )');
legend({'f_x (rec)','f_y (rec)','f_z'}); box on; grid on;

subplot(2,3,2);plot(t,m2(:,1),'r--',t,m2(:,2),'b-',t,m2(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('Moment VS Time
(FP2)')
legend({'m_x','m_y','m_z'});  box on; grid on;
subplot(2,3,5);plot(t,m2r(:,1),'r--',t,m2r(:,2),'b-',t,m2r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )');
legend({'m_x (rec)','m_y (rec)','m_z'}); box on; grid on;
```

```
subplot(2,3,3);plot(t,cop2(:,1),'r--',t,cop2(:,2),'b-',t,cop2(:,3),'k-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('CoP VS Time (FP2)')
legend({'cop_x','cop_y','cop_z'});  box on; grid on;
subplot(2,3,6);plot(t,cop2r(:,1),'r--',t,cop2r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )');
legend({'cop_x (rec)','cop_y (rec)'}); box on; grid on;

figure(13) % FP3
subplot(2,3,1);plot(t,f3(:,1),'r--',t,f3(:,2),'b-',t,f3(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )'); title('Force VS Time (FP3)  ')
legend({'f_x','f_y','f_z'}); box on; grid on;
subplot(2,3,4);plot(t,f3r(:,1),'r--',t,f3r(:,2),'b-',t,f3r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )');
legend({'f_x (rec)','f_y (rec)','f_z'},'location','best'); box on; grid on;

subplot(2,3,2);plot(t,m3(:,1),'r--',t,m3(:,2),'b-',t,m3(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('Moment VS Time
(FP3)')
legend({'m_x','m_y','m_z'});  box on; grid on;
subplot(2,3,5);plot(t,m3r(:,1),'r--',t,m3r(:,2),'b-',t,m3r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )');
legend({'m_x (rec+thick)','m_y (rec+thick)','m_z'},'location','best'); box
on; grid on;

subplot(2,3,3);plot(t,cop3(:,1),'r--',t,cop3(:,2),'b-',t,cop3(:,3),'k-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('CoP VS Time (FP3)')
legend({'cop_x','cop_y','cop_z'});  box on; grid on;
subplot(2,3,6);plot(t,cop3r(:,1),'r--',t,cop3r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )');
legend({'cop_x (rec+thick)','cop_y (rec+thick)'},'location','best'); box
on; grid on;

%% Three FPs - (1) Left foot; (2) Right foot; (3) Seat pan

figure ( 21 )
set(gcf,'position',[10 5 700 800]);

subplot(3,3,1);plot(t,f1r(:,1),'r--',t,f1r(:,2),'b-',t,f1r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )'); title('FP1');
legend({'f_x (rec)','f_y (rec)','f_z'}); box on; grid on;
subplot(3,3,2);plot(t,m1r(:,1),'r--',t,m1r(:,2),'b-',t,m1r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('FP1');
legend({'m_x (rec)','m_y (rec)','m_z'}); box on; grid on;
subplot(3,3,3);plot(t,cop1r(:,1),'r--',t,cop1r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('FP1');
legend({'cop_x (rec)','cop_y (rec)'}); box on; grid on;

subplot(3,3,4);plot(t,f2r(:,1),'r--',t,f2r(:,2),'b-',t,f2r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Force ( N )'); title('FP2');
legend({'f_x (rec)','f_y (rec)','f_z'}); box on; grid on;
subplot(3,3,5);plot(t,m2r(:,1),'r--',t,m2r(:,2),'b-',t,m2r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('FP2');
legend({'m_x (rec)','m_y (rec)','m_z'}); box on; grid on;
subplot(3,3,6);plot(t,cop2r(:,1),'r--',t,cop2r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('FP2');
legend({'cop_x (rec)','cop_y (rec)'}); box on; grid on;

subplot(3,3,7);plot(t,f3r(:,1),'r--',t,f3r(:,2),'b-',t,f3r(:,3),'k-');
```

```
xlabel('Time ( s )'); ylabel('Force ( N )'); title('FP3');
legend({'f_x (rec)','f_y (rec)','f_z'},'location','best'); box on; grid on;
subplot(3,3,8);plot(t,m3r(:,1),'r--',t,m3r(:,2),'b-',t,m3r(:,3),'k-');
xlabel('Time ( s )'); ylabel('Moment ( Nm )'); title('FP3');
legend({'m_x (rec+thick)','m_y (rec+thick)','m_z'},'location','best'); box
on; grid on;
subplot(3,3,9);plot(t,cop3r(:,1),'r--',t,cop3r(:,2),'b-');
xlabel('Time ( s )'); ylabel('CoP ( mm )'); title('FP3');
legend({'cop_x (rec+thick)','cop_y (rec+thick)'},'location','best'); box
on; grid on;

%% Display 3D force (x-y-z) and COP (x-y)
% p0 = COP coordinate x-y-z(0s) in mm, p1 = force x-y-z in N but same scale
as mm COP

fp1 = (1 * myData.Force(1).ForcePlateLocation')' ; % Force-plate 1 rotation
to model coord : 1 = no rotation, r_b = model coord rotation
fp1x = fp1(:,1) ; fp1y = fp1(:,2) ; fp1z = fp1(:,3) ;
fp2 = (1 * myData.Force(2).ForcePlateLocation')' ; % Force-plate 2 rotation
to model coord
fp2x = fp2(:,1) ; fp2y = fp2(:,2) ; fp2z = fp2(:,3) ;
fp3 = (1 * myData.Force(3).ForcePlateLocation')' ; % Force-plate 2 rotation
to model coord
fp3x = fp3(:,1) ; fp3y = fp3(:,2) ; fp3z = fp3(:,3) ;

for n = start/(1/fs_cam):gap:stop/(1/fs_cam) % frame range to be plotted
Figure 31

figure ( 31 )
set(gcf,'position',[750 5 550 800]); % position of figure window on screen

% fp3_p0 = [cop3r(n,1)+mean(fp3x) cop3r(n,2)+mean(fp3y)
cop3r(n,3)+mean(fp3z)] ; % Coordinate (x y z) of the first point p0
fp3_p0 = [cop3r(n,1)+mean(fp3x) cop3r(n,2)+mean(fp3y)
cop3r(n,3)+mean(fp3z)+1000*az0] ; % Coordinate (x y z) of the first point
p0 raised az0 for FP3
fp3_p1 = fp3_p0 + f3r(n,:) ;     % Coordinate (x y z) of the second point p1
vect3d(fp3_p0,fp3_p1) ; hold on ;

fp2_p0 = [cop2r(n,1)+mean(fp2x) cop2r(n,2)+mean(fp2y)
cop2r(n,3)+mean(fp2z)] ; % Coordinate (x y z) of the first point p0
fp2_p1 = fp2_p0 + f2r(n,:) ;     % Coordinate (x y z) of the second point p1
vect3d(fp2_p0,fp2_p1) ; hold on ;

fp1_p0 = [cop1r(n,1)+mean(fp1x) cop1r(n,2)+mean(fp1y)
cop1r(n,3)+mean(fp1z)] ; % Coordinate (x y z) of the first point p0
fp1_p1 = fp1_p0 + f1r(n,:) ;     % Coordinate (x y z) of the second point p1
vect3d(fp1_p0,fp1_p1) ; hold on ;

% Trajectory data:
% myData.Trajectories.Labeled.Data(1,1,:) ;
% Data(67,4,2000), 67 markers, 4 channels (x,y,z,tol), 2000 data points
(200 Hz)

% Find upper body centroid:
upper_x = mean([trj(iLFHD,1,n) trj(iRFHD,1,n) ... % Head
                trj(iLBHD,1,n) trj(iRBHD,1,n) ...
                trj(iLACR,1,n) trj(iRACR,1,n) ... % Trunk
                trj(iRBAC,1,n) trj(iSTER,1,n) ...
                trj(iC7,1,n) trj(iT10,1,n) ...
                trj(iLPSIS,1,n) trj(iRPSIS,1,n) ...
                trj(iLASIS,1,n) trj(iRASIS,1,n) ...
                trj(iLUPA,1,n) trj(iRUPA,1,n) ... % Arms
                trj(iLUPP,1,n) trj(iRUPP,1,n) ...
                trj(iLUD,1,n) trj(iRUD,1,n) ...
                trj(iLLELB,1,n) trj(iRLELB,1,n) ...
                trj(iLMELB,1,n) trj(iRMELB,1,n) ...
                trj(iLFPA,1,n) trj(iRFPA,1,n) ...
                trj(iLFPP,1,n) trj(iRFPP,1,n) ...
                trj(iLFD,1,n) trj(iRFD,1,n) ...
                trj(iLWRA,1,n) trj(iRWRA,1,n) ...
                trj(iLWRB,1,n) trj(iRWRB,1,n) ...
                trj(iLMCP3,1,n) trj(iRMCP3,1,n) ]) ;

upper_y = mean([trj(iLFHD,2,n) trj(iRFHD,2,n) ... % Head
                trj(iLBHD,2,n) trj(iRBHD,2,n) ...
                trj(iLACR,2,n) trj(iRACR,2,n) ... % Trunk
                trj(iRBAC,2,n) trj(iSTER,2,n) ...
                trj(iC7,2,n) trj(iT10,2,n) ...
                trj(iLPSIS,2,n) trj(iRPSIS,2,n) ...
                trj(iLASIS,2,n) trj(iRASIS,2,n) ...
                trj(iLUPA,2,n) trj(iRUPA,2,n) ... % Arms
                trj(iLUPP,2,n) trj(iRUPP,2,n) ...
                trj(iLUD,2,n) trj(iRUD,2,n) ...
                trj(iLLELB,2,n) trj(iRLELB,2,n) ...
                trj(iLMELB,2,n) trj(iRMELB,2,n) ...
                trj(iLFPA,2,n) trj(iRFPA,2,n) ...
                trj(iLFPP,2,n) trj(iRFPP,2,n) ...
                trj(iLFD,2,n) trj(iRFD,2,n) ...
                trj(iLWRA,2,n) trj(iRWRA,2,n) ...
                trj(iLWRB,2,n) trj(iRWRB,2,n) ...
                trj(iLMCP3,2,n) trj(iRMCP3,2,n) ]) ;

upper_z = mean([trj(iLFHD,3,n) trj(iRFHD,3,n) ... % Head
                trj(iLBHD,3,n) trj(iRBHD,3,n) ...
                trj(iLACR,3,n) trj(iRACR,3,n) ... % Trunk
                trj(iRBAC,3,n) trj(iSTER,3,n) ...
                trj(iC7,3,n) trj(iT10,3,n) ...
                trj(iLPSIS,3,n) trj(iRPSIS,3,n) ...
                trj(iLASIS,3,n) trj(iRASIS,3,n) ...
                trj(iLUPA,3,n) trj(iRUPA,3,n) ... % Arms
                trj(iLUPP,3,n) trj(iRUPP,3,n) ...
                trj(iLUD,3,n) trj(iRUD,3,n) ...
                trj(iLLELB,3,n) trj(iRLELB,3,n) ...
                trj(iLMELB,3,n) trj(iRMELB,3,n) ...
                trj(iLFPA,3,n) trj(iRFPA,3,n) ...
                trj(iLFPP,3,n) trj(iRFPP,3,n) ...
                trj(iLFD,3,n) trj(iRFD,3,n) ...
                trj(iLWRA,3,n) trj(iRWRA,3,n) ...
                trj(iLWRB,3,n) trj(iRWRB,3,n) ...
                trj(iLMCP3,3,n) trj(iRMCP3,3,n) ]) ;

for mk = 1:nmkr % s1static.mat only contains 66 markers
    scatter3(trj(mk,1,n),trj(mk,2,n),trj(mk,3,n),100,...
        'MarkerEdgeColor','k','MarkerFaceColor','g') ; hold on

scatter3(upper_x,upper_y,upper_z,150,'MarkerEdgeColor','k','MarkerFaceColor
','r') ; hold on
end
```

```
ax = gca; ax.Projection = 'perspective'; % Foreshortening to perceive depth
in 2D representations of 3D objects
% 'orthographic' as defualt to maintain correct relative dimensions of
graphic objects
view(0,10); %
axis([-50 1000 -50 1500 -50 1500]) ; % frontal view
ax = gca; ax.XTick = [ 0 500 1000 1500]; ax.YTick = [0 500 1000 1500];
ax.ZTick = [0 500 1000 1500];
title(['start - stop / total time = ' num2str(start) ' - ' num2str(stop) '
/ ' num2str(t(end)) '  at ' num2str(fs_cam) ' Hz' ]) ;

fill3( fp1x , fp1y , fp1z , [0.85 0.85 0.85] ) ; hold on ; % FP1
text((fp1x(2)+fp1x(1))/2+100,(fp1y(4)+fp1y(1))/2-250,10,'FP1','Color',[0 0
1]) ; % FP1
line([(fp1x(1)+fp1x(2))/2 (fp1x(3)+fp1x(4))/2] , [(fp1y(1)+fp1y(2))/2
(fp1y(3)+fp1y(4))/2] ,...
    [(fp1z(1)+fp1z(2))/2 (fp1z(3)+fp1z(4))/2] , 'color' , [0 0 0] ,
'linewidth' , 1 ) ; hold on ;
line([(fp1x(1)+fp1x(4))/2 (fp1x(2)+fp1x(3))/2] , [(fp1y(1)+fp1y(4))/2
(fp1y(2)+fp1y(3))/2] ,...
    [(fp1z(1)+fp1z(4))/2 (fp1z(2)+fp1z(3))/2] , 'color' , [0 0 0] ,
'linewidth' , 1 ) ; hold on ;

fill3( fp2x , fp2y , fp2z , [0.85 0.85 0.85] ) ; hold on ; % FP2
text((fp2x(2)+fp2x(1))/2+100,(fp2y(4)+fp2y(1))/2-250,10,'FP2','Color',[0 0
1]) ; % FP2
line([(fp2x(1)+fp2x(2))/2 (fp2x(3)+fp2x(4))/2] , [(fp2y(1)+fp2y(2))/2
(fp2y(3)+fp2y(4))/2] ,...
    [(fp2z(1)+fp2z(2))/2 (fp2z(3)+fp2z(4))/2] , 'color' , [0 0 0] ,
'linewidth' , 1 ) ; hold on ;
line([(fp2x(1)+fp2x(4))/2 (fp2x(2)+fp2x(3))/2] , [(fp2y(1)+fp2y(4))/2
(fp2y(2)+fp2y(3))/2] ,...
    [(fp2z(1)+fp2z(4))/2 (fp2z(2)+fp2z(3))/2] , 'color' , [0 0 0] ,
'linewidth' , 1 ) ; hold on ;

% FP3 raised by az0
fill3( fp3x , fp3y , fp3z+1000*az0 , [0.85 0.85 0.85] ) ; hold on ; % FP3
text((fp3x(2)+fp3x(1))/2+50,(fp3y(4)+fp3y(1))/2-
250,30+1000*az0,'rFP3','Color',[0 0 1]) ; % FP3
line([(fp3x(1)+fp3x(2))/2 (fp3x(3)+fp3x(4))/2] , [(fp3y(1)+fp3y(2))/2
(fp3y(3)+fp3y(4))/2] ,...
    [(fp3z(1)+fp3z(2))/2+1000*az0 (fp3z(3)+fp3z(4))/2+1000*az0] , 'color' ,
[0 0 0] , 'linewidth' , 1 ) ; hold on ;
line([(fp3x(1)+fp3x(4))/2 (fp3x(2)+fp3x(3))/2] , [(fp3y(1)+fp3y(4))/2
(fp3y(2)+fp3y(3))/2] ,...
    [(fp3z(1)+fp3z(4))/2+1000*az0 (fp3z(2)+fp3z(3))/2+1000*az0] , 'color' ,
[0 0 0] , 'linewidth' , 1 ) ; hold on ;

% Upper body centroid vertical projection
line([upper_x upper_x] , [upper_y upper_y] , [0 upper_z] , 'color' , [1 0
0] , 'linewidth' , 1 , 'linestyle' , '--') ;  hold on ;
w_up = mass*0.75*9.81 ;                     % N, upper body weight = 75% of
body weight
upc_p0 = [upper_x upper_y upper_z] ;        % Coordinate (x y z) of the
first point p0
upc_p1 = [upper_x upper_y upper_z-w_up] ;   % Coordinate (x y z) of the
second point p1
vect3d_red(upc_p0,upc_p1) ; hold on ;
```

```
% Head
fill3( [trj(iLBHD,1,n) trj(iLFHD,1,n) trj(iRFHD,1,n) trj(iRBHD,1,n)],...  %
head x
        [trj(iLBHD,2,n) trj(iLFHD,2,n) trj(iRFHD,2,n) trj(iRBHD,2,n)],... %
y
        [trj(iLBHD,3,n) trj(iLFHD,3,n) trj(iRFHD,3,n) trj(iRBHD,3,n)],... %
z
       [0.5 0.5 0.5] ) ; hold on ;

% Trunk
line( [trj(iLACR,1,n) trj(iCLAV,1,n) trj(iSTER,1,n)],...
      [trj(iLACR,2,n) trj(iCLAV,2,n) trj(iSTER,2,n)],...
      [trj(iLACR,3,n) trj(iCLAV,3,n) trj(iSTER,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ; hold on ;
line( [trj(iCLAV,1,n) trj(iRACR,1,n) trj(iRBAC,1,n)],...
      [trj(iCLAV,2,n) trj(iRACR,2,n) trj(iRBAC,2,n)],...
      [trj(iCLAV,3,n) trj(iRACR,3,n) trj(iRBAC,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ; hold on ;
line( [trj(iLACR,1,n) trj(iC7,1,n) trj(iRACR,1,n)],...
      [trj(iLACR,2,n) trj(iC7,2,n) trj(iRACR,2,n)],...
      [trj(iLACR,3,n) trj(iC7,3,n) trj(iRACR,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ; hold on ;
fill3( [trj(iLPSIS,1,n) trj(iLASIS,1,n) trj(iRASIS,1,n)
trj(iRPSIS,1,n)],...
       [trj(iLPSIS,2,n) trj(iLASIS,2,n) trj(iRASIS,2,n)
trj(iRPSIS,2,n)],...
       [trj(iLPSIS,3,n) trj(iLASIS,3,n) trj(iRASIS,3,n) trj(iRPSIS,3,n)],
[0.5 0.5 0.5] ) ; hold on ;

line([(trj(iLPSIS,1,n)+trj(iRPSIS,1,n))/2 trj(iT10,1,n)],...
     [(trj(iLPSIS,2,n)+trj(iRPSIS,2,n))/2 trj(iT10,2,n)],...
     [(trj(iLPSIS,3,n)+trj(iRPSIS,3,n))/2 trj(iT10,3,n)], 'color' , [0 0 0]
, 'linewidth' , 2 ) ;  hold on ;
line( [trj(iT10,1,n) trj(iSTER,1,n)],...
      [trj(iT10,2,n) trj(iSTER,2,n)],...
      [trj(iT10,3,n) trj(iSTER,3,n)], 'color' , [0 0 0] , 'linewidth' , 2 )
; hold on ;

% Arm left
fill3( [trj(iLUPA,1,n) trj(iLUPP,1,n) trj(iLUD,1,n)],...
       [trj(iLUPA,2,n) trj(iLUPP,2,n) trj(iLUD,2,n)],...
       [trj(iLUPA,3,n) trj(iLUPP,3,n) trj(iLUD,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iLFPA,1,n) trj(iLFPP,1,n) trj(iLFD,1,n)],...
       [trj(iLFPA,2,n) trj(iLFPP,2,n) trj(iLFD,2,n)],...
       [trj(iLFPA,3,n) trj(iLFPP,3,n) trj(iLFD,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iLWRA,1,n) trj(iLWRB,1,n) trj(iLMCP3,1,n)],...
       [trj(iLWRA,2,n) trj(iLWRB,2,n) trj(iLMCP3,2,n)],...
       [trj(iLWRA,3,n) trj(iLWRB,3,n) trj(iLMCP3,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
line([trj(iLMELB,1,n) trj(iLLELB,1,n) trj(iLACR,1,n)],...
     [trj(iLMELB,2,n) trj(iLLELB,2,n) trj(iLACR,2,n)],...
     [trj(iLMELB,3,n) trj(iLLELB,3,n) trj(iLACR,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ;  hold on ;
line([trj(iLWRB,1,n) trj(iLLELB,1,n)],...
     [trj(iLWRB,2,n) trj(iLLELB,2,n)],...
     [trj(iLWRB,3,n) trj(iLLELB,3,n)], 'color' , [0 0 0] , 'linewidth' , 2
) ;  hold on ;
```

```
% Arm right
fill3( [trj(iRUPA,1,n) trj(iRUPP,1,n) trj(iRUD,1,n)],...
       [trj(iRUPA,2,n) trj(iRUPP,2,n) trj(iRUD,2,n)],...
       [trj(iRUPA,3,n) trj(iRUPP,3,n) trj(iRUD,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iRFPA,1,n) trj(iRFPP,1,n) trj(iRFD,1,n)],...
       [trj(iRFPA,2,n) trj(iRFPP,2,n) trj(iRFD,2,n)],...
       [trj(iRFPA,3,n) trj(iRFPP,3,n) trj(iRFD,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iRWRA,1,n) trj(iRWRB,1,n) trj(iRMCP3,1,n)],...
       [trj(iRWRA,2,n) trj(iRWRB,2,n) trj(iRMCP3,2,n)],...
       [trj(iRWRA,3,n) trj(iRWRB,3,n) trj(iRMCP3,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
line([trj(iRMELB,1,n) trj(iRLELB,1,n) trj(iRACR,1,n)],...
     [trj(iRMELB,2,n) trj(iRLELB,2,n) trj(iRACR,2,n)],...
     [trj(iRMELB,3,n) trj(iRLELB,3,n) trj(iRACR,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ;  hold on ;
line([trj(iRWRB,1,n) trj(iRLELB,1,n)],...
     [trj(iRWRB,2,n) trj(iRLELB,2,n)],...
     [trj(iRWRB,3,n) trj(iRLELB,3,n)], 'color' , [0 0 0] , 'linewidth' , 2
) ;   hold on ;

% Leg left
fill3( [trj(iLTHIPA,1,n) trj(iLTHIPP,1,n) trj(iLTHIDP,1,n)
trj(iLTHIDA,1,n)],...
       [trj(iLTHIPA,2,n) trj(iLTHIPP,2,n) trj(iLTHIDP,2,n)
trj(iLTHIDA,2,n)],...
       [trj(iLTHIPA,3,n) trj(iLTHIPP,3,n) trj(iLTHIDP,3,n)
trj(iLTHIDA,3,n)], [0.5 0.5 0.5] ) ; hold on ;
fill3( [trj(iLSHAPA,1,n) trj(iLSHAPP,1,n) trj(iLSHADP,1,n)
trj(iLSHADA,1,n)],...
       [trj(iLSHAPA,2,n) trj(iLSHAPP,2,n) trj(iLSHADP,2,n)
trj(iLSHADA,2,n)],...
       [trj(iLSHAPA,3,n) trj(iLSHAPP,3,n) trj(iLSHADP,3,n)
trj(iLSHADA,3,n)], [0.5 0.5 0.5] ) ; hold on ;
fill3( [trj(iLHE,1,n) trj(iLMTP1,1,n) trj(iLMTP5,1,n)],...
       [trj(iLHE,2,n) trj(iLMTP1,2,n) trj(iLMTP5,2,n)],...
       [trj(iLHE,3,n) trj(iLMTP1,3,n) trj(iLMTP5,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iLHE,1,n) trj(iLMTP1,1,n) trj(iLMANK,1,n)],...
       [trj(iLHE,2,n) trj(iLMTP1,2,n) trj(iLMANK,2,n)],...
       [trj(iLHE,3,n) trj(iLMTP1,3,n) trj(iLMANK,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iLHE,1,n) trj(iLMTP5,1,n) trj(iLLANK,1,n)],...
       [trj(iLHE,2,n) trj(iLMTP5,2,n) trj(iLLANK,2,n)],...
       [trj(iLHE,3,n) trj(iLMTP5,3,n) trj(iLLANK,3,n)], [0.5 0.5 0.5] ) ;
hold on ;

line([trj(iLLANK,1,n) trj(iLLKN,1,n) trj(iLMKN,1,n)],...
     [trj(iLLANK,2,n) trj(iLLKN,2,n) trj(iLMKN,2,n)],...
     [trj(iLLANK,3,n) trj(iLLKN,3,n) trj(iLMKN,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ; hold on ;
line([trj(iLLKN,1,n) trj(iLASIS,1,n)],...
     [trj(iLLKN,2,n) trj(iLASIS,2,n)],...
     [trj(iLLKN,3,n) trj(iLASIS,3,n)], 'color' , [0 0 0] , 'linewidth' , 2
) ;  hold on ;

% Leg right
fill3( [trj(iRTHIPA,1,n) trj(iRTHIPP,1,n) trj(iRTHIDP,1,n)
trj(iRTHIDA,1,n)],...
```

```
       [trj(iRTHIPA,2,n) trj(iRTHIPP,2,n) trj(iRTHIDP,2,n)
trj(iRTHIDA,2,n)],...
       [trj(iRTHIPA,3,n) trj(iRTHIPP,3,n) trj(iRTHIDP,3,n)
trj(iRTHIDA,3,n)], [0.5 0.5 0.5] ) ; hold on ;
fill3( [trj(iRSHAPA,1,n) trj(iRSHAPP,1,n) trj(iRSHADP,1,n)
trj(iRSHADA,1,n)],...
       [trj(iRSHAPA,2,n) trj(iRSHAPP,2,n) trj(iRSHADP,2,n)
trj(iRSHADA,2,n)],...
       [trj(iRSHAPA,3,n) trj(iRSHAPP,3,n) trj(iRSHADP,3,n)
trj(iRSHADA,3,n)], [0.5 0.5 0.5] ) ; hold on ;
fill3( [trj(iRHE,1,n) trj(iRMTP1,1,n) trj(iRMTP5,1,n)],...
       [trj(iRHE,2,n) trj(iRMTP1,2,n) trj(iRMTP5,2,n)],...
       [trj(iRHE,3,n) trj(iRMTP1,3,n) trj(iRMTP5,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iRHE,1,n) trj(iRMTP1,1,n) trj(iRMANK,1,n)],...
       [trj(iRHE,2,n) trj(iRMTP1,2,n) trj(iRMANK,2,n)],...
       [trj(iRHE,3,n) trj(iRMTP1,3,n) trj(iRMANK,3,n)], [0.5 0.5 0.5] ) ;
hold on ;
fill3( [trj(iRHE,1,n) trj(iRMTP5,1,n) trj(iRLANK,1,n)],...
       [trj(iRHE,2,n) trj(iRMTP5,2,n) trj(iRLANK,2,n)],...
       [trj(iRHE,3,n) trj(iRMTP5,3,n) trj(iRLANK,3,n)], [0.5 0.5 0.5] ) ;
hold on ;

line([trj(iRLANK,1,n) trj(iRLKN,1,n) trj(iRMKN,1,n)],...
     [trj(iRLANK,2,n) trj(iRLKN,2,n) trj(iRMKN,2,n)],...
     [trj(iRLANK,3,n) trj(iRLKN,3,n) trj(iRMKN,3,n)], 'color' , [0 0 0] ,
'linewidth' , 2 ) ; hold on ;
line([trj(iRLKN,1,n) trj(iRASIS,1,n)],...
     [trj(iRLKN,2,n) trj(iRASIS,2,n)],...
     [trj(iRLKN,3,n) trj(iRASIS,3,n)], 'color' , [0 0 0] , 'linewidth' , 2
) ;  hold on ;
end

%% Display FP3 and 8 trj

figure ( 52 ) % Force and moment with loading rate 2019-02-08
set(gcf,'position',[10 200 700 550]);

subplot(3,1,1);
h(1) = plot(t,f3r(:,1),'k-','linewidth',0.5); hold on
h(2) = plot(t,f3r(:,2),'k:','linewidth',0.5);
h(3) = plot(t,f3r(:,3),'k-','linewidth',1.5);
ylabel('Force ( N )'); % xlabel('Time ( s )');
legend(h(1:3),{'\itF_x','\itF_y','\itF_z'},'location','northeast'); box on;
grid off;
axis([0 10 -250 1750]) ;
ax = gca; ax.YTick = [-250 0 250 500 750 1000 1250 1500 1750]; % ax.XTick =
[0 500 1000 1500];
title(['FP3     ' tName.name]);

subplot(3,1,2);
h(4) = plot(t,m3r(:,1),'k:','linewidth',0.5); hold on;
h(5) = plot(t,m3r(:,2),'k-','linewidth',1.5);
ylabel('Moment ( Nm )'); % xlabel('Time ( s )');
legend(h(4:5),{'\itM_x','\itM_y'},'location','northeast'); box on; grid
off;
axis([0 10 -100 50]) ;
ax = gca; ax.YTick = [-100 -50 0 50];

subplot(3,1,3);
```

```
h(6) = plot(t,theta_PSIS,'k-','linewidth',0.5); hold on
h(7) = plot(t,theta_C7T10,'k:','linewidth',0.5); hold on
h(8) = plot(t,theta_BT,'k-','linewidth',1.5);
xlabel('Time ( s )'); ylabel('Angle ( degree )');
legend(h(6:8),{'\it\theta_P_S_I_S','\it\theta_C_7_T_1_0','\it\theta_B_T'},'
location','northeast'); box on; grid off;
axis([0 10 -30 120]) ;
ax = gca; ax.YTick = [-30 0 30 60 90 120]; % ax.XTick = [0 500 1000 1500];

figure ( 61 ) % Angular data
set(gcf,'position',[750 5 700 800]);

subplot(2,1,1);
plot(t,theta_ASIS,'k-','linewidth',0.5); hold on
plot(t,theta_CS,'k:','linewidth',0.5); hold on
plot(t,theta_FT,'k-','linewidth',1.5);
xlabel('Time ( s )'); ylabel('Angle ( degree )');
legend({'\it\theta_A_S_I_S','\it\theta_C_S','\it\theta_F_T'},'location','no
rtheast'); box on; grid off;
axis([0 10 -20 120]) ;
ax = gca; ax.YTick = [-20 0 20 40 60 80 100 120]; % ax.XTick = [0 500 1000
1500];
% title('FP3');

subplot(2,1,2);
plot(t,theta_PSIS,'k-','linewidth',0.5); hold on
plot(t,theta_C7T10,'k:','linewidth',0.5); hold on
plot(t,theta_BT,'k-','linewidth',1.5);
xlabel('Time ( s )'); ylabel('Angle ( degree )');
legend({'\it\theta_P_S_I_S','\it\theta_C_7_T_1_0','\it\theta_B_T'},'locatio
n','northeast'); box on; grid off;
axis([0 10 -20 120]) ;
ax = gca; ax.YTick = [-20 0 20 40 60 80 100 120]; % ax.XTick = [0 500 1000
1500];

% csvwrite('s1m1t1_t_force_angle.csv',[t' f3r m3r theta_ASIS theta_CS
theta_FT theta_PSIS theta_C7T10 theta_BT]); % check data structure

function [out] = fillnan(in) ;
for i = 1:3
    out(:,i) = fillmissing(in(:,i),'nearest');
end
end"
```