FreeBody 1.0

Software and Model Description

And

User's Guide

v1.1

Dr Dan Cleather

St Mary's University

7th July, 2015

Please cite the following papers:

Cleather, D. J., & Bull, A. M. J. (2015). The development of a segment-based musculoskeletal model of the lower limb: Introducing FreeBody. *Royal Society Open Science, 2,* 140449. doi: 10.1098/rsos.140449

Cleather, D. J., & Bull, A. M. J. (2011). An optimization-based simultaneous approach to the determination of muscular, ligamentous, and joint contact forces provides insight into musculoligamentous interaction. *Annals of Biomedical Engineering*, *39*, 1925-1934. doi: 10.1007/s10439-011-0303-8

Cleather, D. J., Goodwin, J. E., & Bull, A. M. J. (2011). An optimization approach to inverse dynamics provides insight as to the function of the biarticular muscles during vertical jumping. *Annals of Biomedical Engineering*, *39*, 147-160. doi: 10.1007/s10439-010-0161-9

Cleather, D. J., & Bull, A. M. J. (2010). Lower extremity musculoskeletal geometry affects the calculation of patellofemoral forces in vertical jumping and weightlifting. *Proceedings of the Institution of Mechanical Engineers: Part H: Journal of Engineering in Medicine*, *224*, 1073-1083. doi: 10.1243/09544119JEIM731

Contents

§ 1: Introduction

This document describes the FreeBody modelling software. FreeBody consists of a framework for musculoskeletal modelling that incorporates a model of the lower limb. The purpose of the software is to allow the musculoskeletal geometry of a subject to be described, based upon motion capture data. This in turn then allows the calculation of muscle and joint forces, based upon an inverse optimization calculation method.

The philosophy behind the public release of FreeBody is to provide a musculoskeletal modelling tool with the greatest amount of utility for the largest range of users. To this end, the software is provided both as a pre-compiled MATLAB® application that is ready for immediate use and as the original source code. This should then allow a range of implementations, from the use of the model as a "black box" for calculating muscle and joint forces, through to using the code as a library of functions or a framework for the development of bespoke models.

The purpose of this manual is thus to provide the information that any user might need in order to apply FreeBody in the fashion desired. The manual is thus organised to make it as easy as possible for a user to find the required information.

Chapter 2 describes the version of FreeBody that is packaged as a MATLAB® application. This application allows the software to be used as a black box application for the calculation of muscle and joint forces in the lower limb. The chapter describes the data that must be collected to facilitate this, how the data should be organised for use in the model (data pre-processing) and how to install and run the model.

In contrast, Chapter 3 provides a detailed description of the source code. This includes an overview of how each source code file relates to each other. In addition, there is a focus on the description of the various input files, as the manipulation of these is the easiest way to alter the model.

If using FreeBody, please cite the following papers:

Cleather, D. J., & Bull, A. M. J. (2015). The development of a segment-based musculoskeletal model of the lower limb: Introducing FreeBody. *Royal Society Open Science, 2*, 140449. doi: 10.1098/rsos.140449

Cleather, D. J., & Bull, A. M. J. (2011). An optimization-based simultaneous approach to the determination of muscular, ligamentous, and joint contact forces provides insight into musculoligamentous interaction. *Annals of Biomedical Engineering, 39*, 1925-1934. doi: 10.1007/s10439-011-0303-8

Cleather, D. J., Goodwin, J. E., & Bull, A. M. J. (2011). An optimization approach to inverse dynamics provides insight as to the function of the biarticular muscles during vertical jumping. *Annals of Biomedical Engineering, 39*, 147-160. doi: 10.1007/s10439-010-0161-9

Cleather, D. J., & Bull, A. M. J. (2010). Lower extremity musculoskeletal geometry affects the calculation of patellofemoral forces in vertical jumping and weightlifting. *Proceedings of the Institution of Mechanical Engineers: Part H: Journal of Engineering in Medicine, 224*, 1073-1083. doi: 10.1243/09544119JEIM731

§ 2: The FreeBody MATLAB® App

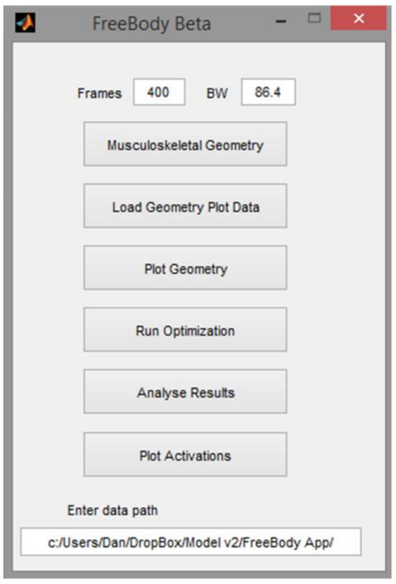
Installing FreeBody

This chapter describes an implementation of FreeBody that is packaged as a MATLAB® app. The software is provided as a .ZIP file that can be downloaded from www.msksoftware.org.uk. Installation is straightforward – simply extract the contents of the .ZIP file to the desired directory, and then run the file "FreeBody Beta.mlappinstall". This will launch MATLAB® and ask the user if they wish to install the application. Once the user has confirmed this, the app will be fully installed.

Running the FreeBody Demo

The software is provided including a pre-loaded example (that of an 86.4 kg male subject performing a vertical jump). This data can thus be used to try out the software immediately after installation. Clicking on the "FreeBody Beta" icon on the "Apps" pane of MATLAB® launches a graphical user interface (GUI) like the one depicted in Figure 2.1. The first step is to enter the path name to the directory in which FreeBody was installed in the box at the bottom of the GUI (and to press return). The analysis pipeline of FreeBody is then run by clicking each button in turn and waiting for the given operation to complete.

Figure 2.1. The FreeBody GUI.


Frames 400 BW 86.4

Enter data path
c:/Users/Dan/DropBox/Model v2/FreeBody App/

The "Musculoskeletal Geometry" button launches a Windows application that is external to MATLAB®. This application takes the raw data and performs all of the calculations of the model prior to the optimization. In particular, the application specifies the musculoskeletal geometry of the model for each frame. Once this process has been run, the user can plot the musculoskeletal geometry of the model. Firstly, the musculoskeletal geometry must be loaded into MATLAB® by clicking the "Load Geometry Plot Data" button – when this process is complete it will be indicated in the MATLAB® command window. Then the dynamic geometry of the model can be visualised by clicking the "Plot Geometry" button. Note that it is only necessary to load the geometry data once, then the plot can be run multiple times.

The next step is to run the optimization to calculate the muscle and joint forces. This is achieved by clicking the fourth button – "Run Optimization". The optimization will take some time to complete and will be greatly dependent on the speed of the platform being used. The user can monitor the progress of the optimization in the MATLAB® command window.

Finally, once the optimization is complete, the user has two options to visualise the results. The "Analyse Results" button will produce a number of plots describing muscular and joint forces (Figure 2.2). The "Plot Activations" button will launch a dynamic plot of the musculoskeletal geometry where the colour of the muscle elements will change reflecting the level of activity (more red indicates more activation; Figure 2.3).

Figure 2.2. Model analysis within FreeBody.

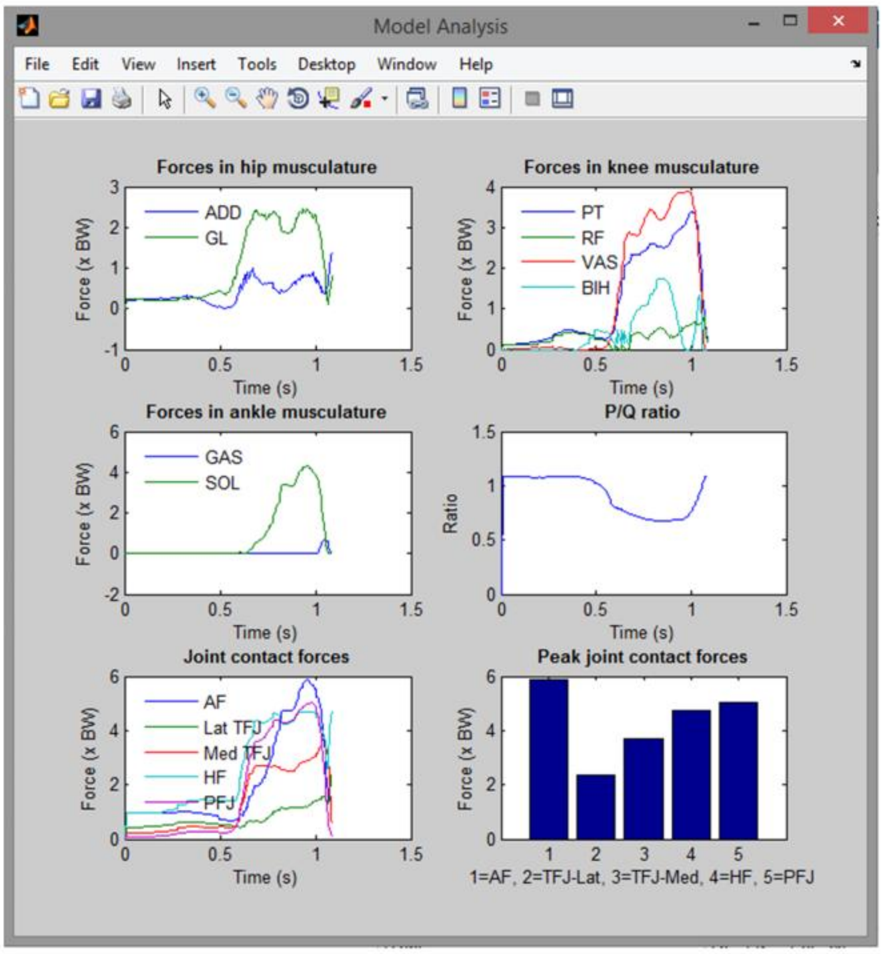
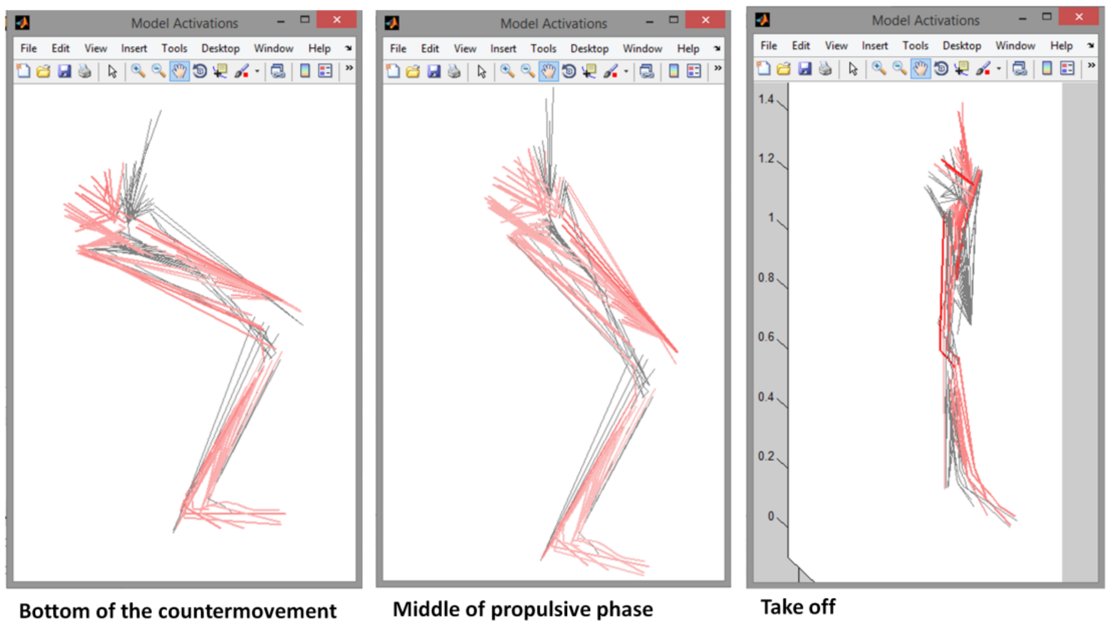

Model Analysis
File Edit View Insert Tools Desktop Window Help
Forces in hip musculature
ADD
GL
Force (x BW)
Time (s)
Forces in knee musculature
PT
RF
VAS
BIH
Forces in ankle musculature
GAS
SOL
P/Q ratio
Ratio
Joint contact forces
AF
Lat TFJ
Med TFJ
HF
PFJ
Peak joint contact forces
1=AF, 2=TFJ-Lat, 3=TFJ-Med, 4=HF, 5=PFJ

Figure 2.3. Visualising muscular activations with FreeBody.


Model Activations
File Edit View Insert Tools Desktop Window Help
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Bottom of the countermovement
Middle of propulsive phase
Take off

For a given data set, it is only necessary to run the "Musculoskeletal Geometry" and "Run Optimization" routines once. These are then saved within the file structure of FreeBody. These results can then be visualised even if FreeBody has been closed and restarted. However, it will be necessary to load the geometry plot data before either of the geometry plots can be utilised.

Using FreeBody with your own data

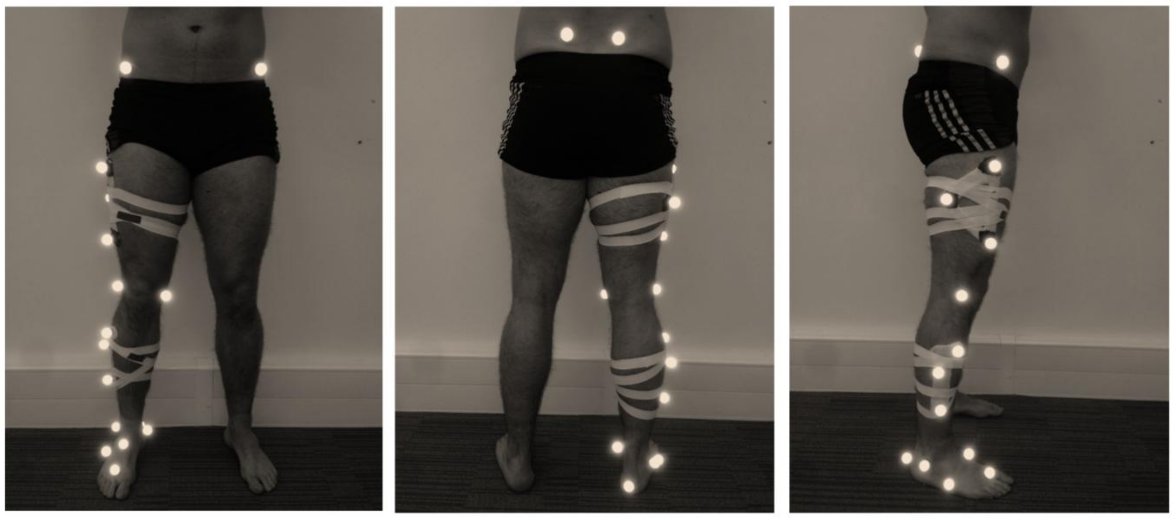
The lower limb model within FreeBody is based upon the input of motion capture data (positions of anatomical landmarks) and the ground reaction force. Table 2.1 lists the standard marker positions necessary to use this model, and Figure 2.4 illustrates the placement of these markers on a subject.

Table 2.1. Marker positions used for data capture.

Marker	Location
FCC	Calcaneus
FMT	Tuberosity of the fifth metatarsal
FM2*	Head of the second metatarsal
TF	Additional marker placed on the foot
FAM*	Apex of the lateral malleolus
TAM*	Apex of the medial malleolus
C1, C2, C3	Additional markers placed on the calf segment
FLE*	Lateral femoral epicondyle
FME*	Medial femoral epicondyle
T1, T2, T3	Additional markers placed on the thigh segment
RASIS*	Right anterior superior iliac spine
LASIS*	Left anterior superior iliac spine
RPSIS*	Right posterior superior iliac spine
LPSIS*	Left posterior superior iliac spine

Notes: * indicates those markers that must be placed upon the indicated anatomical landmark. The remaining markers could be placed in other positions (as long as they are still on the same segment).

Figure 2.4. Position of markers on a subject.

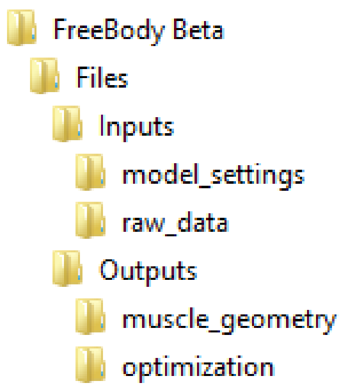


The raw data then needs to be pre-processed into a standard format for input into FreeBody. Details as to the input files that need to be created from the raw data are provided in the section “Raw Data Pre Processing” below.

File structure

The file structure of FreeBody is illustrated in Figure 2.5. In order for a user to use their own data their input files need to be copied into the folder /FreeBody Beta/Files/Inputs/raw_data/. The model settings are found in the folder /FreeBody Beta/Files/Inputs/model_settings and a number of the model assumptions can be varied by altering these files. More detail as to the model settings are provided in the more detailed code description that follows. Finally, after the model has been run, its outputs are saved in the folder /FreeBody Beta/Files/Outputs/.

Figure 2.5. FreeBody’s file structure.



```
FreeBody Beta
  Files
    Inputs
      model_settings
      raw_data
    Outputs
      muscle_geometry
      optimization
```

Raw Data Pre Processing

A number of input files are required in order to run FreeBody, which specify both the raw data, and the characteristics (settings) of the model. Only the raw data files are described in this chapter.

Input File	**File Content**
basics.txt	1 6 5 163 14 200 0.014 400

This file specifies the most fundamental characteristics of the model. The second row gives this information. The first three numbers on this row are the number of segments, muscles and ligaments in the model, and should not be changed if using the MATLAB® app. The fourth, fifth and sixth numbers will need to be changed however. The fourth number is the frequency of the raw data (in Hz), the fifth number is the size of the markers, and the sixth number is the number of frames to be analysed.

force_data.txt	498 6
	227.2 214.4 0 0.2 43.1 432.7
	227.0 214.4 0 0.3 43.1 432.6
	226.9 214.5 0 0.3 43.1 432.4
	226.6 ....

This gives the raw data pertaining to the external forces acting on the subject. The default setting for the MATLAB® app is that there is one external force (the ground reaction force) acting on the foot. The first two numbers in the file describe the size of the following matrix. In the matrix, each row is one frame. Within a row, the first 3 numbers give the point at which the force can be considered to act (the centre of pressure from the force plate), and the second 3 numbers the force itself. These values are given in the laboratory coordinate frame.

position_data.txt	490	54				
	107.0	237.8	39.1	212.0	178.3	.....
	107.0	237.8	39.1	212.0	178.3	.....
	107.1	.....				

This gives the raw data pertaining to the positions of each marker. The first two numbers describe the size of the following matrix. In the matrix, each row is one frame. Each group of three numbers on a row, describes the position of one marker (given in the laboratory coordinate frame). The markers are arranged within each row as follows: first all of the markers on segment 0, next all of the markers on segment 1, etc. The order of the markers (which are described in Table 2.1) within the file should be as follows: FCC, FMT, FM2, FT, FAM, TAM, C1, C2, C3, FLE, FME, T1, T2, T3, RASIS, LASIS, RPSIS, LPSIS.

anthropometry.txt	5	6				
	1.2	0.442	0.558	0.006	0.0014	0.005
	3.7	0.446	0.554	0.0525	0.0086	0.05
	12.2	0.41	0.59	0.2703	0.0554	0.270
	9.7	0.612	0.388	0.0887	0.0808	0.071
	0	0	0	0	0	0

This file gives the anthropometry of each segment. The first two numbers describe the size of the matrix that follows. Each following row the represents a segment, starting with segment 0. Within a row, the first number gives the mass of the segment. The second number gives the location of the COM of the segment (assuming it lies on the longitudinal axis of the segment) expressed as the percentage of total length from proximal to distal. The third number gives the percentage length from COM to the distal end of the segment. The final three numbers describe the moment of inertia of the segment (given in the segment fixed coordinate system). The ICLLM uses the model of de Leva [1] to calculate the subject specific anthropometry, and an Excel file (anthropometry.xls) is provided to ease calculation of these parameters.

calib_data.txt	1	54				
	107.0	237.8	39.1	212.0	178.3	.....

This gives the raw data pertaining to the positions of each marker when the subject is standing in the anatomical position (calibration position). The format of the file is the same as the position_data.txt file.

coord_map.txt	3 1 3 -2

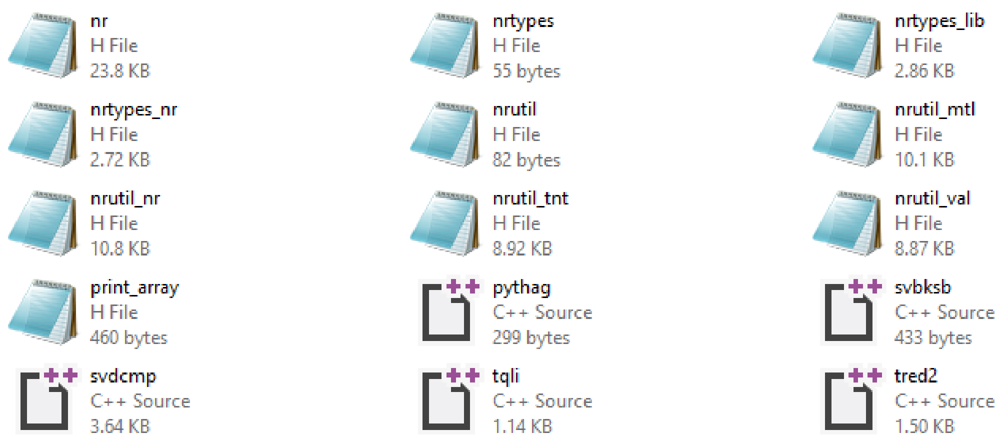
This file specifies the mapping from the coordinate system of the laboratory where the data was collected to the model coordinate system. It is likely that this will need to be changed based on the particular configuration of the user's laboratory. A description of this mapping is provided in Chapter 3.

§ 3: Detailed Software Description

The following pages present a detailed description of the software, arranged to mirror the order of the software.

The model makes an extensive use of objects in order to store data. In particular there are 3 important classes (`Segment`, `Muscle`, `Structure`) which are used to store data about the individual segments, muscles and the calibration data. Each class also has a library of functions that operate upon its objects, and that are used for manipulating the data.

In addition to the class functions, the model also draws from a number of different libraries. Firstly, the model uses a number of routines from "Numerical Recipes in C++". Due to license restrictions these are not included with the model. Users should download the following files from www.nr.com and compile them alongside the FreeBody source code:



nr H File 23.8 KB	nrtypes H File 55 bytes	nrtypes_lib H File 2.86 KB
nrtypes_nr H File 2.72 KB	nrutil H File 82 bytes	nrutil_mtl H File 10.1 KB
nrutil_nr H File 10.8 KB	nrutil_tnt H File 8.92 KB	nrutil_val H File 8.87 KB
print_array H File 460 bytes	pythag C++ Source 299 bytes	svbksb C++ Source 433 bytes
svdcmp C++ Source 3.64 KB	tqli C++ Source 1.14 KB	tred2 C++ Source 1.50 KB

In addition, the model is provided with two further libraries, described in the header files mathematics.h and kinematics.h (and their associated .cpp files). These provide common mathematical and kinematic operations.

It should be noted that the detail of class and library functions are not described in this document.

The model largely relies on the quaternion representation of rotation, and many of the calculations are made using quaternion algebra and operations.

Software Main File (Model v2.cpp)

- The entry point for FreeBody is the file “Model v2.cpp”.

- The number of segments, muscles, and ligaments, the frequency of the data and the number of frames are input here:

Input File	**File Content**
basics.txt	1 6 5 163 14 200 0.014 400

This file specifies the most fundamental characteristics of the model. The first two numbers specify the size of the matrix that follows. The second row gives the basic information. The first three numbers on this row are the number of segments, muscles and ligaments in the model. The fourth number is the frequency of the raw data (in Hz), the fifth number is the size of the markers, and the sixth number is the number of frames to be analysed.

- This file first defines the fundamental variables and data structures used in the model. In particular, the objects `Segment **segment_data[frames]`, `Muscle **muscle_data[frames]` and `Muscle *muscle_model[muscles]` are initialized here.

- Each module involved in the analysis pipeline is then called sequentially from this main file.

Module 1: Data Input (m01_00_data_input.cpp)

The purpose of this module is to input the raw data into the model and then to process it into the format that will be used in the rest of the model. The raw data includes a specification as to the structure of the model itself.

The model is provided with a routine to input and process raw data in a given format (described in this chapter). However, it is perfectly feasible to input data in other formats provided that new code is written to process this data into the format required by the model (also described in this chapter). If this code is written, it can be slotted into the model in this module.

Inputs

- Raw data (marker positions, external forces, calibration data)
- Mapping of laboratory to model
- Model structure

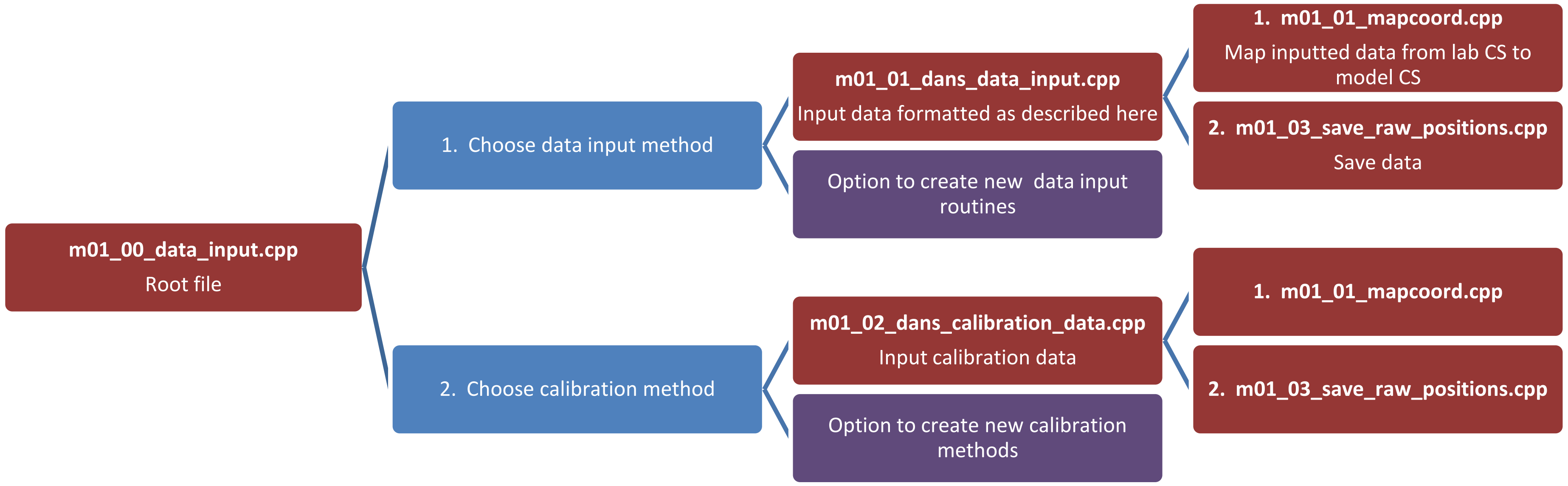
Processes

Data is input from text files and saved to relevant objects for later use.

Outputs

- All raw data saved to `Segment **segment_data[]`
- All calibration data saved to `Structure *calibrate_pos[]`

Module Overview


m01_00_data_input.cpp
Root file
1. Choose data input method
2. Choose calibration method
m01_01_dans_data_input.cpp
Input data formatted as described here
Option to create new data input routines
m01_02_dans_calibration_data.cpp
Input calibration data
Option to create new calibration methods
1. m01_01_mapcoord.cpp
Map inputted data from lab CS to model CS
2. m01_03_save_raw_positions.cpp
Save data
1. m01_01_mapcoord.cpp
2. m01_03_save_raw_positions.cpp

Selected Detailed Software Description

m01_01_dans_data_input.cpp

This routine takes raw data that is formatted as described below, and processes it for use by FreeBody. Raw data is provided in a number of text files that specify both the details of the model, and the kinematics and kinetics of the movement being analysed. The data for each segment and each frame is saved to `Segment **segment_data[]`.

Input File	**Example File Content**

Model Settings

coord_map.txt	3 1 3 -2

This file specifies the mapping from the coordinate system of the laboratory where the data was collected to the model coordinate system. The model coordinate system is with the x axis running from posterior to anterior, the y axis from inferior to superior, and the z axis running medio-laterally (with the positive direction being towards the person's right hand).

The first number in the file simply specifies the number of numbers to follow. The remaining 3 numbers specify the mapping based upon the following system. The first number given describes the laboratory axis that maps to the model x axis, the second number the laboratory axis that maps to the model y axis, etc. The numbers themselves represent the laboratory axes, with 1 being the x axis, 2 the y axis and 3 the z axis. Thus in the example the mapping described is as follows:

Laboratory	➡ Model
x'	x
z'	y
-y'	z

markers.txt	5 4 5 5 4 0

This file specifies the number of markers per segment. The first number gives the remaining number of numbers in the file, then each following number gives the number of markers per segment starting from segment 0 (where the segment numbers are as defined in the model_tree_structure.txt).

model_tree_structure.txt	5 5 0 1 2 3 4

	-1 0 1 2 2
	1 1 1 0 0
	0 1 1 1 0
	1 1 1 0 0

This file specifies the order in which segments link to one another in the model. The first two numbers represent the size of the matrix that follows. Within this matrix, each column represents a segment. The first row gives the number of each segment (numbered in order). The following row then gives the number of the segment that attaches distally to the segment in that column (-1 is the ground). The remaining rows are used in the definition of centres of rotations of joints and contact positions in the next module.

external_forces.txt	5 1 0 0 0 0

This file describes how many external forces are acting on each segment. The first number gives the remaining number of numbers in the file. The remainder of the numbers indicate how many external forces act on each segment, starting from segment 0.

Raw Data

force_data.txt	498 6
	227.2 214.4 0 0.2 43.1 432.7
	227.0 214.4 0 0.3 43.1 432.6
	226.9 214.5 0 0.3 43.1 432.4
	226.6 ....

This gives the raw data pertaining to the external forces acting on the subject. The first two numbers describe the size of the following matrix. In the matrix, each row is one frame. Within a row, the first 3 numbers give the point at which the force can be considered to act, and the second 3 numbers the force itself. These values are given in the laboratory coordinate frame.

position_data.txt	490 54
	107.0 237.8 39.1 212.0 178.3 .....
	107.0 237.8 39.1 212.0 178.3 .....
	107.1 .....

This gives the raw data pertaining to the positions of each marker. The first two numbers describe the size of the following matrix. In the matrix, each row is one frame. Each group of three numbers on a row, describes the position of one marker (given in the laboratory coordinate frame). The markers are arranged within each row as follows: first all of the markers on segment 0, next all of the markers on segment 1, etc. The order of the markers must be the same as in the calibration file (calib_data.txt).

m01_02_dans_calibration_data.cpp

This routine inputs the calibration data for the model. This is saved to `Structure *calibrate_pos[]`.

Calibration Data

anthropometry.txt	5	6				
	1.2	0.442	0.558	0.006	0.0014	0.005
	3.7	0.446	0.554	0.0525	0.0086	0.05
	12.2	0.41	0.59	0.2703	0.0554	0.270
	9.7	0.612	0.388	0.0887	0.0808	0.071
	0	0	0	0	0	0

This file gives the anthropometry of each segment. The first two numbers describe the size of the matrix that follows. Each following row the represents a segment, starting with segment 0. Within a row, the first number gives the mass of the segment. The second number gives the location of the COM of the segment (assuming it lies on the longitudinal axis of the segment) expressed as the percentage of total length from proximal to distal. The third number gives the percentage length from COM to the distal end of the segment. The final three numbers describe the moment of inertia of the segment (given in the segment fixed coordinate system). The ICLLM uses the model of de Leva [1] to calculate the subject specific anthropometry, and an Excel file is provided to ease calculation of these parameters.

calib_data.txt	1	54				
	107.0	237.8	39.1	212.0	178.3	.....

This gives the raw data pertaining to the positions of each marker when the subject is standing in the anatomical position (calibration position). The format of the file is the same as the position_data.txt file. The position (within the matrix) of the anatomical landmarks to be used to create the LCSs is specified in the landmarks file (landmarks.txt). For the lower body model (ICLLM) provided as part of FreeBody, the order of the markers (which are described in Table 2.1) within the file should be as follows: FCC, FMT, FM2, TF, FAM, TAM, C1, C2, C3, FLE, FME, T1, T2, T3, RASIS, LASIS, RPSIS, LPSIS.

m01_01_map_coord.cpp

This routine maps the input raw data from the laboratory coordinate system to the model coordinate system using the schema defined in the coord_map.txt file (as described underneath the description of the coord_map.txt file).

m01_03_save_raw_positions.cpp

This routine saves the input raw data and calibration data to the `Segment **segment_data[]` and `Structure *calibrate_pos[]` objects, respectively.

Module 2: Model Construction (m02_00_create_segments.cpp)

The purpose of this module is to create the system of rigid linked segments that describe the model. In particular, the calibration data is used to describe a local coordinate system (LCS) for each of the segments of the model. This is then used to find a transformation (rotation and translation) from the LCS of each segment to its position in each of the frames of the input data.

Inputs

- Positions of all markers saved in `Segment **segment_data[]`
- Positions of all markers with the subject standing in the anatomical position saved in `Structure *calibrate_pos[]`

Processes

Create rigid segments representing the model. Specify the position and orientation of each segment in each frame by finding the transformation from the LCS of the segment to its position in the laboratory fixed GCS.

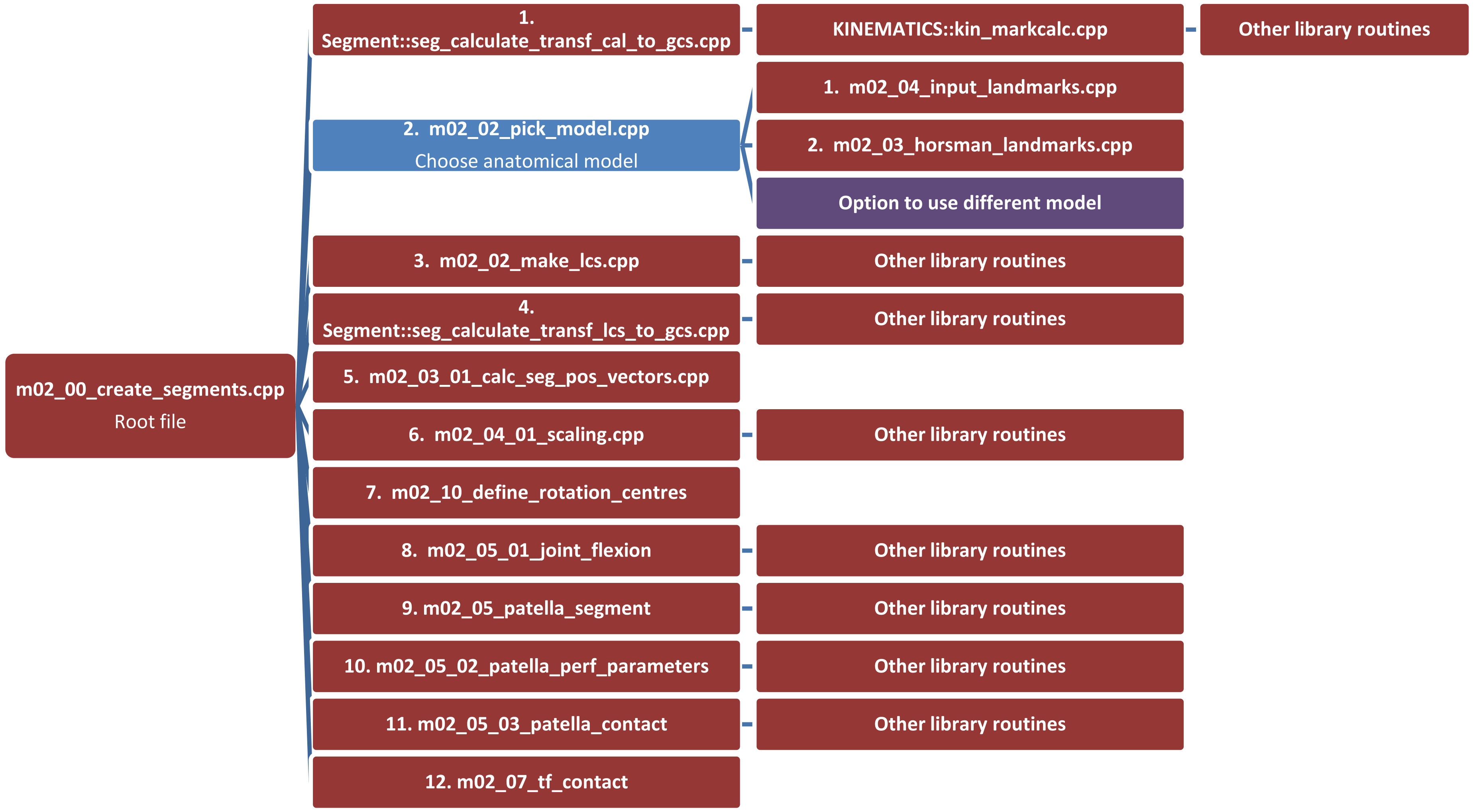
In particular, firstly find the transformation of each segment from the calibration position to the position of the segment in the GCS for each frame (Segment::seg_calculate_transf_cal_to_gcs.cpp). Then find the transformation of each segment from the LCS to the calibration position (m02_02_make_lcs.cpp). Finally, combine these two to find the transformation from LCS to GCS (Segment::seg_calculate_transf_lcs_to_gcs.cpp). This transformation can then be used to find the position of any point in the LCS in the GCS for any particular frame (Segment::seg_pos_point_on_seg.cpp).

In addition, and as part of the above, this module specifies the position of the patellar segment, and the tibiofemoral joint reaction forces.

Outputs

- Positions and orientations of each segment saved in `Segment **segment_data[]`
    - `segment_data[][]->origin`
    - `segment_data[][]->rot_lcs_to_gcs`

Module Overview



m02_00_create_segments.cpp
Root file
1. Segment::seg_calculate_transf_cal_to_gcs.cpp
KINEMATICS::kin_markcalc.cpp
Other library routines
2. m02_02_pick_model.cpp
Choose anatomical model
1. m02_04_input_landmarks.cpp
2. m02_03_horsman_landmarks.cpp
Option to use different model
3. m02_02_make_lcs.cpp
Other library routines
4. Segment::seg_calculate_transf_lcs_to_gcs.cpp
Other library routines
5. m02_03_01_calc_seg_pos_vectors.cpp
6. m02_04_01_scaling.cpp
Other library routines
7. m02_10_define_rotation_centres
8. m02_05_01_joint_flexion
Other library routines
9. m02_05_patella_segment
Other library routines
10. m02_05_02_patella_perf_parameters
Other library routines
11. m02_05_03_patella_contact
Other library routines
12. m02_07_tf_contact

Selected Detailed Software Description

m02_04_input_landmarks.cpp

This routine specifies the location of the calibration anatomical landmarks within the calibration data.

Input File	Example File Content

landmarks.txt	5 12
	-1 -1 -1 -1 -1 -1 0 2 1 0 1 1
	-1 -1 -1 -1 -1 -1 -1 -1 2 0 2 1
	-1 -1 -1 -1 -1 -1 -1 -1 2 0 2 1
	3 0 3 2 -1 -1 -1 -1 3 0 3 1
	-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

This file describes the location of the landmarks for the Klein Horsman muscle model within the calibration raw data file. The first two figures specify the size of the matrix that follows. Following this, each row represents a segment starting from segment 0. Each row can be split into 3 sections containing four numbers. Each of these segments refers to one axis of the segment in the order x, y, z. Within each segment, the first two numbers refers to the most positive end of that axis, and the second two numbers refers to the most negative end of that axis.

The two numbers representing the markers are used in the following way. If the number is minus 1, then there is no anatomical marker that is used for the purpose of defining that end of the axis. Otherwise, the first number indicates the segment on which the marker lies. The second number gives the number of the marker on that segment (where the markers are numbered in the order they are given within the raw data file).

Thus for the example given here, for the foot segment, there are three anatomical landmarks that are used. That is, a marker defining the most negative end of the y axis (the third marker of the foot segment) and markers defining the z axis (the medial marker being the first marker of the calf segment and the lateral marker being the second marker of the calf segment – note that in the Klein Horsman model, the positive z axis points from medial to lateral).

m02_03_horsman_landmarks.cpp

Calculate the longitudinal limits of each segment based upon the inputted landmarks, the Klein Horsman muscle model and the assumptions as to joint centres. Define the origins of each segment as follows: pelvis origin is the midpoint of ASISs; thigh origin is the hip joint centre; calf origin is the knee joint centre; and foot origin is the ankle centre.

m02_02_make_lcs.cpp

This routine finds the vectors that represent the LCS of each segment. Once these vectors have been found, they are then used to calculate the rotation from the LCS to the calibration position of the subject.

make_segments.txt	5 2
	0 2
	0 2
	0 2
	1 0
	1 2

This file specifies the instructions for creating each segment. The first row gives the size of the matrix that follows. Each following row represents one segment, starting from segment 0. The two numbers describe the order in which axes should be made. So in the example given, segment 0 is created by first finding the x axis, then the z axis and finally the y axis.

The landmarks to be used to create each axis have already been specified. These are then used to create the intermediate unit vectors representing each axis. The axes are then created as follows. The first axis is created by taking the cross product of the intermediate vectors representing the other two axes. Next, the second axis to be created is found by taking the cross product of the other two axes. The final axis is defined to be the intermediate vector that was used to represent it in the former calculations. The cross products of axes are calculated using the following conventions:

$$x = y \times z$$

$$y = z \times x$$

$$z = x \times y$$

m02_03_01_calc_seg_pos_vectors.cpp

This routine finds the positions of the COM and distal point on each segment in the GCS for each frame. It also calculates the vector from the origin to the distal point for later use in the inverse dynamics calculations (also in the GCS).

m02_04_01_scaling.cpp

This routine calculates the parameters that will be used to scale the Klein Horsman data set to the subject's specific geometry. The first operation performed here is to create LCSs of the Klein Horsman segments, using the same methodology as for the subject. This requires the location of the same anatomical landmarks in the Klein Horsman data set. These are specified in the file horsman_landmarks.txt:

horsman_landmarks.txt	15	6				
	-1	-1	-1	-1	-1	-1
	0.079	-0.804	0.028	0.198	-0.908	0.013
	0.045	-0.816	0.046	0.112	-0.792	0.010
	-1	-1	-1	-1	-1	-1
	0.054	-0.402	0.011	0.079	-0.804	0.028
	0.032	-0.400	0.055	0.077	-0.405	-0.032
	-1	-1	-1	-1	-1	-1
	0	0	0	0.054	-0.402	0.011
	0.032	-0.400	0.055	0.077	-0.405	-0.032
	0.038	0.088	0.042	-0.113	0.086	-0.045
	-1	-1	-1	-1	-1	-1
	0.038	0.088	0.042	0.038	0.088	-0.221
	-1	-1	-1	-1	-1	-1
	-1	-1	-1	-1	-1	-1
	-1	-1	-1	-1	-1	-1

The first two numbers give the size of the matrix that follows. In the following matrix, each group of 3 rows represents a segment (starting from segment 0) and each row represents an axis (first the x axis, then the y axis, then the z axis). On each row there are six figures – representing two groups of three figures – that is the coordinates of up to two anatomical landmarks which are used to define that axis). On each row, the first set of coordinates represents the most positive landmark on that axis (so for instance, for the x axis, the most anterior landmark). Where there are groups of three -1s, these indicate that there is no landmark specified.

The instructions for creating each LCS are given in the file horsman_instructions.txt:

horsman_instructions.txt	5 2
	0 2
	0 2
	0 2
	1 0
	1 2

The format of this file is exactly the same as the file make_segments.txt. The Klein Horsman segments are created based upon these instructions in the same way as the LCSs of the subject, using the Klein Horsman landmarks.

This file also calculates the length scaling factors based on the comparison of the dimensions of the Klein Horsman segments to the subject's segments.

m02_10_define_rotation_centres.cpp

This routine defines the centre of rotations of the joints. These are taken to be the points on each segment at which the joint reaction forces act. The routine also calculates the vectors from the centre of rotation of the segments to the positions at which the joint reaction forces act. The routine uses the data from the model_tree_structure.txt file (described earlier) and the rotation_centres.txt file:

rotation_centres.txt	5	3	
	0.01478	-0.0096	0.00343
	-0.0159	-0.0055	0.00248
	0	0	0
	0	0	0
	0	0	0

The file gives the centre of rotation of the proximal joint of each segment in the Klein Horsman LCS of the segment. Some of the joint centres of rotation are undefined in this file – this information is found in the model_tree_structure.txt file.

m02_05_01_joint_flexion.cpp

This routine finds the joint angles between each segment. This is based upon the definition of all joint angles being zero when the subject is stood in the calibration position. A quaternion is then found which represents the relative rotation of the distal segment relative to the proximal segment from this calibration position. Finally, this quaternion is decomposed into three Euler angles representing the joint angles. The order of rotation (about the body fixed axes of the distal segment) is first about the x axis, then the y axis and then finally the z axis.

m02_05_patella_segment.cpp

This routine calculates the position of the patella segment based on the file patella_parameters.txt:

patella_parameters.txt	9	6				
	0.06	0.02	0.001	0	0	0
	0.036	-0.056	0.009	0	0	0
	-0.001	0.05	0.0006	-0.001	0.047	0.0006
	20.4	-0.26	0	0	0	0
	10.88	-0.23	0.0019	-0.000006	0	0
	0	0.15	-0.0019	0.000007	0	0
	5.59	0.66	0	0	0	0
	1.63	0.067	0.00014	-0.000005	0	0
	1.43	0.106	-0.00345	0.00005	-0.0000002	0

This file contains the parameters used to create the patellofemoral joint model. The first line specifies the size of the matrix that follows. Following this, the numbers in the 9x6 matrix represent the following:

(1,1):	length of the patellar tendon (in the Klein Horsman data)
(1,2):	the thickness of the patella
(1,3):	radius of quadriceps tendon wrapping cylinder
(1,4) – (1,6):	3 dummy zeros
(2,1) – (2,3):	position of patellar tendon insertion on the tibia in the tibial local frame
(2,4) – (2,6):	3 dummy zeros
(3,1) – (3,3):	position of quadriceps tendon insertion on the patella in the patellar local frame
(3,4) – (3,6):	position of the most anterior part of the quadriceps tendon wrapping cylinder in the patellar local frame in the anatomical position
Rows 4-9:	specify the regression relationships between the knee flexion angle and the following parameters:

- Row 4: patellar tendon sagittal plane angle (a_0, a_1, then 4 dummy zeros)
- Row 5: patellar tendon coronal plane angle (b_0 = 10.88, b_1 = -0.233, b_2 = 0.00189, b_3 = -0.00000569, then two dummy zeros)
- Row 6: patellar tendon strain (c_0 = 0, c_1 = 0.153, c_2 = -0.00186, c_3 = 0.00000749, then two dummy zeros)
- Row 7: patellar flexion (d_0 = 5.59, d_1 = 0.6601, then 4 dummy zeros)
- Row 8: patellar tilt (e_0 = 1.628, e_1 = 0.0667, e_2 = 0.000144, e_3 = -0.00000537, then 2 dummy zeros)
- Row 9: patellar rotation (f_0 = 1.427, f_1 = 0.1056, f_2 = -0.00345, f_3 = 0.000054656, f_4 = -0.00000023756, then 1 dummy zero)

m02_05_02_patella_perf_parameters.cpp

This routine finds the pivot point of the patella based upon the maintenance of force and moment equilibrium of the patella in the sagittal plane.

m02_05_03_patella_contact.cpp

This routine finds the contact point of the patella on the femur.

m02_07_tf_contact.cpp

This routine finds the contact positions for the medial and lateral compartments of the tibiofemoral joint using the data from tf_contact.txt:

tf_contact.txt	2	3	
	-0.0251	-0.0044	0.0202
	-0.0067	-0.0066	-0.0152

This file gives the contact positions of the medial and lateral aspects of the tibiofemoral joint on the tibial segment in the Klein Horsman frame.

Module 3: Calculate Model Kinematics (m03_00_calculate_kinematics.cpp)

The purpose of this module is to calculate the kinematics of each segment based upon the changes in the position and orientation data of each segment. These calculations rely heavily upon the kinematic formalism described by Dumas and colleagues[2], and upon quaternion algebra. This can be found on pages 163-164 of the article:

Dumas, R., Aissaoui, R. & De Guise, J. A. (2004). A 3D generic inverse dynamics method using wrench notation and quaternion algebra. *Computer Methods in Biomechanics and Biomedical Engineering, 7,* 159-166.

Inputs

- Positions and orientations of each segment saved in `Segment **segment_data[]`
- Anthropometry of each segment saved in `Segment **segment_data[]`

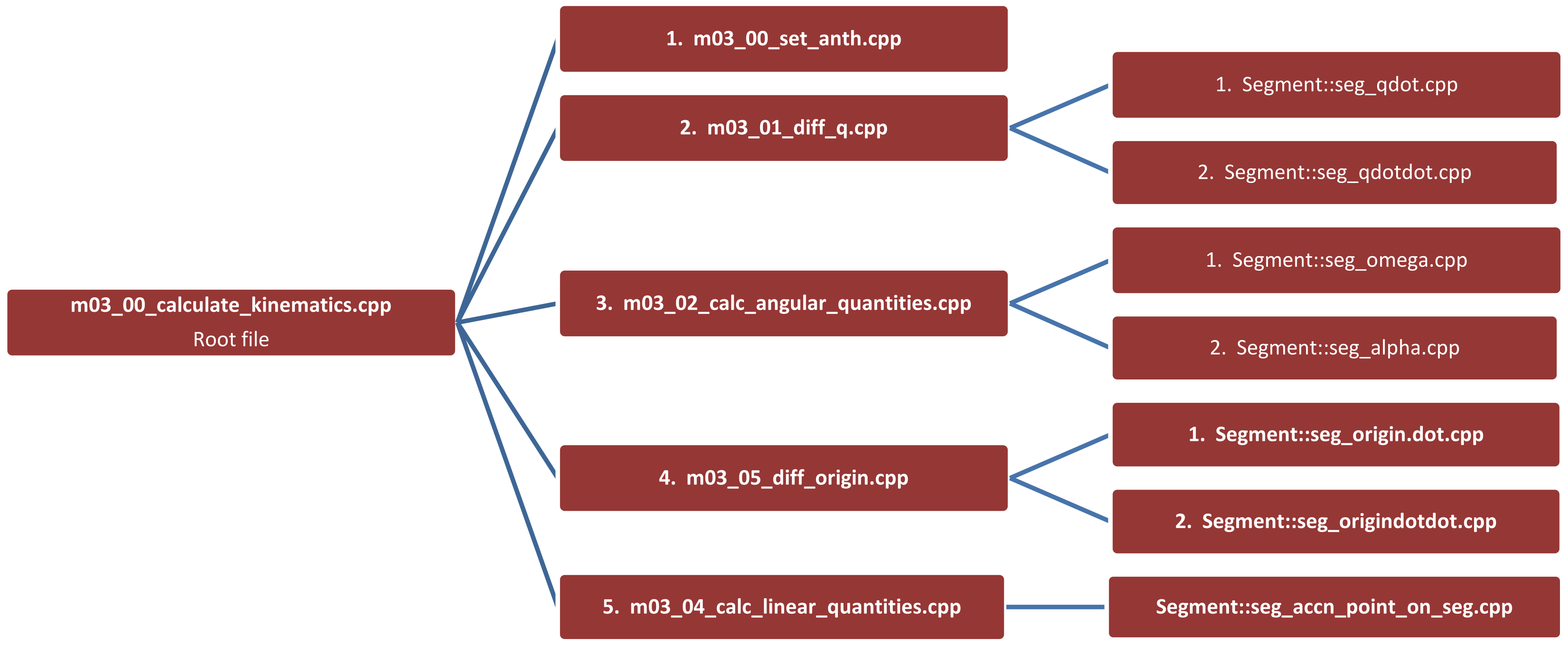
Processes

Calculation of kinematics from position and orientation data (by consideration of the changes in position and orientation over time).

Outputs

- Kinematic variables for each segment saved in `Segment **segment_data[]`
    - `segment_data[][]->omega`
    - `segment_data[][]->alpha`
    - `segment_data[][]->joint_angles`
    - `etc.`

Module Overview



m03_00_calculate_kinematics.cpp
Root file
1. m03_00_set_anth.cpp
2. m03_01_diff_q.cpp
3. m03_02_calc_angular_quantities.cpp
4. m03_05_diff_origin.cpp
5. m03_04_calc_linear_quantities.cpp
1. Segment::seg_qdot.cpp
2. Segment::seg_qdotdot.cpp
1. Segment::seg_omega.cpp
2. Segment::seg_alpha.cpp
1. Segment::seg_origin.dot.cpp
2. Segment::seg_origindotdot.cpp
Segment::seg_accn_point_on_seg.cpp

Selected Detailed Software Description

m03_00_set_anth.cpp

Takes the anthropometry of each segment from the calibration `Structure *calibrate_pos[]` object and saves it in the `Segment **segment_data[]` object for each segment and each frame.

m03_01_diff_q.cpp

Calculates the first and second derivatives of the quaternion representing the orientation of each segment, and saves it in `Segment **segment_data[]` for each segment and each frame.

m03_02_calc_angular_quantities.cpp

Calculates the angular velocity and acceleration of each segment and saves it in `Segment **segment_data[]` for each segment and each frame.

m03_05_diff_origin.cpp

Calculates the first and second derivatives of the vector specifying the origin of each segment, and saves it in `Segment **segment_data[]` for each segment and each frame.

m03_04_calc_linear_quantities.cpp

Calculates the linear acceleration of the COM of each segment for each frame and saves it in `Segment **segment_data[]`.

Note that the above routines make extensive use of the functions of the `Segment` class.

Module 4: Calculate Segment Kinetics (m04_00_inverse_dynamics.cpp)

The purpose of this module is to perform an inverse dynamics analysis using a method based upon the work of Dumas and colleagues[2]:

Dumas, R., Aissaoui, R. & De Guise, J. A. (2004). A 3D generic inverse dynamics method using wrench notation and quaternion algebra. *Computer Methods in Biomechanics and Biomedical Engineering, 7,* 159-166.

In the first place, this analysis produces the inter-segmental forces and moments based entirely on the work of Dumas et al. The routine also calculates all known inter-segmental parameters for the equations of motion that need to be solved in an optimization based approach to the inverse dynamics[3–6]. These calculations are described in more detail in the publications below:

Cleather, D. J., & Bull, A. M. J. (2015). The development of a segment-based musculoskeletal model of the lower limb: Introducing FreeBody. *Royal Society Open Science, in press.* doi:

Cleather, D. J., & Bull, A. M. J. (2011). An optimization-based simultaneous approach to the determination of muscular, ligamentous, and joint contact forces provides insight into musculoligamentous interaction. *Annals of Biomedical Engineering, 39,* 1925-1934. doi: 10.1007/s10439-011-0303-8

Cleather, D. J., Goodwin, J. E., & Bull, A. M. J. (2011). An optimization approach to inverse dynamics provides insight as to the function of the biarticular muscles during vertical jumping. *Annals of Biomedical Engineering, 39*, 147-160. doi: 10.1007/s10439-010-0161-9

Cleather, D. J., Goodwin, J. E., & Bull, A. M. J. (2011). Erratum to: An optimization approach to inverse dynamics provides insight as to the function of the biarticular muscles during vertical jumping. *Annals of Biomedical Engineering, 39,* 2476-2478. doi: 10.1007/s10439-011-0340-3

Inputs

- Kinematic data and external forces saved in `Segment **segment_data[]`
- Anthropometry of each segment saved in `Segment **segment_data[]`

Processes

Solution of wrench based equations of motion described in Dumas et al.[2]

Outputs

- Inter-segmental forces and moments saved in `Segment **segment_data[]`
  - `segment_data[][]->fd[]`
  - `segment_data[][]->fp[]`
  - `segment_data[][]->md_segmental_approach[]`

  - `segment_data[][]->mp_segmental_approach[]`
- Parameters for optimization based inverse dynamics analysis saved in `Segment **segment_data[]`
  - `segment_data[][]->rhs_opt_approach_2[]`

Selected Detailed Software Description

m04_00_inverse dynamics.cpp

Simply loops sequentially through each segment and frame, calling the routine that solves the wrench based equations of motion. Iterates from the distal to proximal segment, establishing the force and moment acting on the distal end of the segment by using the proximal values of the previous (more distal) segment and a consideration of Newton’s third law.

Segment::seg_solve_wrench_eqns.cpp

Calculates the wrench based equations of motion for the given segment.

Module 5: Calculate Muscle Geometry (m05_00_muscle_model.cpp)

The purpose of this module is to calculate the musculoskeletal geometry for each frame. The muscle and ligament models used here rely heavily on the work of Klein Horsman and colleages[7].

Inputs

- Positions and orientations of each segment saved in `Segment **segment_data[]`
  - `segment_data[][]->origin`
  - `segment_data[][]->rot_lcs_to_gcs`
- Parameters of muscle and ligament models stored in the text files:
  - horsman_muscle_model.txt
  - wrapping_cylinders.txt
  - ligament_model.txt

Processes

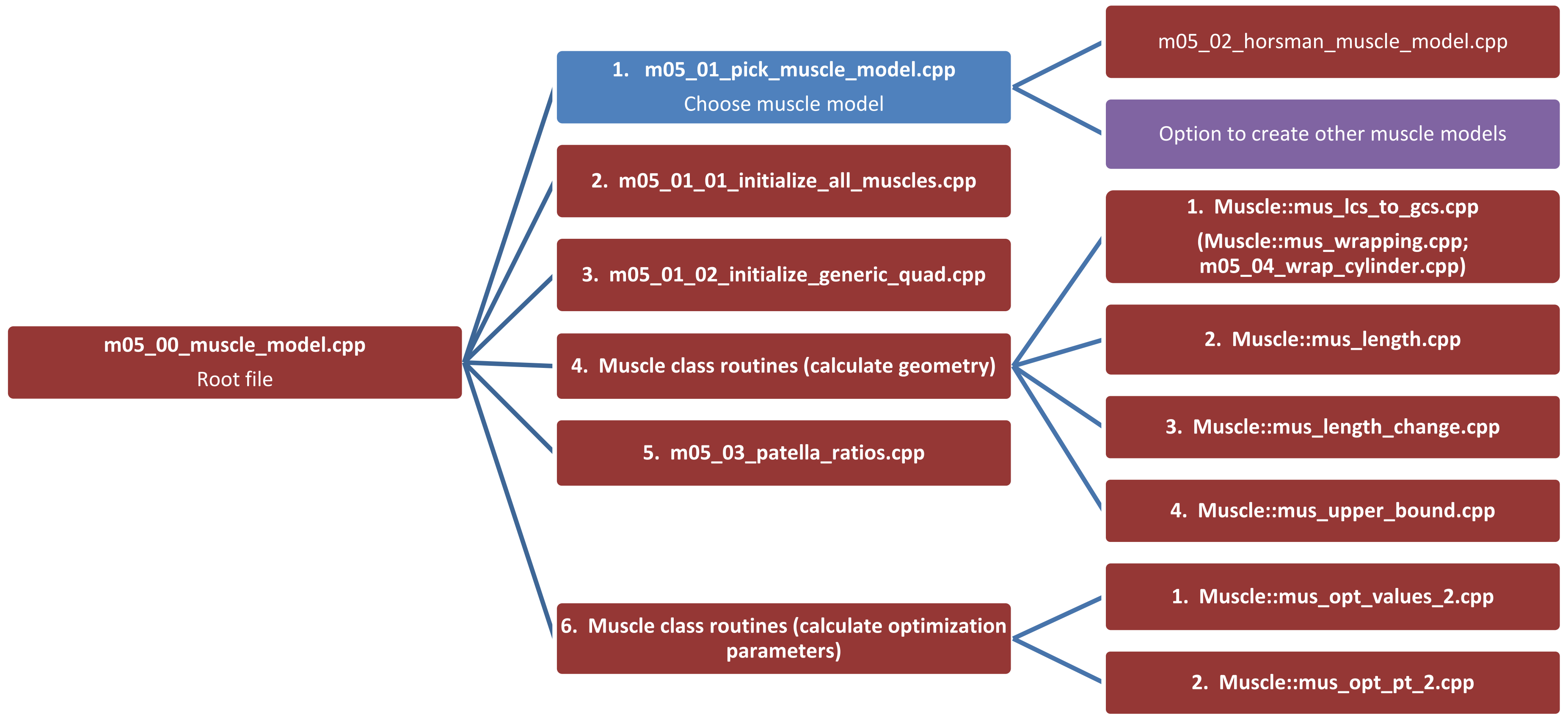
Calculate the position of all muscle and ligament points (i.e. the paths of muscles and ligaments) in the global coordinate system for all frames, based upon their position in the local frame (taken from the Klein Horsman data set).

Calculate the parameters to be used in the optimization calculation based upon the musculoskeletal geometry of the model.

Outputs

- Muscle and ligament paths saved in `Muscle **muscle_data[]` and `Muscle **ligament_data[]`
  - `muscle_data[][]->gblpnts`
  - `ligament_data[][]->gblpnts`
- Optimization parameters saved in `Muscle **muscle_data[]`, `Muscle **ligament_data[]` and `Muscle *patellar_tendon[]`
  - `muscle_data[][]->opt_2`
  - `muscle_data[][]->max_force`
  - `muscle_data[][]->pq_ratio`
  - `ligament_data[][]->opt_2`
  - `ligament_data[][]->max_force`
  - `patellar_tendon[]->opt_2`

Module Overview


m05_00_muscle_model.cpp
Root file
1. m05_01_pick_muscle_model.cpp
Choose muscle model
2. m05_01_01_initialize_all_muscles.cpp
3. m05_01_02_initialize_generic_quad.cpp
4. Muscle class routines (calculate geometry)
5. m05_03_patella_ratios.cpp
6. Muscle class routines (calculate optimization parameters)
m05_02_horsman_muscle_model.cpp
Option to create other muscle models
1. Muscle::mus_lcs_to_gcs.cpp
(Muscle::mus_wrapping.cpp; m05_04_wrap_cylinder.cpp)
2. Muscle::mus_length.cpp
3. Muscle::mus_length_change.cpp
4. Muscle::mus_upper_bound.cpp
1. Muscle::mus_opt_values_2.cpp
2. Muscle::mus_opt_pt_2.cpp

Selected Detailed Software Description

m05_02_horsman_muscle_model.cpp

This routine inputs the data for the muscle and ligaments (muscle objects are used to store both muscle and ligament data, and this routine is run twice – once with the muscle data and once with the ligament data). It also inputs the wrapping data (described below). The format of the input files is as follows:

Input File	**Example File Content**

Model Details

horsman_muscle_model.txt	add_b_p1 2 2 0.0086 -0.1008 0.0205 3 0 0 -0.0185 -0.0834 0.0181 2 0 0 0.00019 2 313900 0.13962634 add_b_p2 2 2 ... ...

ligament_model.txt	ilfem_a 2 2 -0.018 -0.057 0.1007 3 0 0 -0.022 -0.037 0.0120 2 0 0 0.00135 2 313900 0.13962634 ilfem_l 2 2 ... ...

These files specify the detail of each muscle and ligament element given in the model. For instance, for each muscle, firstly the name of the muscle is specified. Next the number of points specifying the path of that muscle is given (e.g. origin and insertion and any via points) and then the number of segments spanned by the muscle. Following this, each subsequent line gives the detail for one of the muscle points, starting at the most proximal end and working distally. On each of these lines, the first three numbers specifies the location of that point within the LCS of the segment upon which it is located. The next number indicates which segment the point is located on. The next two numbers are flags that indicate whether there is any wrapping, and if so which cylinder to use. Finally, the last line gives some parameters relating to the architecture of the particular muscle (PCSA, PCSA multiple, maximum tension, dummy).

m05_01_01_initialize_all_muscles.cpp

This routine initializes all of the muscle objects that will be used (muscle objects are used to store both muscle and ligament data, and this routine is run twice – once with the muscle data and once with the ligament data). The objects describing each muscle and ligament are parameterized from the above input files, and then used to store the location of muscle and ligament paths in the GCS for each frame.

m05_03_patella_ratios.cpp

Calculates the P/Q ratio for each quadriceps muscle element.

m05_04_wrap_cylinder.cpp

This routine calculates the wrapping of a muscle about its wrapping cylinder (if one is specified), using the method of Charlton and Johnson[8]. The parameters of the wrapping cylinders are defined in wrapping_cylinders.txt (and have been previously input and stored as part of the routine m05_02_horsman_muscle_model.cpp):

wrapping_cylinders.txt	3	8						
	0.061	-0.402	-0.018	-0.37	0.04	0.93	0.025	-1
	-0.050	-0.078	0.072	-0.21	0.14	0.79	0.039	1
	0	0	0	-0.46	0.055	0.886	0.001	1

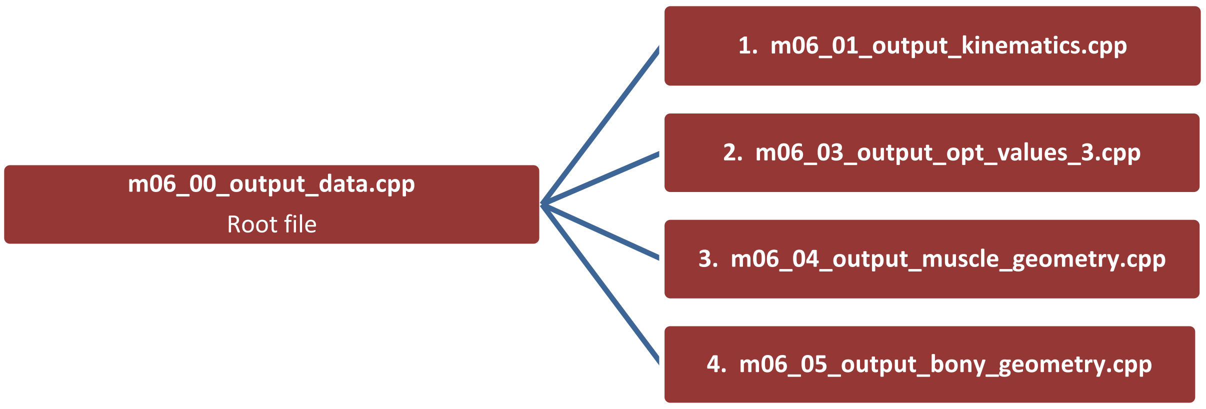
The first two numbers specify the size of the matrix that follows. Then, each row of the following matrix describes a wrapping cylinder. The muscles that use a wrapping cylinder specify which one by giving the row number (as described in the description of the horsman_muscle_model.txt file). On each row, the first triple gives the location of a point on the central axis of the wrapping cylinder and the second triple gives a vector along this axis. This vector should be orientated laterally. The seventh number gives the radius of the wrapping cylinder. Finally, the eighth number specifies the direction the muscle should wrap about the cylinder. That is, if observing the model from the right hand side (lateral side) then -1 represents an anti-clockwise wrapping and 1 a clockwise wrapping.

Note that, for the quadriceps, the wrapping point is defined during the creation of the patellofemoral joint model.

Module 6: Output Calculation Results (m06_00_output_data.cpp)

This module simply outputs the results of the preceding calculations as text files that can then be used by the MATLAB® routine that follows.

Module Overview


m06_00_output_data.cpp
Root file
1. m06_01_output_kinematics.cpp
2. m06_03_output_opt_values_3.cpp
3. m06_04_output_muscle_geometry.cpp
4. m06_05_output_bony_geometry.cpp

Selected Detailed Software Description

m06_01_output_kinematics.cpp

This routine outputs the joint angles in degrees, in the text file joint_angles.txt. The output is a matrix, with each row representing a frame. The first 3 columns give the ankle joint angles, the next 3 the knee, etc. Within each set of 3 angles, the first gives the ab/adduction angle, the second gives the internal/external rotation angle, and the third gives the flexion/extension angle.

m06_03_output_opt_values.cpp

This routine outputs the values that are used within MATLAB® to perform the optimization. These files are described in the next section.

m06_04_output_muscle_geometry.cpp

This routine outputs the path of each muscle for each frame. These muscle paths can be visualized within the FreeBody MATLAB® app. There is a file for each muscle element (muscle_path0.txt to muscle_path162.txt). Within each file, each row represents one frame. Each row consists of a given number of triplets representing the coordinates of each muscle point for that muscle.

m06_05_output_bony_geometry.cpp

This routine outputs the location of some geometrical landmarks:

origins.txt	- position of segment origins
patella.txt	- position of patella and patellar tendon
rot_centres_gcs.txt	- centres of rotation of the joints
tf_contact_gcs.txt	- tibiofemoral joint contact points.

The routine also outputs the scaling factors that have been used for each segment as scaling_factors.txt.

Module 7: Optimization (in MATLAB®)

Inputs

MATLAB® uses the following text files that are output from module 6. The files are colour coded to match the equations of motion below where they are used:

muscle_force_seg00.txt – muscle_force_seg22.txt
muscle_moment_seg00.txt – muscle_moment_seg22.txt
ligament_force_seg00.txt – ligament_force_seg22.txt
ligament_moment_seg00.txt – ligament_moment_seg22.txt
muscle_force_seg40.txt – muscle_force_seg42.txt
pat_tendon.txt
rhs.txt
force_ub.txt
ligament_ub.txt
ratios.txt
joint_contacts.txt

Processes

The optimization finds a solution to the following equations of motion for each frame:

$$\left(\begin{array}{ccccccccccccc}
\hat{p}_1^1 & \cdots & \hat{p}_M^1 & \hat{p}_{pt}^1 & \hat{q}_1^1 & \cdots & \hat{q}_N^1 & -I_{3\times3} & E_{3\times3} & E_{3\times3} & E_{3\times3} & E_{3\times3} \\
\hat{p}_1^2 & \cdots & \hat{p}_M^2 & \hat{p}_{pt}^2 & \hat{q}_1^2 & \cdots & \hat{q}_N^2 & I_{3\times3} & -I_{3\times3} & -I_{3\times3} & E_{3\times3} & E_{3\times3} \\
\hat{p}_1^3 & \cdots & \hat{p}_M^3 & \hat{p}_{pt}^3 & \hat{q}_1^3 & \cdots & \hat{q}_N^3 & E_{3\times3} & I_{3\times3} & I_{3\times3} & -I_{3\times3} & I_{3\times3} \\
\hat{r}_1^1 \times \hat{p}_1^1 & \cdots & \hat{r}_M^1 \times \hat{p}_M^1 & \hat{r}_{pt}^1 \times \hat{p}_{pt}^1 & \hat{s}_1^1 \times \hat{q}_1^1 & \cdots & \hat{s}_N^1 \times \hat{q}_N^1 & E_{3\times3} & E_{3\times3} & E_{3\times3} & E_{3\times3} & E_{3\times3} \\
\hat{r}_1^2 \times \hat{p}_1^2 & \cdots & \hat{r}_M^2 \times \hat{p}_M^2 & \hat{r}_{pt}^2 \times \hat{p}_{pt}^2 & \hat{s}_1^2 \times \hat{q}_1^2 & \cdots & \hat{s}_N^2 \times \hat{q}_N^2 & \tilde{d}^2 & -\tilde{h}_1^2 & -\tilde{h}_2^2 & E_{3\times3} & E_{3\times3} \\
\hat{r}_1^3 \times \hat{p}_1^3 & \cdots & \hat{r}_M^3 \times \hat{p}_M^3 & \hat{r}_{pt}^3 \times \hat{p}_{pt}^3 & \hat{s}_1^3 \times \hat{q}_1^3 & \cdots & s_N^3 \times \hat{q}_N^3 & E_{3\times3} & \tilde{d}_1^3 & \tilde{d}_2^3 & E_{3\times3} & \tilde{f}^3 \\
\hat{p}_1^{pat} & \cdots & \hat{p}_M^{pat} & \hat{p}_{pt}^{pat} & & E_{3\times N} & & E_{3\times3} & E_{3\times3} & E_{3\times3} & E_{3\times3} & -I_{3\times3} \\
\rho_1 & \cdots & \rho_M & -1 & & E_{1\times N} & & E_{1\times3} & E_{1\times3} & E_{1\times3} & E_{1\times3} & E_{1\times3}
\end{array}\right)
\begin{pmatrix} F_1 \\ \vdots \\ F_M \\ F_{pt} \\ L_1 \\ \vdots \\ L_N \\ \hat{R}^1 \\ \hat{R}_1^2 \\ \hat{R}_2^2 \\ \hat{R}^3 \\ \hat{R}^{pat} \end{pmatrix}$$

$$= \begin{pmatrix}
m^1(\hat{a}^1 - \hat{g}) - \hat{S}^0 \\
m^2(\hat{a}^2 - \hat{g}) \\
m^3(\hat{a}^3 - \hat{g}) \\
m^1\hat{c}^1 \times (\hat{a}^1 - \hat{g}) + Y_{3\times3}^1 \ddot{\hat{\varphi}}^1 + \dot{\hat{\varphi}}^1 \times Y_{3\times3}^1 \dot{\hat{\varphi}}^1 - \hat{d}^1 \times \hat{S}^0 - M^0 \\
m^2\hat{c}^2 \times (\hat{a}^2 - \hat{g}) + Y_{3\times3}^2 \ddot{\hat{\varphi}}^2 + \dot{\hat{\varphi}}^2 \times Y_{3\times3}^2 \dot{\hat{\varphi}}^2 \\
m^3\hat{c}^3 \times (\hat{a}^3 - \hat{g}) + Y_{3\times3}^3 \ddot{\hat{\varphi}}^3 + \dot{\hat{\varphi}}^3 \times Y_{3\times3}^3 \dot{\hat{\varphi}}^3 \\
E_{3\times1} \\
0
\end{pmatrix}$$

Such that the following cost function is optimized:

$$\min_{F_i,\ L_j} J = \sum_{i=1}^{M} \left(\frac{F_i}{Fmax_i}\right)^3 + \sum_{j=1}^{N} \left(\frac{L_i}{Lmax_i}\right)^3$$

And where:

$\hat{a}^k$	linear acceleration of the centre of mass of segment *k*
$\hat{c}^k$	vector from centre of rotation of joint at proximal end of segment *k* to centre of mass of segment *k*
$\hat{d}^k$	vector from centre of rotation of joint at proximal end of segment *k* to centre of rotation joint at distal end of segment *k*
$\tilde{d}^k$	skew-symmetric matrix of vector $\tilde{d}^k$
$\tilde{d}_l^3$	skew-symmetric matrix of vector from centre of rotation of hip to tibiofemoral joint contact *l*
$E_{3\times3}$	3×3 matrix of zeros
$\tilde{f}^3$	skew-symmetric matrix of vector from centre of rotation of hip to contact point of patella with the femur
F_i	magnitude of force in muscle *i*
$Fmax_i$	maximum possible force in muscle *i* (upper bound)
$\hat{g}$	acceleration due to gravity
$\tilde{h}_l^2$	skew-symmetric matrix of vector from centre of rotation of knee to tibiofemoral joint contact *l*
i	muscle number
$I_{3\times3}$	3×3 identity matrix
j	ligament number
J	cost function
k	segment number
L_j	magnitude of force in ligament *j*
$Lmax_j$	maximum possible force in ligament *j* (upper bound)

m^k	mass of segment *k*
M	total number of muscles
N	total number of ligaments
$\hat{p}_i^k$	unit vector representing the line of action of force created by muscle *i* that acts on segment *k* (zero if muscle does not insert on segment *k*)
pat	patella
pt	patellar tendon
P/Q ratio	ratio of patellar tendon to quadriceps tendon force
$\hat{q}_j^k$	unit vector representing the line of action of force created by ligament *j* that acts on segment *k* (zero if ligament does not insert on segment *k*)
$\hat{r}_i^k$	vector from centre of rotation of joint at proximal end of segment *k* to point of action of muscle *i* on segment *k* (zero if muscle does not insert on segment *k*)
Rx^k	*x* component of reaction force acting at proximal end of segment *k*
Ry^k	*y* component of reaction force acting at proximal end of segment *k*
Rz^k	*z* component of reaction force acting at proximal end of segment *k*
$\hat{R}^k$	vector representing *x*, *y* and *z* components of reaction force acting at proximal end of segment *k*
$\hat{R}_l^k$	vector representing *x*, *y* and *z* components of reaction force *l* acting at proximal end of segment *k*
$\hat{s}_j^k$	vector from centre of rotation of joint at proximal end of segment *k* to point of action of ligament *j* on segment *k* (zero if ligament does not insert on segment *k*)
$-\hat{S}^k$	inter-segmental force acting on proximal end of segment *k*
$\hat{v}_i^k$	effective moment arm of muscle *i* on segment *k*
$\hat{w}_j^k$	effective moment arm of ligament *j* on segment *k*
$-\widehat{W}^k$	inter-segmental moment acting on proximal end of segment *k*
$Y_{3\times3}^k$	inertia tensor of segment *k*
ρ_i	P/Q ratio for muscle *i* (zero if the muscle is not part of the quadriceps muscle group)
$\dot{\varphi}^k$	angular velocity of segment *k*
$\ddot{\varphi}^k$	angular acceleration of segment *k*

Outputs

The routine outputs two main text files: force.txt and solve.txt. The file solve.txt is a record of the success of the model at finding a solution – it outputs the MATLAB® exit flag for the optimization calculation for each frame. The file force.txt outputs the vector that represents the solution of the equations of motion found by the optimization. In force.txt, each row represents one frame. The columns are in the same order as the solution vector in the equation above. The order of muscles and ligaments are the same as in the input files describing them.

References

1. De Leva, P. Adjustments to Zatsiorsky - Seluyanov's segment inertia parameters. *J. Biomech.* **29,** 1223–1230 (1996).
2. Dumas, R., Aissaoui, R. & de Guise, J. A. A 3D generic inverse dynamic method using wrench notation and quaternion algebra. *Comput. Methods Biomech. Biomed. Engin.* **7,** 159–166 (2004).
3. Cleather, D. J., Goodwin, J. E. & Bull, A. M. J. An Optimization Approach to Inverse Dynamics Provides Insight as to the Function of the Biarticular Muscles During Vertical Jumping. *Ann. Biomed. Eng.* **39,** 147–160 (2011).
4. Cleather, D. J., Goodwin, J. E. & Bull, A. M. J. Erratum to: An Optimization Approach to Inverse Dynamics Provides Insight as to the Function of the Biarticular Muscles During Vertical Jumping. *Ann. Biomed. Eng.* **39,** 2476–2478 (2011).
5. Cleather, D. J. & Bull, A. M. J. An Optimization-Based Simultaneous Approach to the Determination of Muscular, Ligamentous, and Joint Contact Forces Provides Insight into Musculoligamentous Interaction. *Ann. Biomed. Eng.* **39,** 1925–1934 (2011).
6. Cleather, D. J. & Bull, A. M. J. Introducing FreeBody: An open source musculoskeletal model of the lower limb. *R. Soc. Open Sci.* (In Press).
7. Klein Horsman, M. D., Koopman, H. F. J. M., van der Helm, F. C. T., Poliacu Prose, L. & Veeger, H. E. J. Morphological muscle and joint parameters for musculoskeletal modelling of the lower extremity. *Clin. Biomech.* **22,** 239–247 (2007).
8. Charlton, I. W. & Johnson, G. R. Application of spherical and cylindrical wrapping algorithms in a musculoskeletal model of the upper limb. *J. Biomech.* **34,** 1209–1216 (2001).