NTNU
Norwegian University of
Science and Technology

Knowledge Based Engineering for Human Body Modeling and Simulation

Martha Risnes

Mechanical Engineering
Submission date: June 2016
Supervisor: Bjørn Haugen, IPM

Norwegian University of Science and Technology
Department of Engineering Design and Materials

NTNU - NORWEGIAN UNIVERSITY
OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF ENGINEERING DESIGN
AND MATERIALS

MASTER THESIS SPRING 2016
FOR
STUD.TECHN. MARTHA RISNES

SPECIFY AND IMPLEMENT KBE FOR HUMAN BODY MODELING AND SIMULATIOM

Spesifisere og implementere KBE for modellering og simulering av menneskekroppen

The human body topology is usually equal for all individuals, but sizes and dimensions vary. Some differences are natural from birth, but some are born with handicaps. Variations may come from injuries, but may also be desired by athletes from training over time. The human body is a mechanical system suited for computer simulation.
The partners in this effort have comprehensive and complementary competences for building and simulating the human body:
TechnoSoft Inc. with 30 years research experience with automation in design (developing tools and applications for industry worldwide), a key knowledge for modelling variations in the human body.
NTNU. with 30 years research experience with developing simulation software for mechanical systems, especially mechanisms, based on the Finite Element method and control engineering.
This master assignment is based on the report from the candidate's preproject assignment conducted during the autumn 2015 and other recent work in this area.
Part of the work will be to investigate if this activity could be integrated with ongoing or new European projects on related problems

The assignment include:

1. Study the simulation program FEDEM as a potential tool for human body simulation

2. Investigate if there are running EU projects we could seek integration with or announcement of funding in this area that could have potential for a project proposal

3. Test existing AML code for human body modeling and mechanism modeling in general to decide if it has potential for further development based on the preprojects specifications

4. Specify software for modeling and simulation capabilities based on the conclusion from point 3 above

5. As far as time allow, implement and test AML code for human body modeling and simulation based on the specifications developed in point 4 above.

Formal requirements:

Three weeks after start of the thesis work, an A3 sheet illustrating the work is to be handed in. A template for this presentation is available on the IPM's web site under the menu "Masteroppgave" (https://www.ntnu.edu/web/ipm/master-thesis). This sheet should be updated one week before the master's thesis is submitted.

Risk assessment of experimental activities shall always be performed. Experimental work defined in the problem description shall be planed and risk assessed up-front and within 3 weeks after receiving the problem text. Any specific experimental activities which are not properly covered by the general risk assessment shall be particularly assessed before performing the experimental work. Risk assessments should be signed by the supervisor and copies shall be included in the appendix of the thesis.

The thesis should include the signed problem text, and be written as a research report with summary both in English and Norwegian, conclusion, literature references, table of contents, etc. During preparation of the text, the candidate should make efforts to create a well arranged and well written report. To ease the evaluation of the thesis, it is important to cross-reference text, tables and figures. For evaluation of the work a thorough discussion of results is appreciated.

The thesis shall be submitted electronically via DAIM, NTNU's system for Digital Archiving and Submission of Master's theses.

The contact person is:
Ole Ivar Sivertsen, IPM

[illegible]
for Torgeir Welo
Head of Division

Bjørn Haugen
Bjørn Haugen
Associated professor/Supervisor

NTNU
Norges teknisk-naturvitenskapelige universitet
Institutt for produktutvikling og materialer

Preface

This paper is based on the work done in the spring 2016 semester. The project is conducted in the Department of Engineering Design and Materials(IPM) of NTNU with Professor Bjørn Haugen as supervisor.

The thesis is continuation of the project thesis conducted the fall 2015 based on a literature study on modeling and simulations of the musculoskeletal system.

The author would like to to thank Professor Ole Ivar Sivertsen and Professor Bjørn Haugen for their time, support and discussions throughout the master thesis work.

Trondheim

June 2016

Martha Risnes

Abstract

This thesis has been written to demonstrate the development of a Knowledge-Based Engineering (KBE) application of the musculoskeletal system of the human body. One of the main goals has been to establish what advantages musculoskeletal modeling could have in a KBE modeling framework.

The model has been created in Adaptive Modeling Language (AML) developed by Technosoft to show how a multibody model of the human anatomy can be modeled in a finite element representation. Furthermore, the possibilities of using the AML models to perform simulations in the multibody software Fedem are investigated.

In summary, the goal of this thesis is to develop a conceptual AML model of the human skeleton for future development along with a potential framework for simulations. The challenge is to model the cross field of bio-mechanics and mechanical engineering, to discover possible alternative applications of AML that could benefit musculoskeletal modeling.

Sammendrag

Denne oppgaven er skrevet for å vise utviklingen av et Knowledge Based Engineering (KBE) applikasjon av en muskelskjelett-modell av menneskekroppen. Et av hovedmålene har vært å se hvilke fordeler muskelskjelett-modellering kan dra nytte av i et KBE modellerings-rammeverk.

Modellen har blitt laget ved bruk av Adaptive Modeling Language (AML) utviklet av Technosoft for å vise hvordan en multibodymodell av den menneskelige anatomi kan modelleres i en elementmetode-representasjon. Videre har mulighetene til å bruke AML-modellen for å foreta simuleringer i programvaren Fedem blitt undersøkt.

Oppsummert er målet med denne avhandlingen å utvikle en konseptuell AMLmodell av det menneskelige muskelskjelett-system for fremtidig utvikling, og fremstille et potensielt rammeverk for simuleringer. En av utfordringene er først og fremst å modellere et kryssfelt mellom biomekanikk og tradisjonell mekanikk. Motivasjonen for å gjøre dette er undersøke mulige alternative anvendelser av AML som kan være til nytte i muskel- og skjelettmodellering

Contents

List of Figures

Nomenclature

Musculoskeletal system	The muscular and skeletal system
AML	Adaptive Modeling Language
CAD	Computer Aided Design
CAE	Computer Aided Engineering
FEA	Finite Element Analysis
GUI	Graphical User Interface
KBE	Knowledge Based Engineering

Chapter 1

Introduction

1.1 Background

Musculoskeletal modeling is a challenge because of how the muscles and the skeleton move and function are for the time being not fully understood. The fact that every individual is different does not make it easier. Today's imaging technology makes it possible to learn more about the human body, but this is still expensive and time-consuming.

Modeling the musculoskeletal system is similar to the modeling methodology used in reverse engineering processes. Reverse engineering can be defined as the process of extracting knowledge about a design in order to understand it. This is a term used when analyzing mechanical structures or software. Because of the nature of reverse engineering, the knowledge gained is based on capturing knowledge in a non-destructive way, e.g. by using imaging techniques. Furthermore, the data captured can be translated into computer-aided models.

Knowledge-Based Engineering (KBE) is used to re-use knowledge about designs to generating CAD models to automate design processes. Hence, the KBE framework is a good starting point for creating a musculoskeletal model.

The objective of this thesis is to explore the possibilities of making a generic human muscle and bone model to be used in simulations in the software Fedem. Simulations performed in Fedem could potentially be used to contribute to understanding of clinical problems with bone and muscle diseases, rehabilitation and the design of medical devices such as prostheses.

In musculoskeletal modeling there are especially two domains used to carry out such simulations: finite element modeling simulations and rigid multibody simulations. Often there is a need to perform both types of simulations to model the situation accurately, due to the complexity of the materials and movements. This process can be time-consuming, because the different fields require different software and the data are in different formats.

For this reason there is a need to bridge the gap between the two domains, which would lead to more accurate modeling of the musculoskeletal system, and time saved when switching between the two domains.

Current and previous work on creating such models include an NIH-funded collaboration to make a foot model including all mass, not only muscles and bones [1]. The aim of this research is to create a complete finite element model to use in a full gait cycle. Other projects are demu2neck [2], which is a finite element model of the neck.

1.2 Research questions

RQ1: Can a finite element model created in a Knowledge Based Engineering (KBE) framework improve musculoskeletal modeling?

RQ2: How can Fedem be used to perform musculoskeletal simulations?

1.3 Related work

The work carried out on this master thesis is based on the models and documentation provided by the open-source software OpenSim and the models provided by the OpenSim community.

1.4 Structure

The scope of this master thesis is based on the assignment tasks specified by the supervisor. After discussion with the supervisor the assignment point task number two regarding EU projects have not been prioritized, but is briefly presented in chapter 2 and discussed in chapter 6.

Chapter 2 presents the underlying theory of how to build a KBE system and an AML application. Furthermore, it will cover the anatomy of the musculoskeletal system and existing musculoskeletal modelling. Lastly, the data formats used will be presented.

Chapter 3 is a review of the methods and tools used when developing the Adaptive Modeling Language (AML) application. Chapter 4 presents the development process of the musculoskeletal program. In chapter 5 the results are presented, while chapter 6 consists of a discussion. Chapter 7 presents the conclusions, and in chapter 8 further work can be found.

Chapter 2

Theory

2.1 Knowledge-Based Engineering

Knowledge-based engineering is described by Technosoft as the merger between object-oriented programming, artificial intelligence and computer aided design.

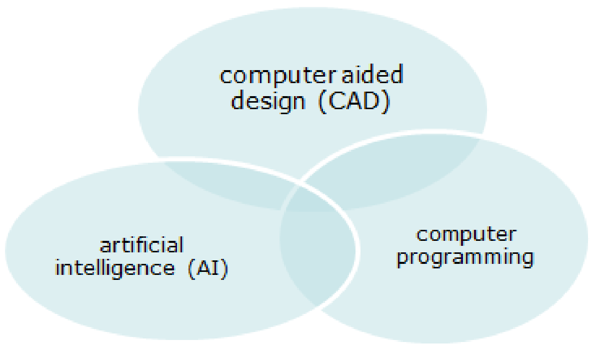


Figure 2.1: Knowledge-based engineering

Furthermore, KBE systems can be used automate the design process by taking advantage of flexible geometry handling and the re-use of knowledge. Processes such as iterations of geometry or "routine design" are examples of tasks with potential for automation. Other uses are reversed engineering applications where the geometry is defect after wear. By determining the accurate state of the geometry, the analysis can be re-done, whereas before approximations to fit standards had to be followed.

2.2 Adaptive Modeling Language

Adaptive modeling language (AML) is developed by Technosoft and is a knowledge-based engineering modeling language. AML programing is based on the programming language lisp.

One of the main advantages of using AML is the available methods for automating finite element modeling and mesh generation. The graphical user interface makes it possible to visualize geometry modeling and create custom made front-end to the AML applications.

The basic building-blocks when making an AML application are the use of:

- Objects
- Classes
- Inheritance

The objects in the program are similar to what we think of as an object in the real world, and therefore in AML objects is the representation of a real world entity. Objects can also consist of other objects in an object/sub-object relationship. One example of this could be a human body model where the legs could consist of bones and muscles.

Classes can be viewed as the "template" of an object. Hence, the classes can be reused to create new objects, with the same attribute name (state) and methods (behavior). Each of these new objects is an instance of the class.

Methods are defined on classes and are used for the objects to send information from one to another. Defining methods are similar to defining functions, only that methods can only be used on classes which it is defined upon. Another method of retrieving information about an object is using the-referencing.

The-referencing will look up the object tree for a property or an object with the same name. In comparison the use of default functions is similar to the-referencing. A property set as default will look up the tree for a property with the same name. The default function is encouraged because of its many functional advantages. By setting a property as default, generic classes can be made that enable reuse of class properties and methods. It also makes it easier to define properties for an instance in only one place, which makes it easier to get an overview of the code.

The-referencing has other applications, such as referencing down the object tree instances. Hence, in this case the structures of the instances need to be known. Whereas if methods are used, the object that is calling the method does not know how the objects are arranged. Also, methods are more flexible to use if the code is changed, because not every path has to be changed.

An example of using the-referencing to find an object ”underneath” the starting point, using the lower-body as an example could look something like this *(the lower-body leg knee)*. A shortcut for the-referencing is the use of the *!*. The same example done again would look like this: *!lower-body leg knee*. It is important to be aware of that the brackets are built in when using the shortcut.

When the-referencing is used to find an object or property “above” the object, it is not necessary to specify the path. Using the same example, but changing it to defining a property in the knee but needing the lower-body property, one can simply put *!lower-body*.

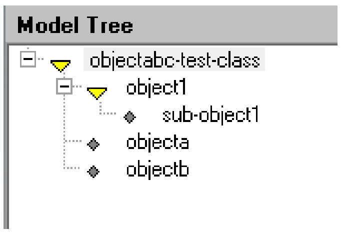


Figure 2.2: Caption from the AML model tree

An example which illustrates this is the object tree in **figure 2.2**. The sub-object1 can reach everything in the tree by using the standard the-referencing. Objecta can only reach the objects on the same level, which means everything except sub-object1. Below is the code for the model from **figure 2.2**. What might be not so intuitive is that in the properties of the initiated class objectabc-test-class can reach the sub-objects object1, objecta and objectb even though it is visually underneath the object.

Listing 2.1: The-referencing example AML code.

```
(define−class sub−object1−class
   :inherit−from (
      object
   )
   :properties (
      sub−object1−property 'sub−1a−properties
   )
   :subobjects (
      (sub−object1 :class 'object
         test2 !objecta
      )
   )
)
(define−class object1−class
   :inherit−from (
      object
   )
```

```
  :properties (
    object1-property '1a-properties
  )
  :subobjects (
    (object1 :class 'sub-object1-class
      test1 !objecta
    )
  )
)

(define-class objectabc-test-class
  :inherit-from (
    object1-class
  )
  :properties (
    object-a-b-list (list !objecta !objectb )
    test1 !object1
  )
  :subobjects (
    (objecta :class 'object
      object-id 'a
    )
    (objectb :class 'object
      object-id 'b
      test1 !object-a-b-list
      test2 !object-id
      test3 (the object-id (:from !objecta))
      test4 !object1
      test5 !objecta
    )
  )
)
```

Because the-referencing will search for the first object or property with the same name, it is important to be aware of this when writing code. The-referencing an unwanted object or property can be avoided by making sure that the object that is referred to has a name that cannot be mistaken or already be a default property of the above objects.

Other built-in keywords for the-referencing are *Superior* and *superior superior*. *Superior* starts the the-referencing relatively one and two levels up. This is practical in order to avoid circular dependency if built-in properties are demanded from the same object. This could easily be avoided if different names are used. However, to use the superior function, *superior superior* has to be used.

An illustration of this can be seen in the example below of tree spheres, with id's defined descending with the red big sphere with id 3. **Figure 2.3** shows the AML model tree and the graphics display of the system.

By using the inspect tool by right-clicking in the model tree on an object, the

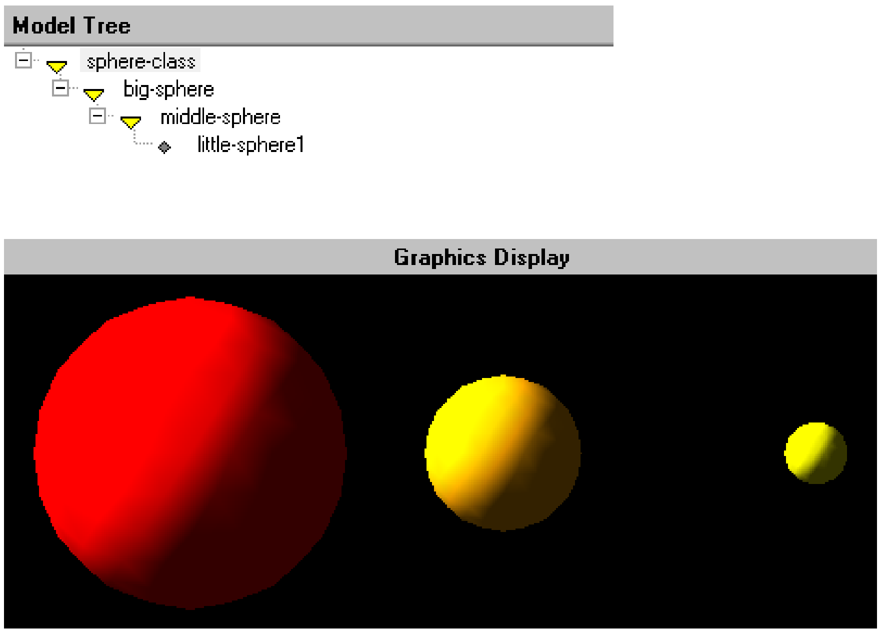


Figure 2.3: AML sphere superior example

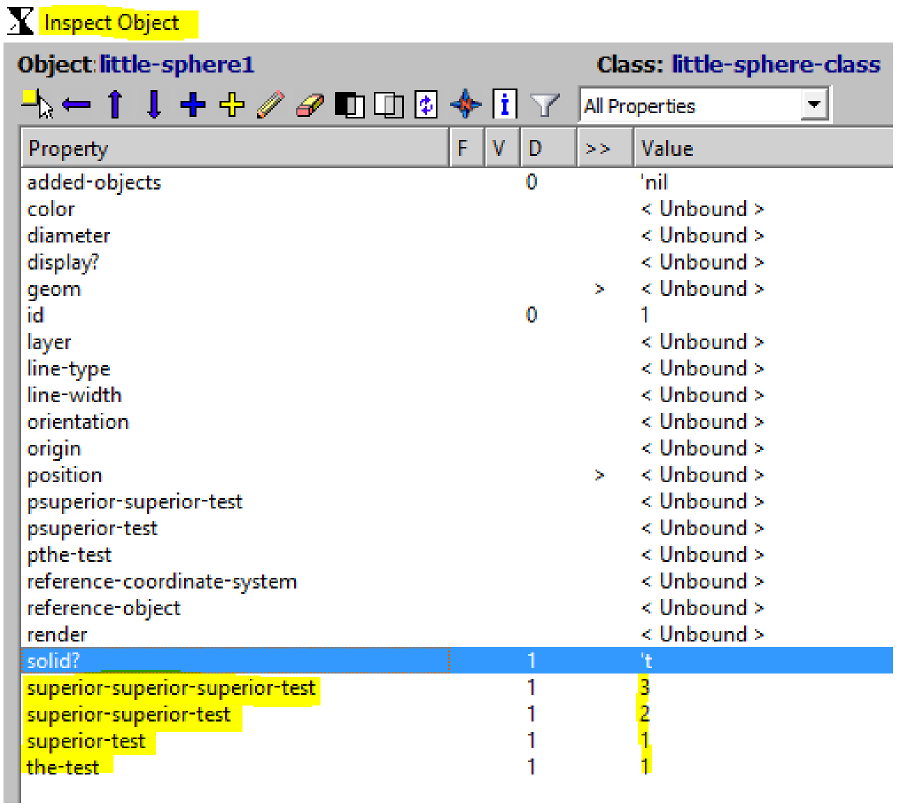
properties of the objects can be seen and changed by pressing the green checked button. This is a great tool to use when developing and debugging the code. **Figure 2.4** shows the properties of little-sphere1. The test properties show the id's chosen when using the-referencing and the superior keyword. As can be seen, the superior superior (∧∧) starts only one object up.

Listing 2.2: AML object little-sphere1.

```
(little−sphere1  :class  'little−sphere−class
   orientation (translate ( list   1 0 0 ))
   reference−object !middle−sphere
   id '1
   the−test !id
   superior−test ^id
   superior−superior−test ^^id
)
```

As can be seen from **figure 2.4** the test properties defined in the class have taken the value of the id's. A triple superior is needed to get the id of the red sphere.

Another the-referencing keyword is *:from*. As well as the-referencing *:from* also need to have brackets. The function *:from* is defined in the same way as the word from, by setting the starting point, in this case the starting point of the the-referencing. An example using the *:from* with the leg example is *(the knee (:from (the lower-body leg)))*. *:from* is a much simpler keyword then counting superior to


Inspect Object

Object: little-sphere1 Class: little-sphere-class

All Properties

Property	F	V	D	>>	Value
added-objects			0		'nil
color					< Unbound >
diameter					< Unbound >
display?					< Unbound >
geom				>	< Unbound >
id			0		1
layer					< Unbound >
line-type					< Unbound >
line-width					< Unbound >
orientation					< Unbound >
origin					< Unbound >
position				>	< Unbound >
psuperior-superior-test					< Unbound >
psuperior-test					< Unbound >
pthe-test					< Unbound >
reference-coordinate-system					< Unbound >
reference-object					< Unbound >
render					< Unbound >
solid?			1		't
superior-superior-superior-test			1		3
superior-superior-test			1		2
superior-test			1		1
the-test			1		1

Figure 2.4: AML inspect tool of the little-sphere1

avoid the-referencing reference to the wrong object or property. *:from* is also used to start the-referencing in a different object tree.

Inheritance is an important feature of the classes. A sub-class inherits the attributes and operations of the super-class. Object should only inherit from one class, while classes can have multiple inheritance.

2.3 Musculoskeletal anatomy

Next some general terms of human anatomy and an introduction to the biomechanical properties will be presented. This is the basis for modeling the human body and the basic terminology for the thesis.

The human bone consists of compact bone covering the surface, while on the inside the bones are spongy and have a marrow cavity. As a consequence, the bones' ability to restrain various types of forces such as tension, torsion and bending is dependent on the direction of the applied force.

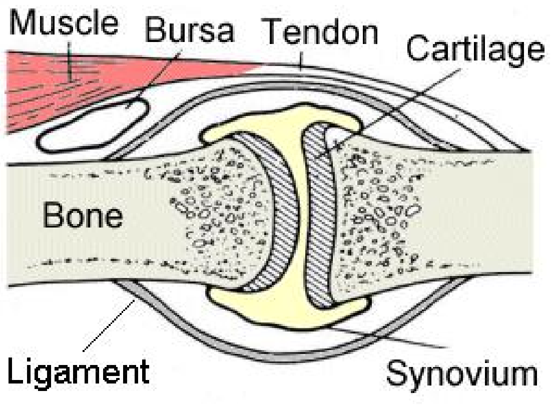


Figure 2.5: Components of the musculoskeletal model visualized. The bursa and the synovium (also called synovial membrane) consist of synovial fluid. Illustrations by Lauren Baker and Lizet Sosa (HSTE project)

The bone is a biological material and will grow and change, depending on many factors such as training, gender, age and lifestyle. Different parts of the bone will grow depending on these factors.

Cartilages' main functions are to allocate the pressure from the bones and make sure the friction is small. The collagen fibers in the cartilage molecular matrices make the cartilage able to resistant tension, while the proteoglycans (protein) and water make the cartilage able to handle compression

The synovial fluid has the mechanical properties of making the movement of bones on the cartilage as frictionless as possible and of absorbing shock.

The muscles are on average half of the body's total mass [3]. The force that the muscle can produce is dependent on the fibers in the muscles. The way the muscle works is that in each fiber there is a contractive element called a cross-bridge formation that is activated by an electrical potential, sent by the neural control system.

The tendons are what connect the muscle to the bones. Thus, the tendons have important nonlinear properties as transfer force from muscle to bone, and storage of elastic strain energy.

Ligaments Are similar to tendons, and the main difference is that they are connected to bones on both sides. The fibers in the ligaments are not as parallel to each other as the tendons. This makes the ligaments able to handle stress in several directions.

2.3.1 Skeletal anatomy

Figure 2.6 shows some of the main anatomical terms of the human body. Details of the skull, neck, hands and toe are not included.

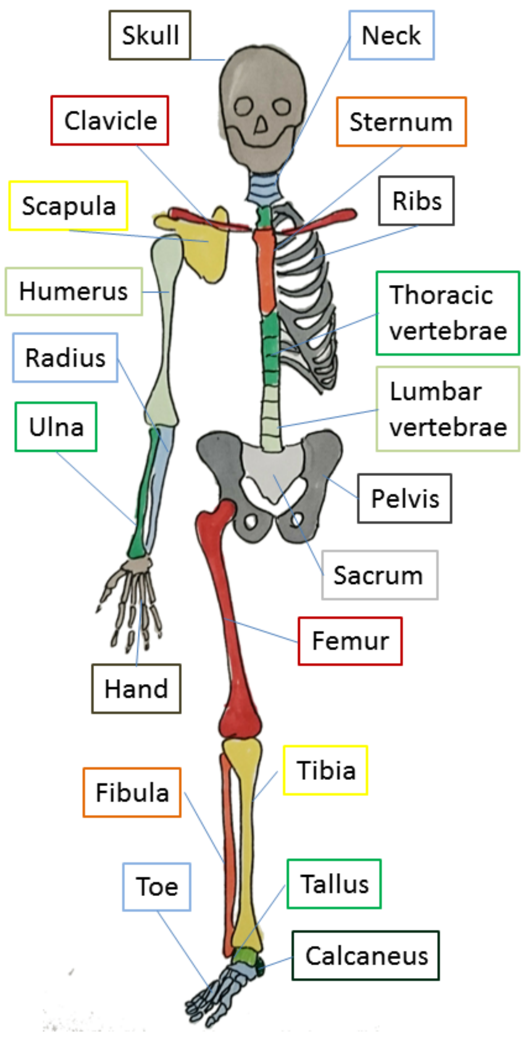


Figure 2.6: Skeletal anatomy

2.4 Multibody system

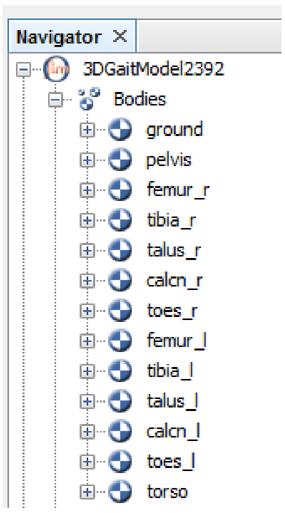
A multibody system is a collection of rigid or deformable bodies interconnected by kinematic joints, which undergo extensive translational and rotational displacement. The dynamic equations that govern the motion of these systems are non-linear and require in most cases numerical solutions.[4]

A multibody system bodies can be either flexible or rigid. The bodies are considered rigid if deformations have no relevant contribution to the system.

2.4.1 OpenSim

OpenSim is a multibody open source software from the National Center for Simulation in Rehabilitation Research (NCSRR). The bodies in OpenSim are modeled as rigid bodies. It is easy to confuse the bodies in the OpenSim model with the bone segments, partly because they are called the same as the bone in many cases. One body is actually the whole body segment and not just the bone. This means that more bones can be in one body. An example of this is the hand. When investigating walking for instance, there is no need to model all the joints and bones in the hand as separate bodies. Different models have different bodies depending on the wanted complexity of the model.

Figure 2.7 shows the bodies that are in the gait2392 model.



```
Navigator
3DGaitModel2392
  Bodies
    ground
    pelvis
    femur_r
    tibia_r
    talus_r
    calcn_r
    toes_r
    femur_l
    tibia_l
    talus_l
    calcn_l
    toes_l
    torso
```

Figure 2.7: Bodies in the gait2392 model, from the OpenSim navigator

2.5 Contact modeling

In OpenSim there are options to model contact forces, however this is not included in the thoracolumbar or the gait 2392 model. The available methods of doing this are using the Hunt-Crossly modelling or the elastic foundation modelling.

The Hunt-Crossly model represents the contact dynamics of a viscoelastic system, based on Hertz's elastic theory with a nonlinear damper. The simulations can be done with both discrete methods and continuous methods, like the finite element method. Example of scenario where the Hunt-Crossly modelling is used is when simulating the ground contact force during walking, instead of using experimental ground force values. [5]

Elastic foundation contact model or also called rigid body spring model. Examples of scenarios using elastic foundation model is to find the contact pressure in the knee joint. [6][7]

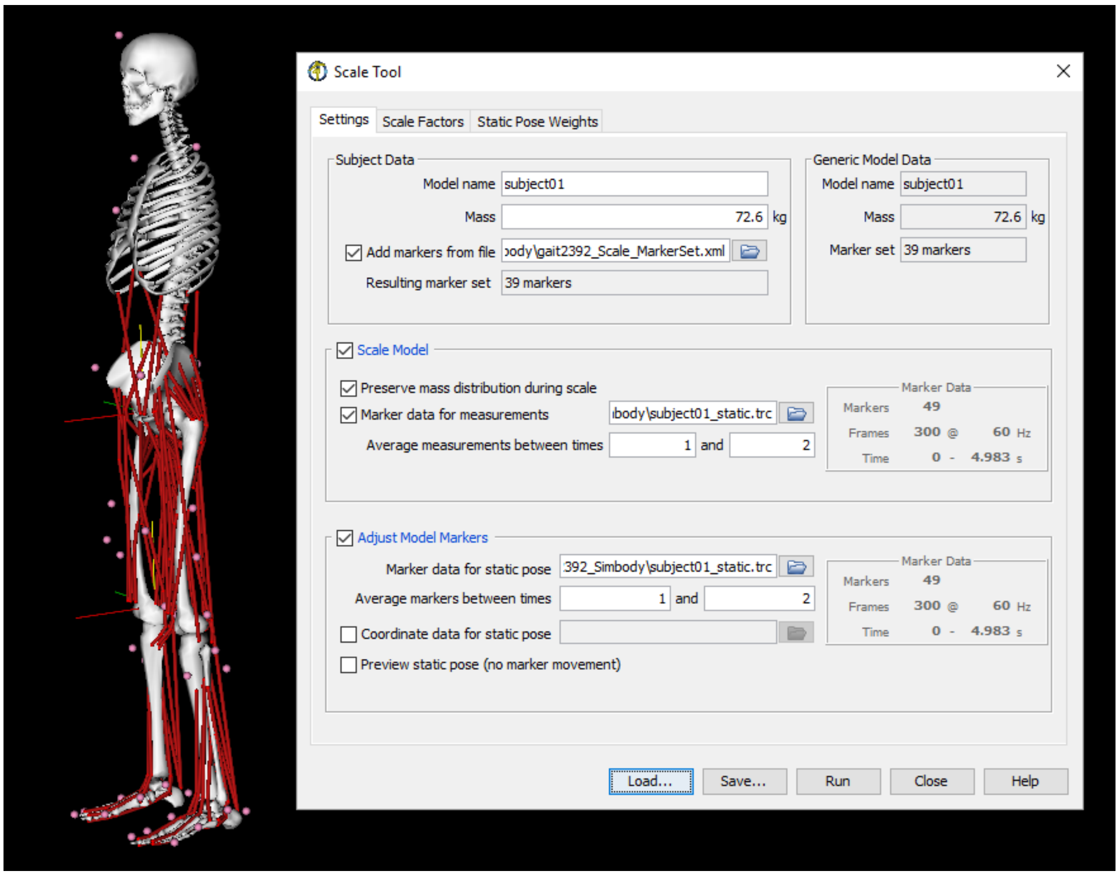
2.6 OpenSim scaling tool

The scaling done in OpenSim is based on fitting experimental marker tracking on the subject to the virtual model. The experimental marker data input is in a .trc (Track Row Column) file format, while the virtual markers are in a XML marker set file called Scale_MarkerSet. In the marker set file the name of the marker and location in parent can be found. After the virtual model is scaled to fit experimental markers, other properties such as mass inertia, body segment dimensions, muscle actuators and wrapping objects are also scaled based on these scale factors.

The automatic way of finding the scale factor is to take the average distance between the experimental and virtual markers over the time frames specified in the .trc file. When the scale factor is found, the rest of the properties can be scaled. The available gait2392 model with scaling data available uses 49 markers and 300 frames. **Figure 2.8** shows the scaling tool and the applied markers to the subject01, which are scaled from the gait2392 model. The markers are visualized by the pink dots.

2.7 OpenSim simulation tools

In OpenSim there are different tools available to simulate motion. OpenSim has mostly focused on gait-related motion. A summary of some of the different tools available in OpenSim will be presented next.


Scale Tool

Settings | Scale Factors | Static Pose Weights

Subject Data
Model name: subject01
Mass: 72.6 kg
☑ Add markers from file: ›ody\gait2392_Scale_MarkerSet.xml
Resulting marker set: 39 markers

Generic Model Data
Model name: subject01
Mass: 72.6 kg
Marker set: 39 markers

☑ Scale Model
☑ Preserve mass distribution during scale
☑ Marker data for measurements: ıbody\subject01_static.trc
Average measurements between times: 1 and 2

Marker Data
Markers: 49
Frames: 300 @ 60 Hz
Time: 0 - 4.983 s

☑ Adjust Model Markers
Marker data for static pose: 392_Simbody\subject01_static.trc
Average markers between times: 1 and 2
☐ Coordinate data for static pose
☐ Preview static pose (no marker movement)

Marker Data
Markers: 49
Frames: 300 @ 60 Hz
Time: 0 - 4.983 s

Figure 2.8: OpenSim Scale tool and gait2392 model scaled

The Inverse Kinematics tool minimizes a sum of weighted squared errors between experimental markers and model markers by varying the joint angle, see **figure 2.9** . The output is a file containing the joint angles and translations. The inverse kinematic tool is necessary in order to use the Static Optimization, Residual Reduction Algorithm, and Computed Muscle Control tools, which are other simulation tools.

Inverse dynamics are using the inertia of the model to calculate the forces based on the joint angles and Newton's 2nd law. However, in OpenSim the geometry is only for visualization purposes, and the mass is represented as point mass. The mass includes the whole body segment, including muscles.

The static optimization tool is an extension of the inverse dynamics tool. It is used to get results such as muscle activations at each time step by minimizing the sum of squared muscle activations.

Figure 2.9: Marker trajectory. Illustrations by OpenSim

Forward Dynamics is used when the muscle activations are known to drive a forward simulation to generate additional data to the analysis.

Residual Reduction Algorithm (RRA) is a tool to minimize the effect of residuals. Residual can be described as the difference between the observed value and the estimated value, which can come from marker data processing errors. These errors can lead to great nonphysical forces. The RRA alters the mass center of a subject-specific model and permits the kinematics to be more dynamically consistent.

Computed Muscle Control (CMC) computes a set of muscle excitations based on the motion kinematics. Hence, the computed muscle control gives a set of data that could be used to control the muscle actuators.

2.8 OpenSim XML format and VTK formats

The OpenSim models are in the Extensible Markup Language (XML) format, which is practical for storing data in a structural way. In the osim file the main tags at the first level are the $< BodySet >$ and $< ForceSet >$. Under $< BodySet >$ are all the bodies, and under $< ForceSet >$ are all the forces.

Underneath is an example from the femur body, showing a caption of the femur joint.

Listing 2.3: A caption of the gait2392.osim XML file

```
<Body name="femur_r">
    <!--Joint that connects this body with the parent body.-->
    <Joint>
        <CustomJoint name="hip_r">
```

```
            <SpatialTransform>
             ....
            <parent_body>pelvis</parent_body>
         <location_in_parent>−0.0707 −0.0661 0.0835</location_in_parent>
         <orientation_in_parent> 0 0 0</orientation_in_parent>
         <location> 0 0 0</location>
         <orientation> 0 0 0</orientation>
   </Joint>
</Body>
```

The spatial transform gives first the rotation in the tree axis and the translations in the tree axis. If the $< coordinates >< /coordinates >$ is empty, the default function is set as constant 0. On the other side, if coordinates are defined $< coordinates > hipflexionr < /coordinates >$, the function can be found under $< CoordinateSet >$ where the joint constraints are saved.

Listing 2.4: A caption of the gait2392.osim XML file

```
<CoordinateSet>
   <Coordinate name="hip_flexion_r">
      <!−−Coordinate can describe rotational, translational, or coupled motion. Defaults to
           rotational.−−>
      <motion_type>rotational</motion_type>
      <!−−The value of this coordinate before any value has been set. Rotational coordinate
          value is in radians and Translational in meters.−−>
      <default_value>0</default_value>
      <!−−The speed value of this coordinate before any value has been set. Rotational
          coordinate value is in rad/s and Translational in m/s.−−>
      <default_speed_value>0</default_speed_value>
      <!−−The minimum and maximum values that the coordinate can range between.
          Rotational coordinate range in radians and Translational in meters.−−>
      <range>−2.0943951 2.0943951</range>
      <!−−Flag indicating whether or not the values of the coordinates should be limited to
          the range, above.−−>
      <clamped>false</clamped>
      <!−−Flag indicating whether or not the values of the coordinates should be
          constrained to the current (e.g. default) value, above.−−>
      <locked>false</locked>
      <!−−If specified, the coordinate can be prescribed by a function of time. It can be
          any OpenSim Function with valid second order derivatives.−−>
   </Coordinate>
```

In the thoracolumbar spine model some of the geometry positioning is defined under the visual object tag because the origin of the geometry is not defined as the joint center and there is no independent joint connecting the geometry.

Listing 2.5: A caption of the gait2392.osim XML file

```
<GeometrySet>
    <objects>
      <DisplayGeometry>
```

```
        <!--Name of geometry file .vtp, .stl, .obj-->
        <geometry_file>pisiform.vtp</geometry_file>
        <!--Color used to display the geometry when visible-->
        <color> 1 1 1</color>
        <!--Name of texture file .jpg, .bmp-->
        <!--in body transform specified as 3 rotations (rad) followed by 3 translations rX
            rY rZ tx ty tz-->
        <transform> -0 0 -0 -0.013388 -0.009886 -0.010593</transform>
        <!--Three scale factors for display purposes: scaleX scaleY scaleZ-->
        <scale_factors> 1 1 1</scale_factors>
        <!--Display Pref. 0:Hide 1:Wire 3:Flat 4:Shaded-->
        <display_preference>4</display_preference>
        <!--Display opacity between 0.0 and 1.0-->
        <opacity>1</opacity>
    </DisplayGeometry>
```

2.8.1 Wrapping functions and via points

In OpenSim there are tree alternative ways to represent the muscle tendon path. What is similar for all of them is that the starting (origin) and ending (insertion) point are fixed in the parent body. However, during movement the path of the muscle will change.

The first and simplest method is the use of via points. These points are also fixed to a body, but during a defined certain motion they are used to change the path of the muscle.

Another method is the wrap points, which are constrained by principal geometrical shapes that the wrap point will automatically wrap around during motion

The last method is the moving muscle points, which are used if the wrap point is not suitable. The moving muscle points' position is also relative to the parent body but is defined by a function related to the motion.

In the OpenSim models there are approximately 26 and 665 moving muscle points found under the XML tag $< ConditionalPathPoints >$.

In the thoracolumbar here are 8 wrapping objects in the abdomen and pelvis under the tag $< wrap\_object >$.

2.8.2 The VTK format used in surface geometry

The Visualization Toolkit (VTK) is an open source software system for 3D computer graphics, image processing, and visualization[1]. VTK is used to pre-process the geometry surface of the OpenSim models. As a result the surface data is stored in an XML format in a VTK file. The structure has three main tags under

[1]VTK http://www.vtk.org/

the header $< PointData >$, $< Points >$ and $< Polys >$. The $< PointData >$ consists of all the normals of a point or the element of a polygon mesh. Next is the $< Points >$, which consists of all the point coordinates of the model. Lastly the $< Polys >$ consists of the points that connect to an element.

2.9 Finite element method and meshing

When having a multibody system with flexible bodies, one approach is using the finite element method to find the deformation. The method takes advantage of finding approximations for partial differential equations on a small element in the body. An essential part of the model is therefore dividing the geometry into small elements called a mesh.

2.9.1 Rigid Body Elements

RBE2 is one of several rigid body elements used in MSC Nastran, used to connect nodes, see **figure 2.10**.

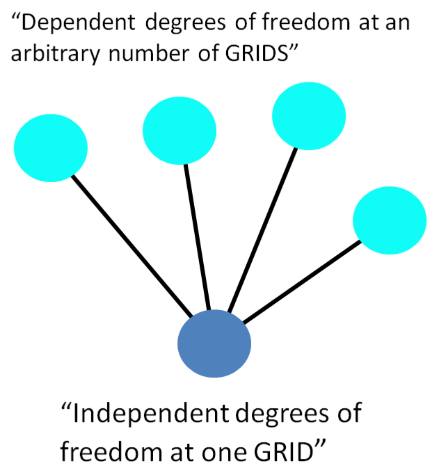

"Dependent degrees of freedom at an arbitrary number of GRIDS"
"Independent degrees of freedom at one GRID"

Figure 2.10: Definition of RBE2 from MSC Nastran user guide [8]

There is no relative motion between the dependent nodes of one RBE2 which gives it additional stiffness. Independent nodes in the RBE2 can not share dependent nodes. RBE3 is a different RBE and is used to distribute applied loads and mass in a model. Unlike the RBE2, the RBE3 does not add additional stiffness to the model, but allows translation between the dependent nodes. [8].

2.10 Horizon 2020

Horizon 2020 is the biggest EU Research and Innovation programme. Over the time frame 2014 until 2020 the program will be funding with nearly 80 billion euro. The overall goal is to drive economic growth, create jobs and break down barriers to contribute to collaboration between nations.

There are three main categories for calls; Excellent Science, Industrial Leadership and Societal challenges. The main categories is divided into smaller topics, where on the online "participant portal"[2] one can search for calls for proposals.

[2]Participant portal `http://ec.europa.eu/research/participants/portal/desktop/en/home.html`

Chapter 3

Methodology

OpenSim was closely reviewed in the literature study in the project thesis prior to the master thesis, and a work-shop in Bologna, Italy was attended to learn more about the modeling and simulations in the software. The workshop was held at the Rizzoli Orthopaedic Institute and led by OpenSim fellows. By attending the workshop practical experience with using the OpenSim software and models was achieved. Based on this it was decided to use the OpenSim models and simulations as a benchmark of the muscular modeling and simulation. Because of the knowledge and experience in the KBE language AML and finite element modeling in Fedem by the supervisor, it was reasonable to combine the different software when making the musculoskeletal model.

The focus of this thesis is on technical development of a KBE application. Previous to the thesis work the course "Introduction to automation with KBE" (TMM4270) was taken at NTNU. This course has been the basis and the background for developing the AML model. All development was performed on a 64-bit computer running Microsoft Windows 10.

Human physiology and bio-mechanics are based on the project thesis[9], and educational literature used in physiotherapy education. [3][10].

3.1 The OpenSim software

OpenSim version 3.3 software was downloaded from the SimTK web page. SimTK is a sister site of OpenSim and is a community of hundreds of biomedical research teams and projects. The program was downloaded and installed on the computer.The program and models are well documented on the simtk-confluence web page, and this is also the references of the OpenSim software and model's used in this thesis.[1]

[1]simtk-confluence `http://simtk-confluence.stanford.edu:8080/dashboard.action`

3.1.1 OpenSim models

Along with the OpenSim software it is possible to download a few models that are maintained by the OpenSim project. The model gait2392 used in this thesis is one of these available models.

When developing the AML models, all surface geometry from the OpenSim model is based on the OpenSim geometry file library. This provides a good starting point for importing the geometry to the AML body. The joints in the model are somewhat different from each other depending on the desired complexity of the model and the definition of the body segments. In this thesis the AML model has been based on the surface geometry, the positioning of joints and muscle origin and insertion point from the model gait2392 and thoracolumbar spine. **Figure 3.1** shows the two models in the OpenSim graphical user interface. The green/blue elliptical shape in **figure 3.1**, is part of the body abdomen, and consist mainly of wrapping objects for the muscles.

The gait2392 model was created by Darryl Thelen (University of Wisconsin-Madison) and Ajay Seth, Frank C. Anderson, and Scott L. Delp (Stanford University). The model was created to investigate gait, and is unscaled, 1.8 m tall and has a mass of 75.16kg.

The thoracolumbar spine model is part of a project aiming to create advances in the field of modeling the thoracolumbar spine from 2015. The model is available for download on SimTK homepage. The full documentation of the model can be found in the publication by Bruno et al. [11]. The skeletal anatomy is based on the CT scan of a 25-year-old male obtained from the OpenSim geometry file library.

The arms, head and neck from the thoracolumbar model is part of the available models from the OpenSim project. The upper extremity model is by Holzbaur et al. [12] and the neck model by Vasvada et al. [13].

3.2 Matlab

Matlab version 7.12.0 (R2011a) was used in this thesis to investigate splitting geometry that consist of different parts and to investigate and find the center of the knee joint.

When investing the splines that were originally used to represent motion of the joints in OpenSim, the built-in natural cubic spline function (cscvn) was used. Documentation of this function can be found on Matlab's documentation page online. The circle-fit function by Izhak Bucher (1981) was used to find the center for the knee joint.

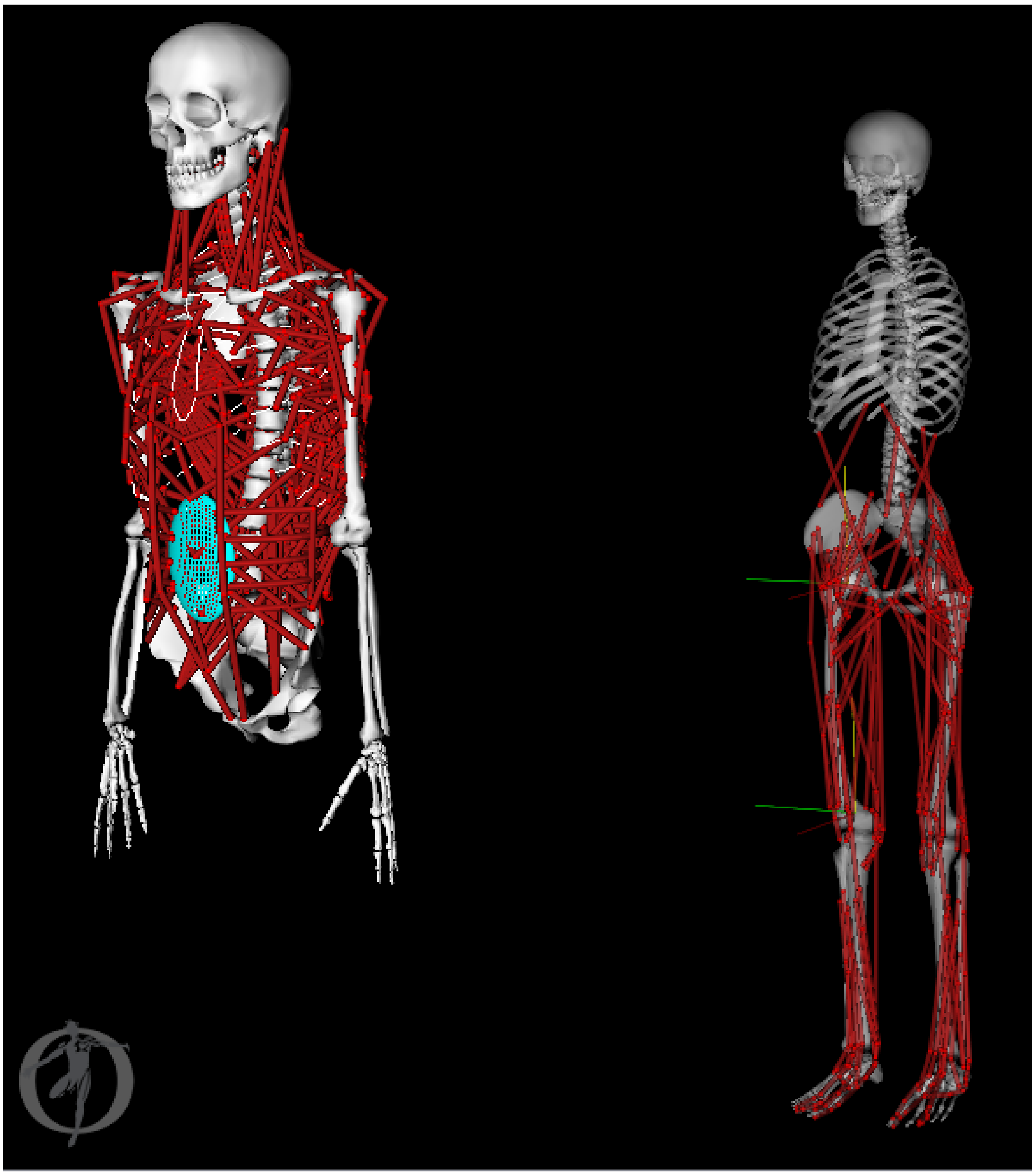


Figure 3.1: OpenSim models used in the development (Thoracolumbar and gait2392)

3.3 XML-Element Tree with Python

At the time there was unfortunately little or no documentation on xml parsing in the reference manual available from Technosoft. Because of the easily available documentation on xml parsing in Python, the xml tree extension was used to pre-process the OpenSim xml files. Python 2.7.0 was used along with the IDLE editor.

3.4 Adaptive Modeling Language

After taking the class “Introduction to automation with KBE” in the previous semester, where the programming language Adaptive Modeling Language (AML) was used, it was natural to continue using AML.

AML version 6.31 was downloaded along with the XEmacs editor used to compile and interface with the running AML process. The AML/Pantran, AML/Analysis and AML/Nastran interface are not included in the standard release. Hence, the aml-analysis-module-pack-type-3-01-06 was downloaded from Technosoft to perform the meshing and analysis. In addition, Nastran needed to be downloaded, and the nastran program path and the file which the nastran data would be written to needed to be added to the logical-path-file (logical.pth).

Listing 3.1: Part of logical path file in AML directory describing Nastran paths

```
:nastran-data     "C:\FedemWorkspace\"
:nastran-path     "C:\ProgramFiles\Siemens\NX8.0\NXNASTRAN\bin\"
```

Technosoft provides documentation for most of the features in AML in the AML-Reference-Guide. Unfortunately, some classes and topics were only partly documented, and this was often the bottleneck in the development. The AML-Basic-Training manual covers the basic AML development and contains good examples.

3.4.1 Source code management

For the application to run properly, the files have to be compiled in the right order, and the right classes have to be initiated in order. In this thesis there is one musculoskeletal-system which consists of the possibility to use the geometry of the thoracolumbar with the spine modeled in detail and one based on the gait2392. Both geometries can be initiated by the body-mesh-analysis class.

Listing 3.2: AMl system.def file

```
(DEFINE-SYSTEM :Musculoskeletal-System
    :files     '(
        "web_surface_class.aml"
        " joint_class_gait .aml"
        " joint_class .aml"
        "sub_geometry_class_gait.aml"
        "sub-geometry_class.aml"
```

```
      "muscles.aml"
      "body_class.aml"
      "meshing-analysis.aml"
   )
)
```

Listing 3.3: Part of logical path file in AML directory describing system path

```
:Musculoskeletal-System   "C:\AML_workspace\"
```

A system is defined by the system.def file and a folder containing the sources files. To be able to compile or load the system from the editor, the path of the system.def needs to be added to the logical path file. If the source code is changed, the system needs to be compiled to accommodate the changes

Listing 3.4: XEmacs command line to compile system

```
(compile-system :My-Body-System)
```

3.5 Fedem

Fedem is multibody dynamics licensed software for mechanical systems. Software version 7.0 was used in this thesis for the possible use as a simulation tool. After the model is created in AML the .bdf files are printed. Each .bdf file contains the elements of the body, a list of the grid points, and the RBE2s from the muscles and joints. The bodies are currently in the thesis added manually by the add part button. When the whole system is put together with joints either manually in Fedem or automatically in AML, this is saved in a .fmm system file. The documentation used in this thesis is from the reference manual available in the software.

3.6 Meshing and boundary conditions

3.6.1 Bone meshing

In this thesis a triangular mesh was chosen with a prescribed thickness. The bone is made up by different materials, and this simplification is meant to represent the compact bone that is covering the surface. The purpose of the mesh generation is to show the automatic mesh capability in AML, and further refinement is necessary to model more realistic properties of the bone.

3.6.2 Boundary conditions and RBE2s

The boundary conditions in the models are the joints that connect the bodies together. The joints in the OpenSim are mainly free joints with constraints in the rotational range and translation. In this thesis the joints are represented in the AML model, but due to time constraints they are not printed to be represented in the data output. However, the rigid body elements (RBE2) that connect the joint node to the slave nodes in the body are implemented.

Chapter 4

Development process

The goal of the development process of the AML application is to create a code that can be reused to model different persons with individual measurements. To achieve this type of reuse of knowledge, it is important that the variables related to individual sizes can be changed without having to change the code significantly.

Another aspect when developing is to create an application that can automate the process of modeling, simulating and analyzing musculoskeletal models.

Figure 4.1 illustrates parts of the whole development process, followed by a more detailed elaboration of each step.

As described in the theory chapter, the bodies in the OpenSim file are independent of the geometry files, which are only for visualization purposes. In the AML file each body consist of rigid bone geometry. As a consequence of this, OpenSim bodies have to be reconstructed into the desired AML-bodies. In more practical terms, this means splitting up OpenSim bodies into AML bodies and creating new joints for the new unconstrained bodies.

This process is easy when the geometry for these new bodies are stored in separate data files. For example, the tibia body in OpenSim consists of the two geometry files tibia.vtp and fibula.vtp. If, on the other hand, they are stored in the same file such as the toe body which consists of the bofoot.vtp that contains all the toes it is a little more complicated.

The difference in how the bodies are defined in the two systems leaves two options. Separate the geometry files into smaller bodies and connect them with rigid connections. Separating the geometry is tested in Matlab by separating groups of elements that are not connected to each other. Because of the work required and the added complexity of adding the rigid connections, this option was not further investigated. The Matlab script can be found in the appendix B.1, for the purpose of further work on creating a more detailed model.

Option number two is leaving the geometry "floating" in air with the intention of finding a method in AML that can "glue" the bone part together into a more

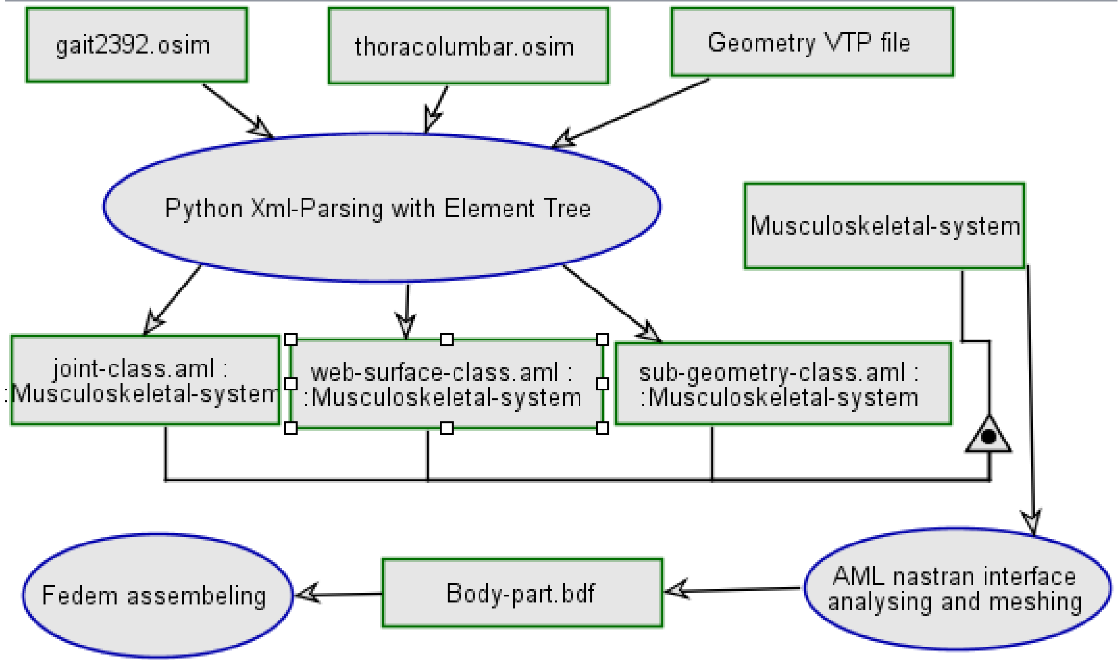

gait2392.osim
thoracolumbar.osim
Geometry VTP file
Python Xml-Parsing with Element Tree
Musculoskeletal-system
joint-class.aml : :Musculoskeletal-system
web-surface-class.aml : :Musculoskeletal-system
sub-geometry-class.aml : :Musculoskeletal-system
Fedem assembeling
Body-part.bdf
AML nastran interface analysing and meshing

Figure 4.1: Opcat figure illustrating the file flow and processes in the development process. The AML files are part of the :musculoskeletal-system.

more stable rigid body.

4.1 OpenSim data pre-processing with Python

The element tree module in Python is an effective way to store hierarchical data structures. For this reason, the module is used in this thesis by simply looping trough the xml structure, assuming that the structure of the xml file and the wanted information is well-known.

Listing 4.1: Python element tree loop to get OpenSim bodies and geometry files

```
import xml.etree.ElementTree as ET
from os import path

#input path
path_input ={
    'gait':'C:\OpenSimWorkspace\Models\Gait2392_Simbody\gait2392_simbody.osim',
'thoracolumbar':'C:\OpenSimWorkspace\Models\Thoracolumbar_OSIM_V1
\Thoracolumbar_Spine_With_RibCage.osim'}

#input: gait or thoracolumbar
model= 'gait'
```

```
tree = ET.parse(path_input[model])
root = tree.getroot()

#function that prints body name and corresponding geometry files
def geometryfiles ():
    for body in root.findall('.//Body'):
        print '  ______________________  '
        print body.get('name')
        for geometry_file in body.findall('.//geometry_file'):
            geometry_file = geometry_file.text
            if geometry_file is not None:
                print geometry_file
```

Python was mainly used for two processes, see **figure4.2** .Firstly, pre-process the data files with points and connectivity into text files formatted so that the AML web-surface-class can use it as input. Secondly, to print the list of sub-objects that initiates the different bodies and joints. The initiated object need to consist of property constants that define the objects as position, orientation and reference to local or global reference system.

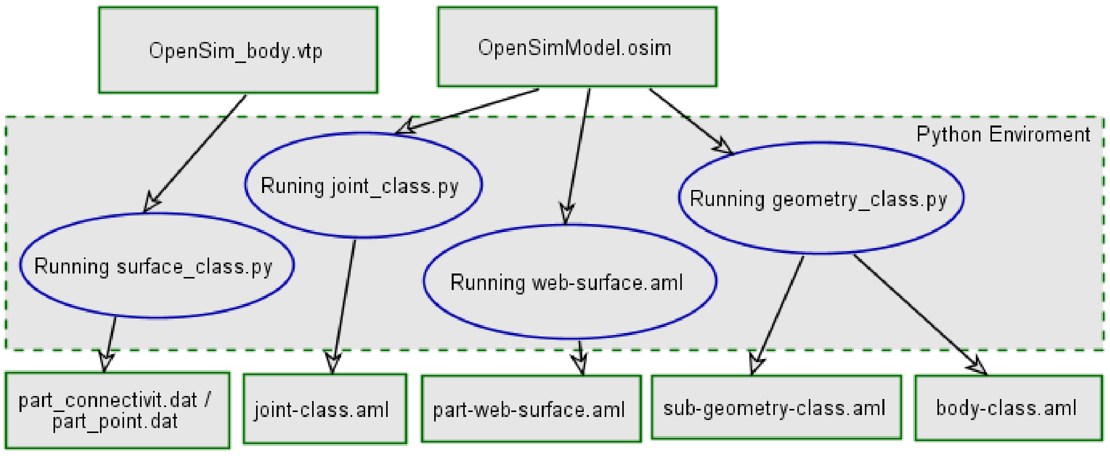


Figure 4.2: Opcat figure illustrating the input and output to the python scripts

In AML it is practical to have more than two levels in the body hierarchy. Mostly, because of the amount of bodies especially in the spine where there are 17 bodies. For this reason the pre-processing of the osim file bodies needed to be printed to the aml file in a way that ensured a tree structure. With this intention the python dictionary was practical when storing the structure of the bodies, see Pyton script below.

Listing 4.2: Python dictionary used to create aml tree structure

```
#input: treestructure of bodies Thoracolumbar
```

```
ground=['ground', 'sacrum', 'pelvis']
lumbar=['lumbar3', 'lumbar4', 'lumbar3', 'lumbar2', 'lumbar1', 'lumbar5']
thoracic=['thoracic12', 'thoracic11', 'thoracic10', 'thoracic9', 'thoracic8', 'thoracic7',
     'thoracic6', 'thoracic5', 'thoracic4', 'thoracic3', 'thoracic2', 'thoracic1']
rib=['sternum','rib12_R', 'rib11_R', 'rib10_R', 'rib9_R', 'rib8_R', 'rib7_R', 'rib6_R',
     'rib5_R', 'rib4_R', 'rib3_R', 'rib2_R', 'rib1_R', 'rib12_L', 'rib11_L', 'rib10_L',
     'rib9_L', 'rib8_L', 'rib7_L', 'rib6_L', 'rib5_L', 'rib4_L', 'rib3_L', 'rib2_L',
     'rib1_L',]
abd=[ 'Abdomen','Abd_L_L1', 'Abd_L_L2', 'Abd_L_L3', 'Abd_L_L4', 'Abd_L_L5', 'Abd_R_L1',
     'Abd_R_L2', 'Abd_R_L3', 'Abd_R_L4', 'Abd_R_L5']
arm=['clavicle_R', 'scapula_R', 'humerus_R', 'ulna_R', 'radius_R', 'clavicle_L',
     'scapula_L', 'humerus_L', 'ulna_L', 'radius_L']
hand=['hand_R', 'hand_L']
headneck=['head_neck']

body_structure_thoracolumbar={'ground':ground, 'lumbar':lumbar,'thoracic':thoracic,
     'rib':rib, 'abd':abd, 'arm':arm,'hand':hand,'headneck':headneck }

#input: treestructure of bodies gait2392
ground=['pelvis']
upper_body=['torso']
feet=['talus_r', 'calcn_r', 'toes_r', 'talus_l', 'calcn_l', 'toes_l']
legs=['femur_r', 'tibia_r', 'femur_l', 'tibia_l']

body_structure_gait2392={'ground':ground, 'upper_body':upper_body,'feet':feet, 'legs':legs }

body_structure={'gait':body_structure_gait2392,
     'thoracolumbar':body_structure_thoracolumbar }
body_structure= body_structure[model]
```

By creating a loop trough this structure and by creating the two functions write-sub-geometry-class (structure, part, element-tree-root) and write-body-class(structure, element-tree-root), the sub-geometry and body-class aml files were generated.

Since the joint-class is only two levels, the script was more straightforward. However, all the python script used the same approach to write to the AML file, see code below.

Listing 4.3: Python code to write AML files

```
def write_class(class_name, cont):
        with open(path.relpath(class_name + '.aml'), 'a') as f:
              f.write(cont)
              print 'Class: "' + part + '" written to AML file.'

write_class('sub-geometry_class_gait', aml_code) #aml_code is the string created from
     functions defined to fit aml classes
```

4.2 Geometry modelling in AML

In AML the tree structure hierarchy reflects the object instances and can be created by defining sub-objects of other object. In the development process the use of inheritance is essential for reuse in terms of properties and methods, but it is also used to inherit sub-objects. For instance, all the body parts have characteristics as points and connectivity describing the surface, which are inherited from the part-web-surface-class

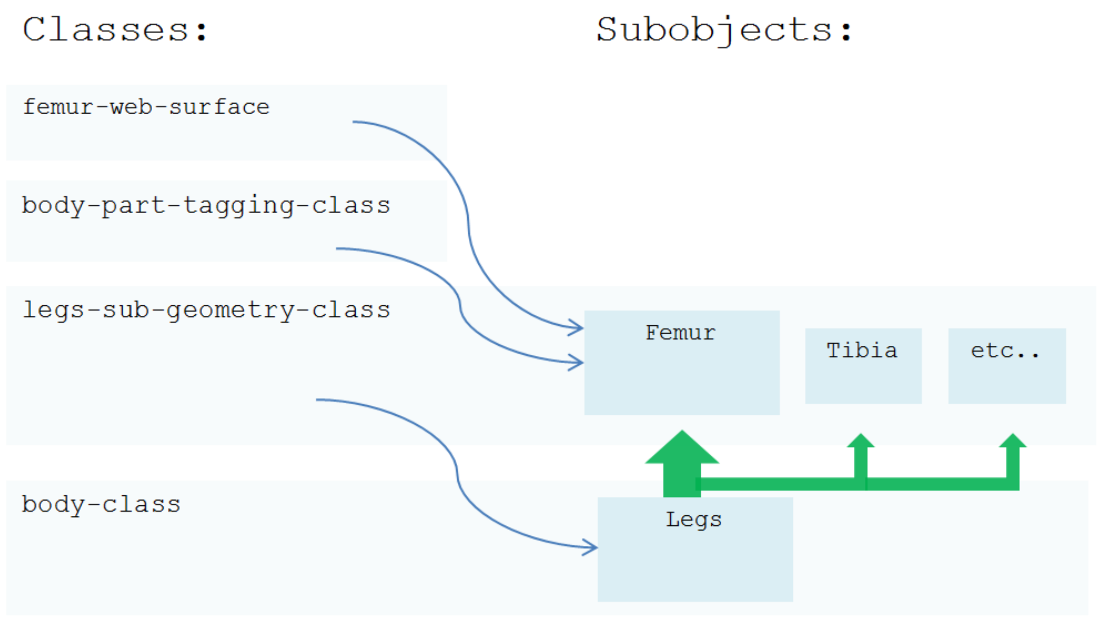


Figure 4.3: Femur example of class inheritance (blue arrow) and object tree hierarchy (green arrow)

Figure 4.3 visualizes how the sub-object tree hierarchy is created by initiating the body-class. The green arrow symbolizes the object hierarchy, while the blue arrows illustrate which classes the sub-objects inherit from. The green arrow or the object hierarchy is never specified; it is a consequence of the inheritance from a class with sub-objects defined. For the system to run, the classes have to be compiled in the order femur-web-surface and down. Since there is normally only one level of sub-object in each class, this is how the object tree is populated.

Below is a caption of the code illustrated in **figure 4.3**. The code show parts of the body-class, where the legs' sub-object is defined.

Listing 4.4: The AML body-class.

```
  :subobjects (
    (joints  :class  'gait-joint-class
    )

    (legs :class '(legs-sub-geometry-class)
        femur_r-reference-joint (the femur_r-joint (:from ^^joints))
        tibia_r -reference-joint (the tibia_r -joint (:from ^^joints))
        femur_l-reference-joint (the femur_l-joint (:from ^^joints))
        tibia_l -reference-joint (the tibia_l -joint (:from ^^joints))
    )
```

The legs are now initiated along with the properties of the reference-joint, which was defined in the joint-class. Following the legs to the legs-sub-geometry-class, the femur sub-object is initiated, and the orientation is set according to the joint-reference. By setting the joint-reference as (default nil), the sub-geometry-classes can be compiled before the body-class, providing flexibility when creating the body structure. In the leg-sub-geometry-class the other bodies in the leg are sub-objects as well, but for simplicity they are not shown in the example. The complete source code can be found in the appendix C.8.

Listing 4.5: The AML legs-sub-geometry-class.

```
(define-class legs-sub-geometry-class
  :inherit-from (
    object
  )
  :properties (
    max-elem-size 0.012
    min-elem-size 0.003

    femur_r-reference-joint (default nil)
     tibia_r -reference-joint (default nil)
    femur_l-reference-joint (default nil)
     tibia_l -reference-joint (default nil)
  )
  :subobjects (
    (femur :class '(body-part-tagging-class femur-web-surface)
      orientation ( list (rotate (radians-to-degrees -0 ) :x-axis)
      (rotate (radians-to-degrees 0 ) :y-axis)
      (rotate (radians-to-degrees -0 ) :z-axis)
      (translate ( list 0 0 0)))
      reference-coordinate-system ^^femur_r-reference-joint
    )
```

The body-part-tagging-class, see listing, is a class that all bodies inherit from to be able to do the meshing and analysis at a later point. Hence, the body-part-tagging-class has properties necessary for the meshing that all bodies inherit. Furthermore, the size of the mesh is part of the property tag-attribute, which is a built-in property of the tagging-class.

Listing 4.6: The AML body-part-tagging.

```
(define−class body−part−tagging−class
   :inherit−from (
      tagging−object
   )
   :properties  (
      max−elem−size (default nil)
      min−elem−size (default nil)
      tag−dimensions '(0 1 2 3)
      tag−attributes ( list  ^^max−elem−size ^^min−elem−size 1 0.1 0 20.0 1.0e−5)
   )
)
```

Lastly, the femur inherits the surface from the femur-web-surface class. The femur-web-surface inherits from the generic built-in web-surface-class. This class creates webs from either triangular or quadrilaterals, or both. The purpose of every object instance having its own web-surface class is simply for the reason that it was practical when testing positioning at an early stage. To save the amount of code writing in the future, this class could easily be made generic and the path of the point data could be defined at the body object instance.

Listing 4.7: The AML femur-web-surface-class.

```
(define−class femur−web−surface
    :inherit−from(web−surface−object)
     :properties  (
    nodes−file "C:\\PythonWorkspace\\femur_points.dat"
    con−file   "C:\\PythonWorkspace\\femur_connectivity.dat"
    cleanup? nil
    method 2
    )
)
```

4.3 Joint modelling

In the AML model the joints are defined independently of the bodies. Each body has a coordinate system which the body is orientated from. The standard joint that the body references from is the slave joint. In OpenSim the joint is positioned in the parent body as illustrated earlier. An example from OpenSim is the femur-joint, which is positioned in the pelvis- or ground- parent body. However, even if the definition of the joints differs a little, the body structure is built methodically in the same way. Start at the ground-joint at the pelvis and add bodies by referring to the previous part. The **figure 4.4** shows this topology from the OpenSim topology view.

The different joint-classes are simply a coordinate-system-object inheriting

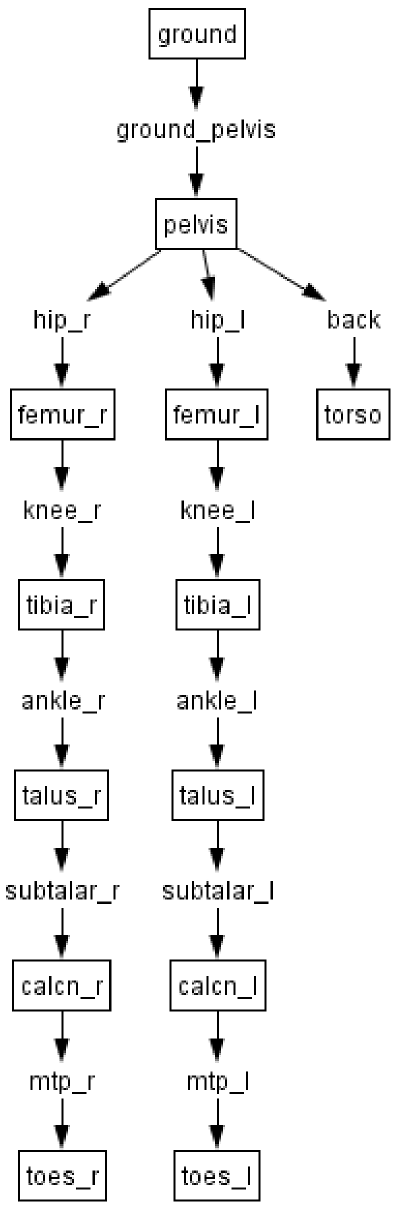


Figure 4.4: Topology view from OpenSim gait2392 model. Bodies are represented by boxes with the joint name written in between

from the coordinate system class. The joint movement constraint properties that are necessary for finite element analysis or dynamical simulation will be defined at a later point. The position of the bodies in the AML graphical user interface will be the design position of the system.

Even so, a user interface was created to be able to visualize the positioning of the joints when constraining the joint angle range. This test was helpful in making sure that the bodies and reference systems were put together correctly or in the same way as in the OpenSim model.

Again the femur will be used as an example. The design position angles of rotation in all directions are set as a default 0. The femur joint or the more commonly referred to as the hip joint, is defined in OpenSim as a free joint with rotation around the axes defined as the hip-adduction-r, hip-rotation-r and hip-flexion-r. By creating a user interface where the angle can be changed in a range that will illustrate the constraints on the joints, the motion can be investigated.

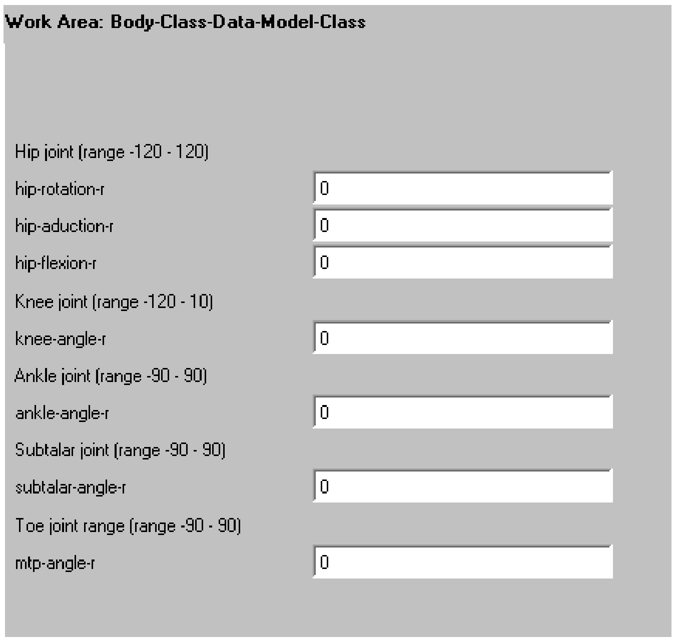
Listing 4.8: Creating a simpler user interface with body-class data-model-node-mixin.

```
(define-class body-class-data-model-class
  :inherit-from (body-class data-model-node-mixin)
  :properties (
     property-objects-list
     ( list
        "Hip joint (range -120 - 120)"
        ( list (the superior hip-rotation-r self) '(automatic-apply? t)
        )
        ( list (the superior hip-aduction-r self) '(automatic-apply? t)
        )
        ( list (the superior hip-flexion-r self) '(automatic-apply? t)
        )
     (hip-flexion-r :class 'editable-data-property-class
        label "hip-flexion-r"
        formula :inherit-formula
     )
     (hip-rotation-r :class 'editable-data-property-class
        label "hip-rotation-r"
        formula :inherit-formula
     )
     (hip-aduction-r :class 'editable-data-property-class
        label "hip-aduction-r"
        formula :inherit-formula
     )
```

Figure 4.5 shows how the work would look when adding the class to the body-class that is used. To get the input angles to be redrawn, the current model has to be undrawn and then redrawn.

4.3.1 Knee Joint

The knee-joint is one of the more complex joints to model because the relative motions between the two bodies are both rotational and translational. In OpenSim


Work Area: Body-Class-Data-Model-Class

Hip joint (range -120 - 120)	
hip-rotation-r	0
hip-aduction-r	0
hip-flexion-r	0
Knee joint (range -120 - 10)	
knee-angle-r	0
Ankle joint (range -90 - 90)	
ankle-angle-r	0
Subtalar joint (range -90 - 90)	
subtalar-angle-r	0
Toe joint range (range -90 - 90)	
mtp-angle-r	0

Figure 4.5: Class to enable visualisation of movement in AML

there is a spline function to draw a spline between points, see listing, where x is the input in radians and y is the translation in meters relative to the axis.

Listing 4.9: Simmspline from gait2392.osim.

```
<coordinates>knee_angle_l</coordinates>
  <!--Rotation or translation axis for the transform.-->
  <axis>1 0 0</axis>
  <!--Transform function of the generalized coordinates used to represent the amount of
      transformation along a specified axis.-->
  <function>
    <SimmSpline>
      <x> -2.0944 -1.74533 -1.39626 -1.0472 -0.698132 -0.349066 -0.174533 0.197344
          0.337395 0.490178 1.52146 2.0944</x>
      <y> -0.0032 0.00179 0.00411 0.0041 0.00212 -0.001 -0.0031 -0.005227 -0.005435
          -0.005574 -0.005435 -0.00525</y>
    </SimmSpline>
  </function>
```

Accordingly, the knee joint is described by such a spline. Instead of giving the knee joint six degrees of freedom, it is reduced to only one. If the knee angle is set as the variable giving the translation in y and x direction, the motion of the tibia relative to the femur will follow an elliptical curve.

To investigate whether a free joint or a cam joint should be used for modeling the knee-joint, the cscvn function is used in Matlab. The cscvn is defined as a natural interpolating cubic spline, which is created by letting the interpolated points be a type of piecewise polynomial to create a smooth curve.

Due the fact that the elliptical shape was close to a circle, a free joint was chosen. With that in mind, the coordinates for the joint RBE2 were found by choosing the best fitted circle in Matlab. The Matlab script can be found in the appendix B.2. **Figure 4.6** shows the results of the best fitted circle and the origin as well as the point generated from the cscvn curve.

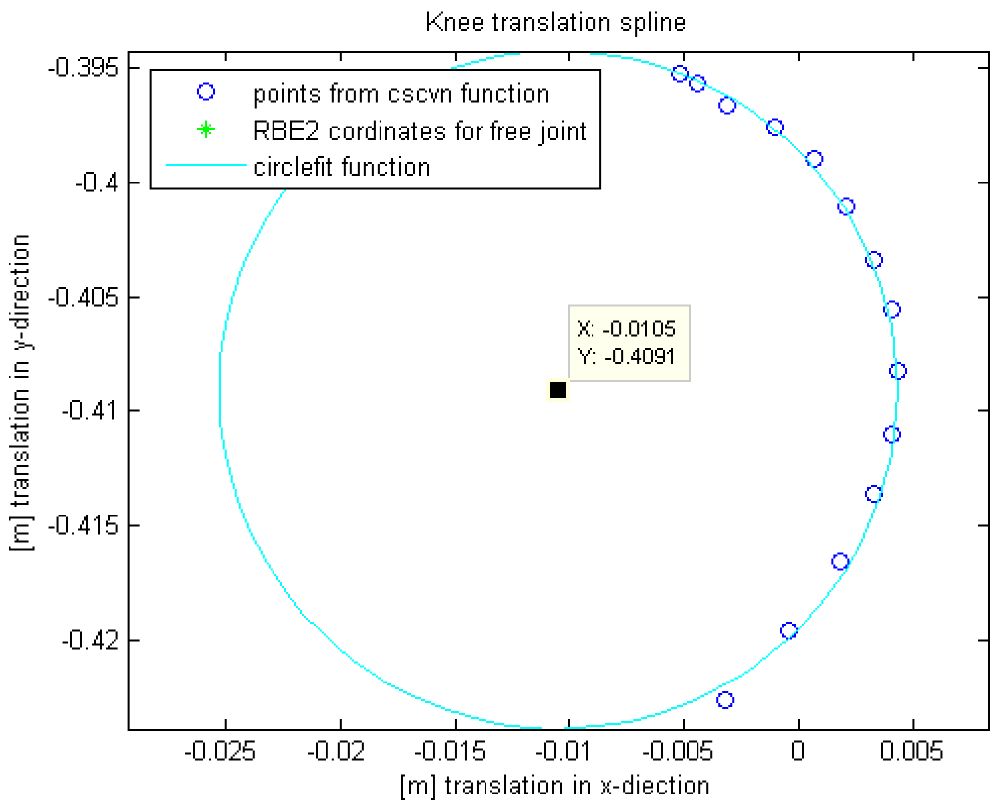


Figure 4.6: Matlab investigation of knee-joint

4.4 Muscle modelling

Muscle modeling is possibly the most complex part of the modeling. First of all, a model describing the force output from the muscle is needed. Furthermore, the muscle model is dependent on input such as the length of the model muscle as well as which muscle is activated when performing a certain motion. Because of the endless possibilities of motions and muscle activations, the muscles are the input driving force as well as the stabilizing dampers in a closed loop control system.

To start with, the main focus of this thesis is to demonstrate how a generic muscle geometry can be implemented in AML. Next, it will be creating a class that creates RBE2 for the muscles' origin and insertion point. A line will be drawn between these points, which could be the input to the force generation of the muscle. For this to be effective, the model would need more refinement to imitate the path that the muscle is actually taking.

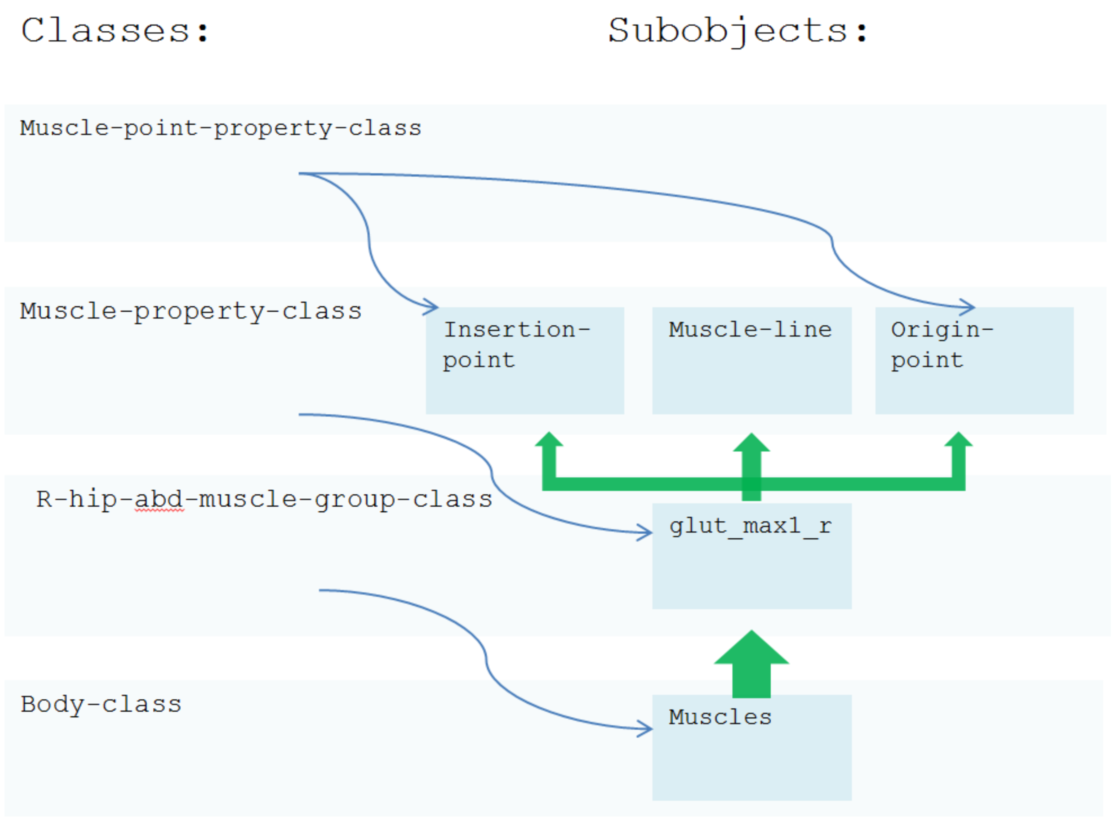


Figure 4.7: Class inheritance and object hierarchy

In **figure 4.6** the generic classes for creating the muscles are visualized. Again, the blue arrows show the inheritance, which creates the tree structure illustrated by green arrows.

The class muscle-point-properties-class inherits from the class point object. This class makes a point on a body by taking in a point coordinate relative to the reference body. Both properties are set as default nil to be initiated at a later point.

Muscle-properties-class is the generic class that all the muscles in the model inherit from. The class has three sub-objects and muscle-line-visualization. Muscle-line-visualization inherits from line-object class and draws a line between point1 and point2. Insertion-point and end-point both inherit from muscle-point-properties-class.

The code underneath show the R-hip-abd-muscle-group-class that initiate ten of the muscles in the hip. The example shows the first muscle in the group, where the information about the point coordinates is taken from the gait2392 model.

Listing 4.10: Parts of the AML R-hip-abd-muscle-group-class.

```
(define-class R-hip-abd-muscle-group-class
  :inherit-from (object)
  :properties (
     pelvis-muscle-ref (default nil)
     femur_r-muscle-ref (default nil)
     tibia_r-muscle-ref (default nil)
  )
  :subobjects(

     (glut_max1_r :class 'muscle-properties-class
        ; originally 4 points defining the muscle line
        insertion-point-coordinates '(-0.1195 0.0612 0.07)
        ending-point-coordinates '(-0.0277 -0.0566 0.047)
        insertion-point-reference-object ^^pelvis-muscle-ref
        ending-point-reference-object ^^femur_r-muscle-ref
        color 'yellow
     )
```

4.5 Meshing and analysis

The first step of the meshing analysis part is tagging. Tagging is essential to be able to grab information about the nodes in the part, and is also controlling the element size in the mesh. When performing the meshing and analysis, the class initiated is the body-mesh-analysis. The body-mesh-analysis-class is a generic class, where the body object is the object that gets meshed and analyzed. The body gets meshed, and a RBE2 is created in the joint referred to as the slave joint and master-joint. Furthermore the muscle points that “belong” to the body needs to inherit from the RBE2 class.

An essential part of the meshing and analysis is the body-analysis-class that inherits from the built in analysis-model-class. In this class all the nodes that need to be added to the output file are defined. In the musculoskeletal-system the sub-object node-set which inherits from analysis-node-set-class needs to have all the information about the added grid points in the pre-defined property query-objects-list.

```
(node-set :class 'analysis-node-set-class
   query-objects-list (append
      ( list ^^node-mesh-entities)
      ( list (get-independent-node (^body-master-RBE2)))
      ( list (get-independent-node (^body-slave-RBE2)))
      ( list (get-independent-node (^muscle-RBE2)))
   )
)
```

The get-independent-node method gets the independent nodes from the nodes that are not already part of the body mesh included in the node-mesh-entities. A node-query is used to get the nodes from the tri-mesh from the body, which is defined as the node-mesh-entities. It is important that the nodes have a coordinate property that has the position of the object.

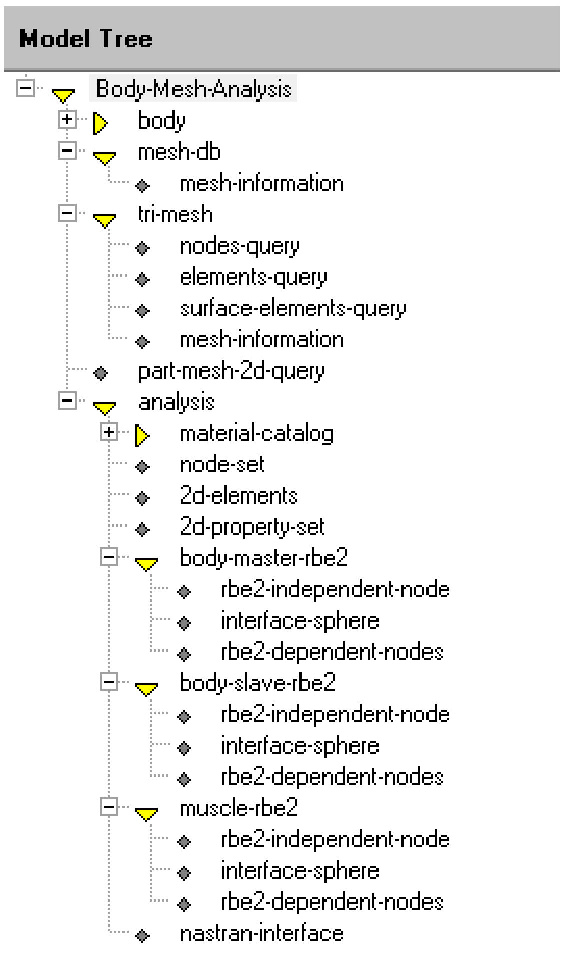
4.5.1 RBE2

The class inherited from to create the RBE2s is the built-in analysis-rigid-body-element-type-1-class. All the RBE2s inherit from the created generic class called RBE2-rigid-node-class. The node query objects need to be included to be able to write the RBE2 to the bdf files.

```
(define-class RBE2-rigid-node-class
   :inherit-from (analysis-rigid-body-element-type-1-class)
   :properties (
      independent-node-query-object (get-independent-node ^self)
      dependent-nodes-query-object (get-dependent-nodes ^self)
      dependent-degrees-of-freedom-list '(1 2 3 4 5 6)
      create-new-independent-node? nil ;;; Node already exists in part mesh.
   )
)
```

The methods defined on RBE2-node-properties-class can be re-used to create a generic class, which will be initiated in the body-mesh-analysis class. **Figure 4.8** shows the model tree from the body-mesh-analysis-class, where the important sub- objects to write the bdf file are the sub-objects of the analysis object. As explained above, the RBE2s inherit from the built-in generic analysis-rigid-body-element- type-1-class.

To perform the analysis and for the bulk data file (bdf) describing the body to be printed to the designated folder, the property from run-nastran@ from the nastran-interface object has to be called.



```
Model Tree
Body-Mesh-Analysis
    body
    mesh-db
        mesh-information
    tri-mesh
        nodes-query
        elements-query
        surface-elements-query
        mesh-information
    part-mesh-2d-query
    analysis
        material-catalog
        node-set
        2d-elements
        2d-property-set
        body-master-rbe2
            rbe2-independent-node
            interface-sphere
            rbe2-dependent-nodes
        body-slave-rbe2
            rbe2-independent-node
            interface-sphere
            rbe2-dependent-nodes
        muscle-rbe2
            rbe2-independent-node
            interface-sphere
            rbe2-dependent-nodes
        nastran-interface
```

Figure 4.8: AML graphics display of femur-scale-example

4.6 User interface

A simple user interface is added to gain more flexibility, without manually having to change the values. The class used to do this is the data-model-node-mixin, which is created to represent a node of the data model. In this application the most important property of this class is the property-object-list, which contains the instances that define the data model properties of the node.

A user interface made is to decide whether to use the model based on the gait2392 or the thoracolumbar. The difference between the two models is that the gait model has the spine modeled as one body, while the thoracolumbar spine is based on multiple bodies with joints between them. The class data-model-node-mixin in use can be seen on the work area when initiating the body-mesh-analysis to select the body and muscle point from the model in the graphics display, see **figure 4.9**. Whenever a value is changed, the green check button has to be pressed and the model has to be re drawn to see the changes.

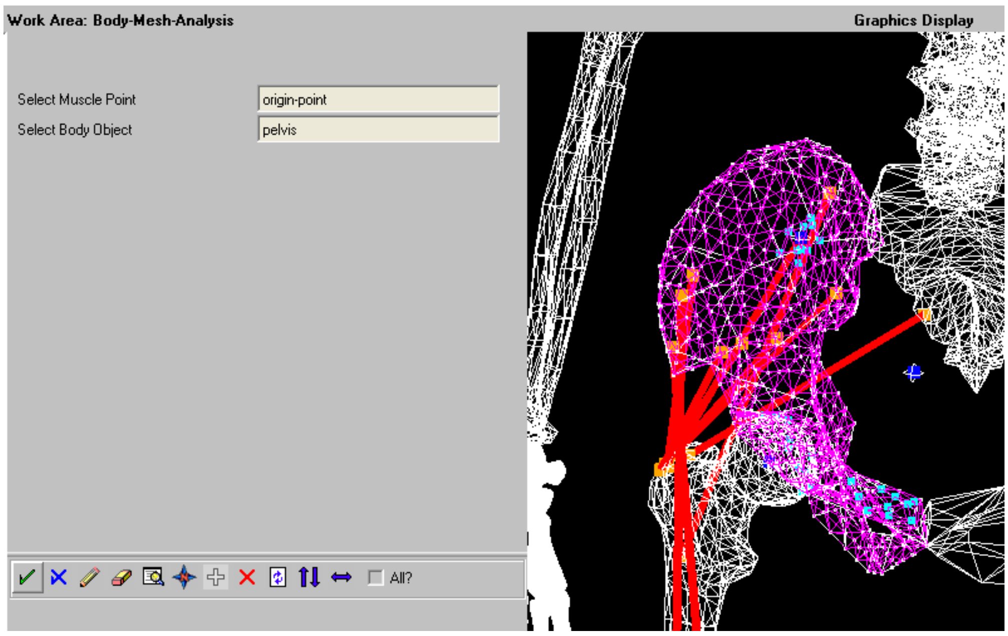


Figure 4.9: AML work area user interface

4.7 Fedem system files

In Fedem the main file with information about the system is the .fmm file. This file explains how all the bodies are connected together by joints. To have a complete joint two triads have to be connected to the joints. In this system the triades are the master-RBE2 and the slave-RBE2. To get an impression in how the fmm file should look like if generated in AML in the future, the model was put together manually with free joints.

4.8 Code refinement

Some code refinement was performed at the end of the thesis work. Due to the time constrains, this was not translated to the whole code but tested on parts of the code.

By creating tree generic classes that inherit to body-part-class, which would be a generic class that all bodies inherit from. For example, instead of the femur sub-object inheriting from tagging-class and femur-web-surface class, it would only inherit from body-part class.

```
(define-class body-part-tagging-class
   :inherit-from (
      tagging-object
   )
   :properties (
      max-elem-size (default nil)
      min-elem-size (default nil)
      tag-dimensions '(0 1 2 3)
      tag-attributes ( list  !max-elem-size !min-elem-size 1 0.1 0 20.0 1.0e-5)
   )
)


(define-class body-part-websurface-class
   :inherit-from (
      web-surface-object

   )
   :properties (
      cleanup? nil
      method (default 2)
      geometry-file-name (default (object-name ^self))
      nodes-file  (format nil "~a~a~a" "C:\\PytonWorkspace\\"(write-to-string
           ^geometry-file-name) "_points.dat" )
      con-file  (format nil "~a~a~a" "C:\\PytonWorkspace\\"(write-to-string
           ^geometry-file-name) "_connectivity.dat" )
   )
)

(define-class body-part-class
   :inherit-from (
      body-part-tagging-class
      body-part-websurface-class

   )
   :properties (
      body-translation-list (default '(0  0 0))
      body-rotation-list (default '(0  0 0))
      orientation  ( list
         (rotate  (radians-to-degrees (first ^body-rotation-list)) :x-axis)
         (rotate  (radians-to-degrees (second ^body-rotation-list)) :y-axis)
         (rotate  (radians-to-degrees (third ^body-rotation-list)) :z-axis)
         (translate  ^body-translation-list)
      )
```

```
        reference−coordinate−system (default nil)
        slave−RBE2−joint ^reference−coordinate−system
        master−RBE2−joint (default nil)

    )
)
```

As a result of this, there would no longer be any need for the web-surface- class, which alone would shorten the code by more than 1,000 lines and make it much easier to work with. The new sub-objects inheriting from body-part-class would only need to have the properties: body-translation-list, body-rotation-list, reference-coordinate-system, slave-RBE2-joint and master-RBE2-joint defined. This makes the code much easier to read and make further changes to.

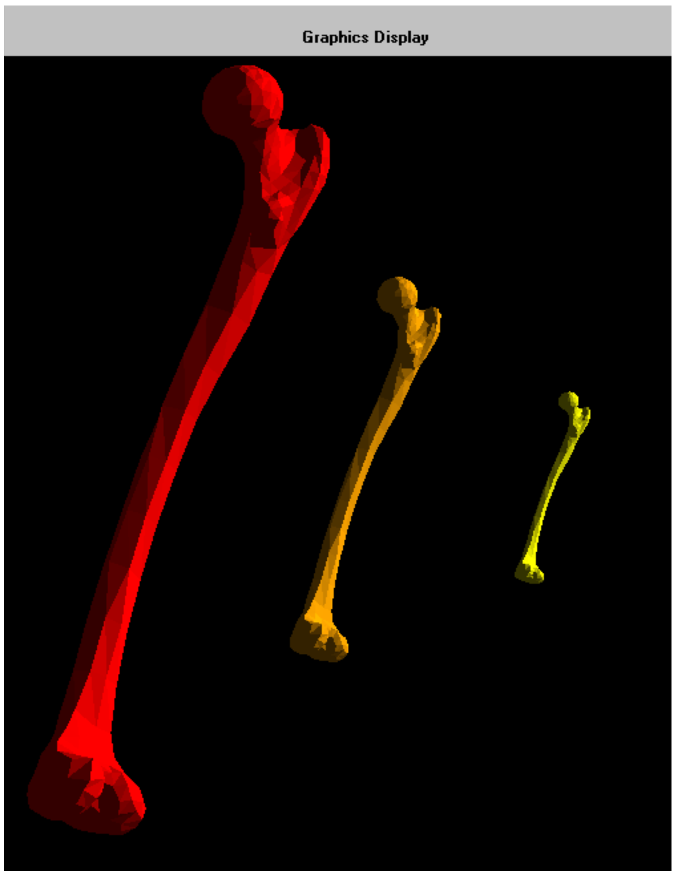


Figure 4.10: AML femur scaling example

Figure 4.10 illustrates how scaling of the model can be done by making a scale- property-class inherit from the built-in scale-object. The body-class inherits from the web-surface-class. The scale-example-class has to be initiated to see the geometry, where the tree femur sub-objects are defined by inheriting from the scale-property-class. The scale-property-class used in the illustration of the femur scaling looks like this:

```
(define-class scale-property-class
   :inherit-from (scale-object)
    :properties (
       source-object ^scale-object
        scale-factor (default 2.0)
   )

   :subobjects (
       (scale-object :class 'body-part-class

       )
   )
)
```

The whole code can be found in the appendix C.5. The red femur is scaled with a scale-factor of 4, the orange with a scale-factor of 2, while the yellow is the original femur.

Chapter 5

Results

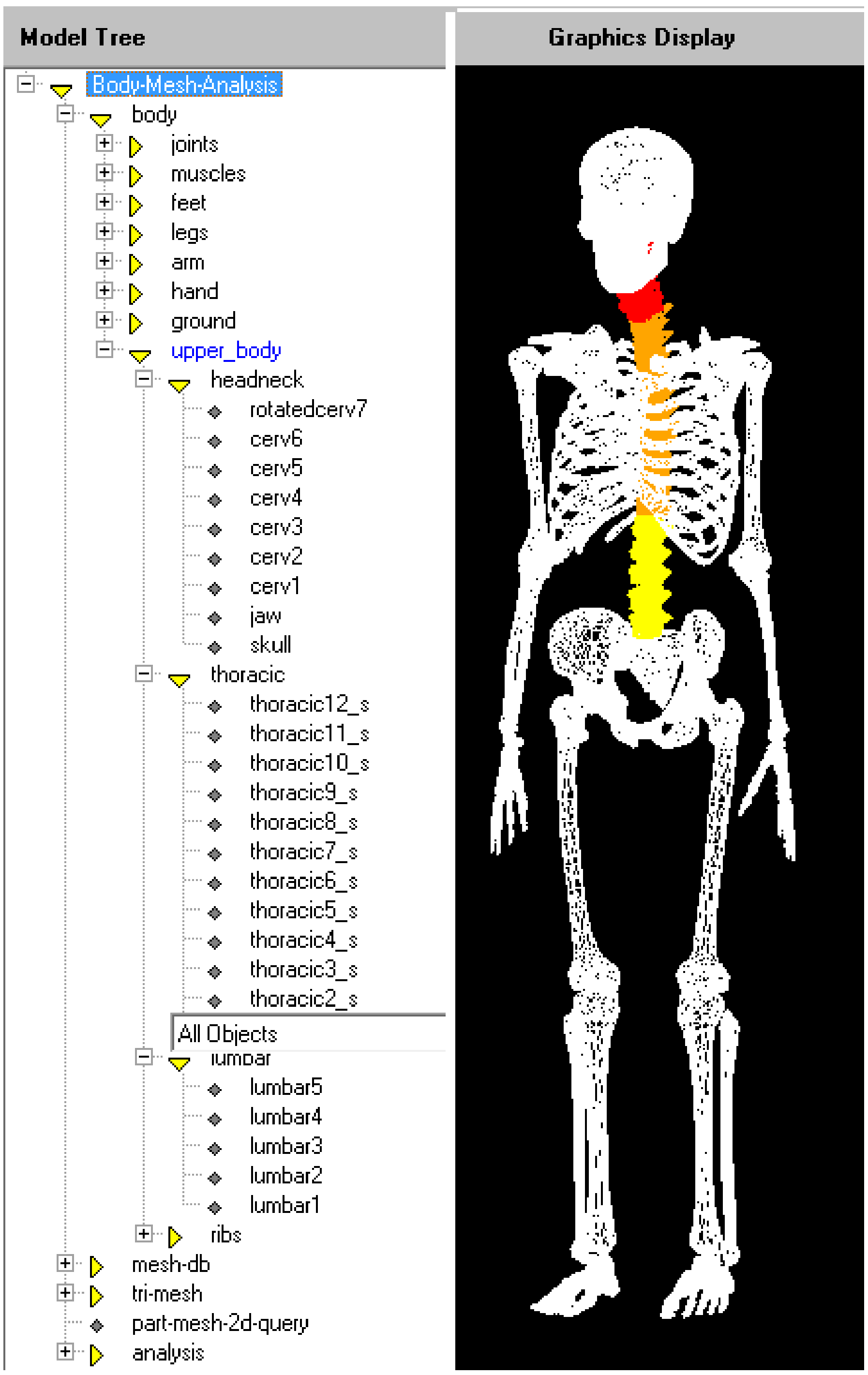
The objective of the work in this thesis is to create a multibody musculoskeletal-system in AML in order to perform further simulation in Fedem. Since the primary focus of this thesis has been on the technical development of the AML system, the results will represent this. The result of the work carried out on this thesis is firstly successfully transferring the geometry and joint locations of two OpenSim models. The two models is part of the system :musculoskeletal-system created. **Figure 5.1** shows the thoracolumbar spine. The geometry of this model is scaled compared to the original gait model, hence the femur joint is located differently than in the original gait model. From the model tree, the names of the bodies in the spine are listed. In red are the neck bodies, the orange bodies are the thoracic bodies, while the lumbar bodies are in yellow. All of the bodies are connected with joints.

The information about the OpenSim joints is pre-processed in Python to create a joint-class. The joint sub-object, which can be seen in **figure 5.1**, inherits from this joint-class. Even though the ribs' geometry is modeled as one rigid body, the rib joints can be found in the joints' sub-objects.

One muscle group was added in the gait model. More muscles could be added using the same generic muscle-class. Since the scaling is different in the two models, the muscle is scaled to the gait model's scaled pelvis. As can be seen in **figure 5.2** the muscle-line is inheriting form the built line-object-class and is just a line. The line is set to go from the origin point to the insertion point and, as can be seen in **figure 5.2**, the muscle will go through the bone, which is obviously undesired if the length of the muscle is used to calculate the muscle force.

The next feature of the model is the meshing and analysis. In **figure 5.3** you can see the femur meshed. The blue points are the independent nodes in the RBE2s, while the lighter blue is the dependent nodes. The number of dependent nodes can be adjusted in the RBE2-node-properties-class.

From the figure it is also visible how the mesh refinement affects the selection of the dependent nodes.The minimum and maximum mesh element sizes can easily



```
Body-Mesh-Analysis
  body
    joints
    muscles
    feet
    legs
    arm
    hand
    ground
    upper_body
      headneck
        rotatedcerv7
        cerv6
        cerv5
        cerv4
        cerv3
        cerv2
        cerv1
        jaw
        skull
      thoracic
        thoracic12_s
        thoracic11_s
        thoracic10_s
        thoracic9_s
        thoracic8_s
        thoracic7_s
        thoracic6_s
        thoracic5_s
        thoracic4_s
        thoracic3_s
        thoracic2_s
      All Objects
      lumbar
        lumbar5
        lumbar4
        lumbar3
        lumbar2
        lumbar1
      ribs
  mesh-db
  tri-mesh
  part-mesh-2d-query
  analysis
```

Figure 5.1: AML model with details of the spine

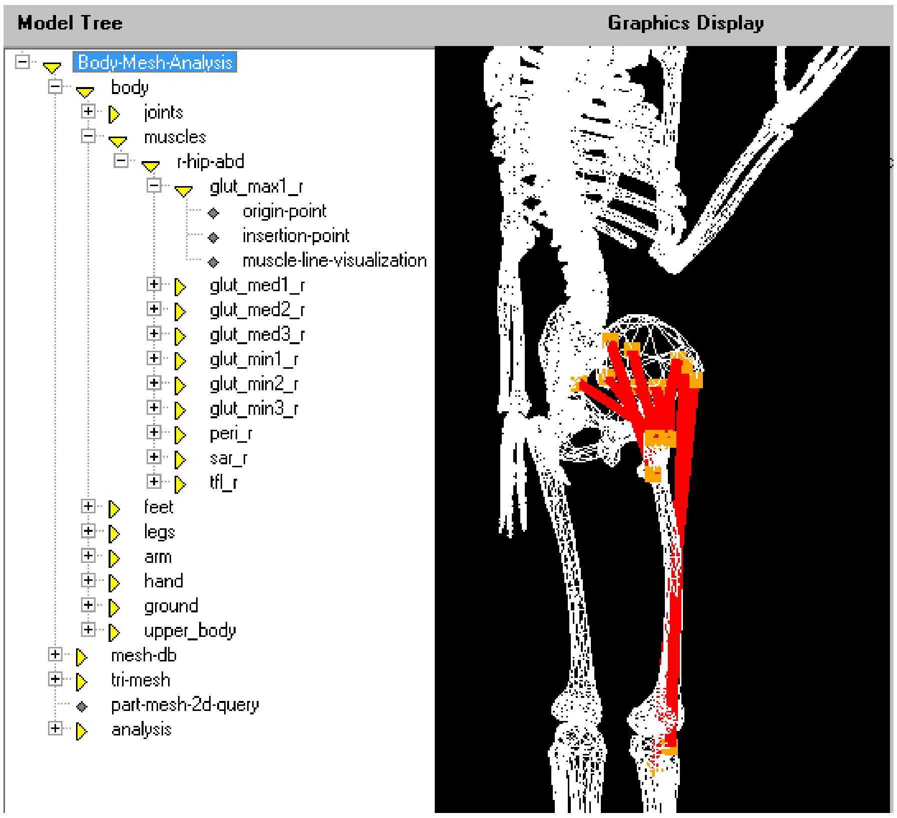


Figure 5.2: AML model with details of the muscle

be changed in the tagging-class or body-part-class. In the musculoskeletal system the class that needs to get initiated is the Body-Mesh-Analysis, as can be seen in the model tree in **figure 5.2** this class is the root. This class is a generic class that meshes the selected body. When the body is selected and meshed the independent and dependent nodes can be evaluated.

The independent and dependent nodes can be seen in the femur joint in **figure 5.3**. Because the mesh elements are so small in this case, the ten closest nodes are going to be very close to each other, causing the force from the joint to be concentrated in one small area.

The bdf file that gets written by the body-mesh-analysis class can be imported to Fedem by pressing the load part button and selecting the bodies. The bodies will automatically be set in the design position as defined in the AML model. The next step is adding the free joints in the joint center on the RBE2 and connecting the bodies together. The appendix A.1 and A.2 contains a caption of the RBE2 bdf file and a caption of the hip joint from the fmm file.

Figure 5.4 shows the parts of the model in Fedem modeler window. The

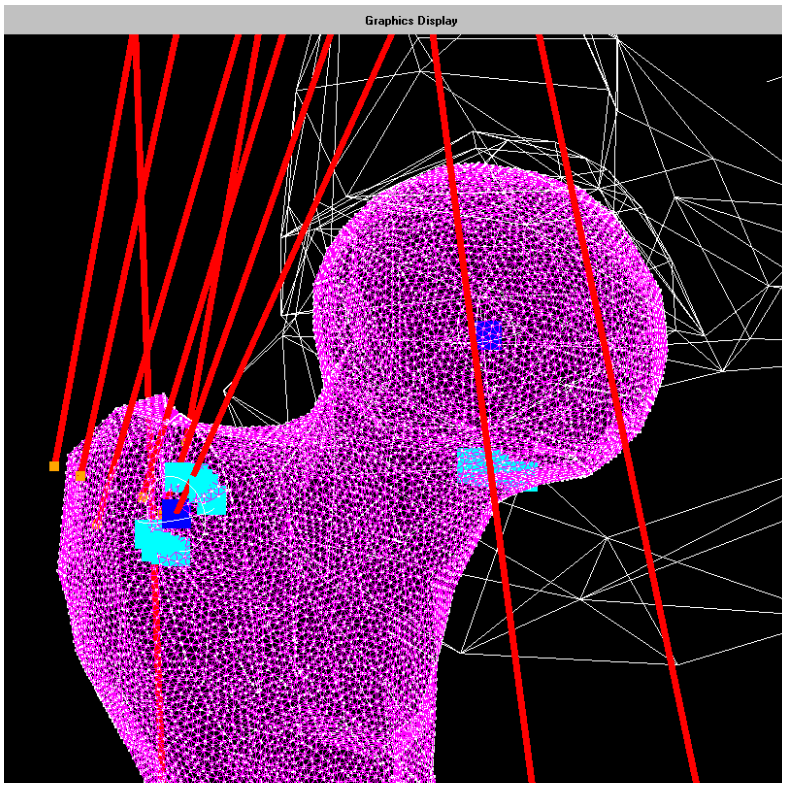


Figure 5.3: AML fine meshing of the femur and the dependent- and independent nodes

different bodies are illustrated by different colors. On the left, the springs joints and parts can be viewed. The connected joints are yellow and green when they are connected properly, as can be seen in figure **figure 5.4**. The spring is also connected properly, which can be seen from the same figure by the dependent nodes in the attachment turning green. To be able to do simulations, the model has to be connected to the ground. Equivalent to the OpenSim model, the pelvis bodies and sacrum are connected to the ground.

Figure 5.5 shows the hip master RBE2. Since the element size of the mesh is much bigger in this body, the independent nodes are spread out more evenly. The goal, however, is to have a finer mesh but using a method that spreads out the dependent nodes more realistically. **Figure 5.6** shows the muscle RBE2 in the Fedem modeler.

Figure 5.7 shows the full gait model as it is modelled in OpenSim except the knee joint, which is changed to the coordinates of the knee-model investigation. Obviously, the connection to ground with the RBE2 and connection from upper body to ground is unrealistic compared to the thoracolumbar spine model. The

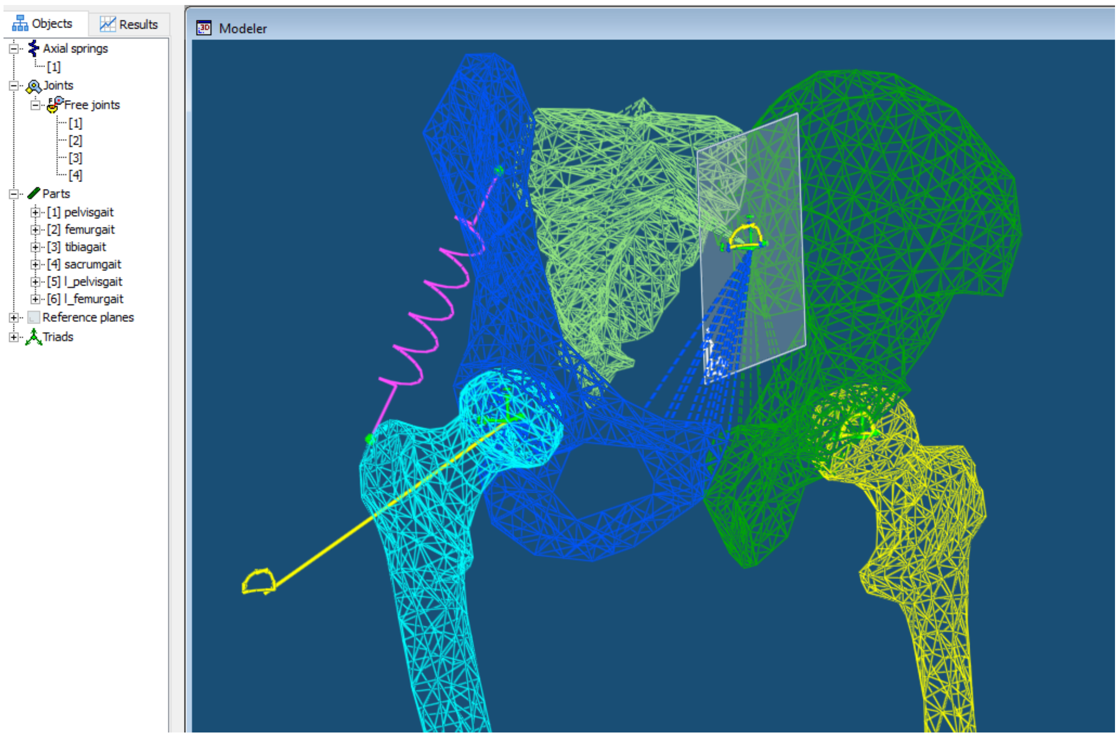


Figure 5.4: Ground joint and hip joint in Fedem. A spring is added to visualize how a muscle could look like.

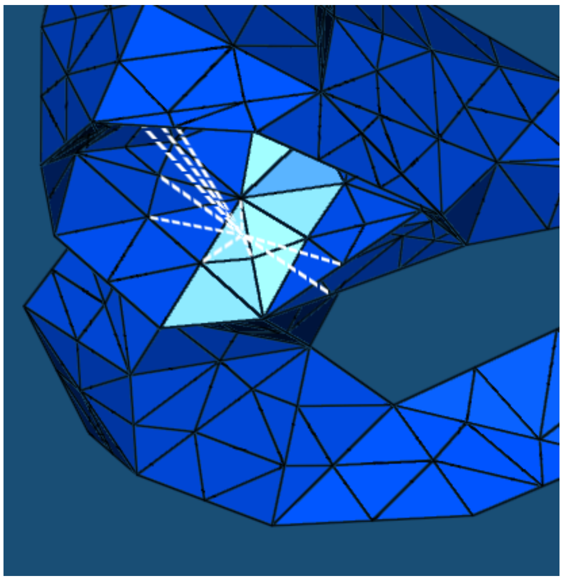


Figure 5.5: Pelvis free joint RBE2 in Fedem

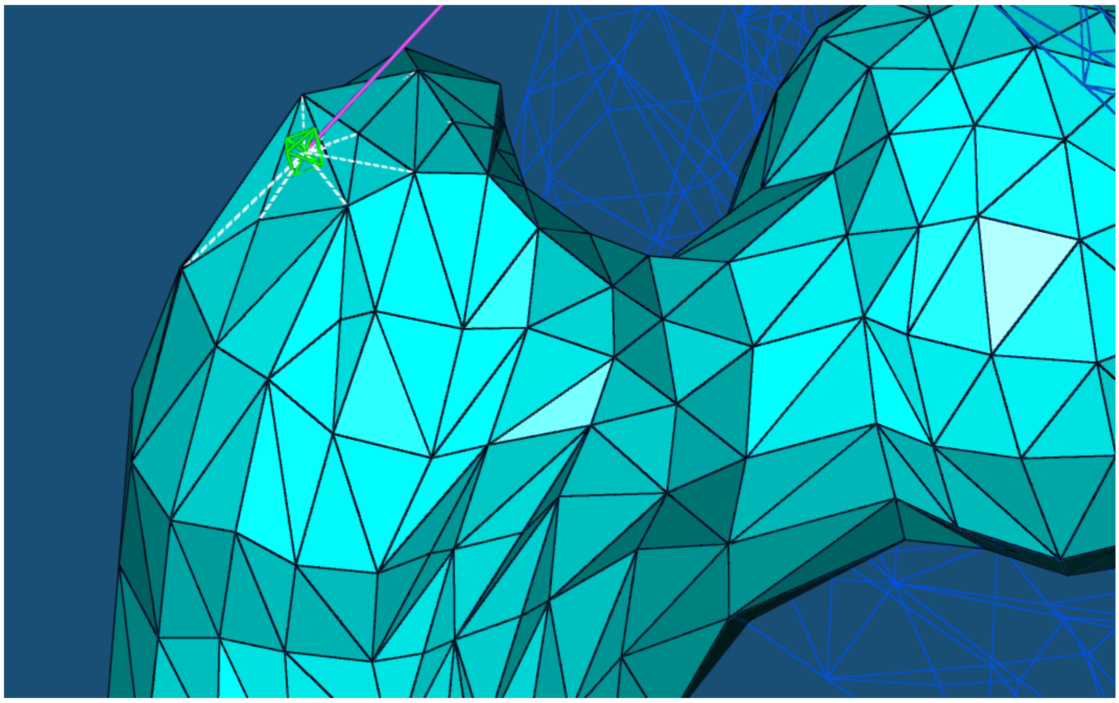


Figure 5.6: Muscle RBE2 in Fedem

main focus is to show how AML can be used to mesh and create a model to use in Fedem based on the OpenSim model.

Nevertheless, all the geometry and joints have been successfully meshed and put together in Fedem. Fibula was not added since it has no functional joint in the OpenSim and is only for visualization purposes. However, it could easily be added due to the fact that it is meshed in the AML model, but no free joint is added to connect it to the rest of the bodies, see **figure 4.3** for the body and joint structure of the gait2392 model.

Figure 5.8 shows the results of the new joint center of the knee joint, by using the joint-class-data-model-class to change the angles in the knee joint. The figure gives an indication of whether the new joint center have been implemented right.

To conclude, the result shows how the development of the body-structure has been effective to create two models based on the OpenSim models. Meshing and analysis classes have been developed to mesh the bodies and create RBE2s. Muscle-classes have been created to show how the points defined in the XML osim file can be transferred to the AML model, to be able to make the attachment rbe2 to the bodies. Lastly, it has been demonstrated that the model can easily be added to Fedem, even though the system file is not automatically written.

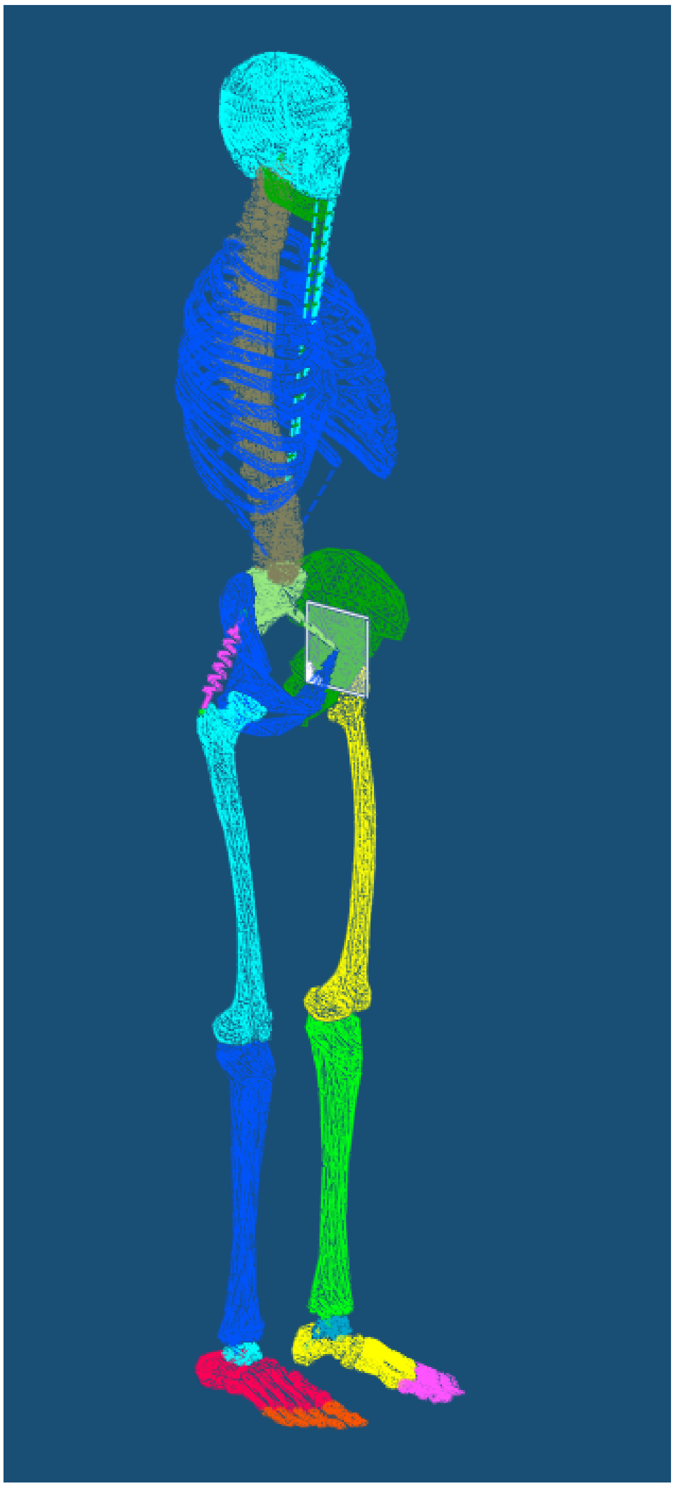
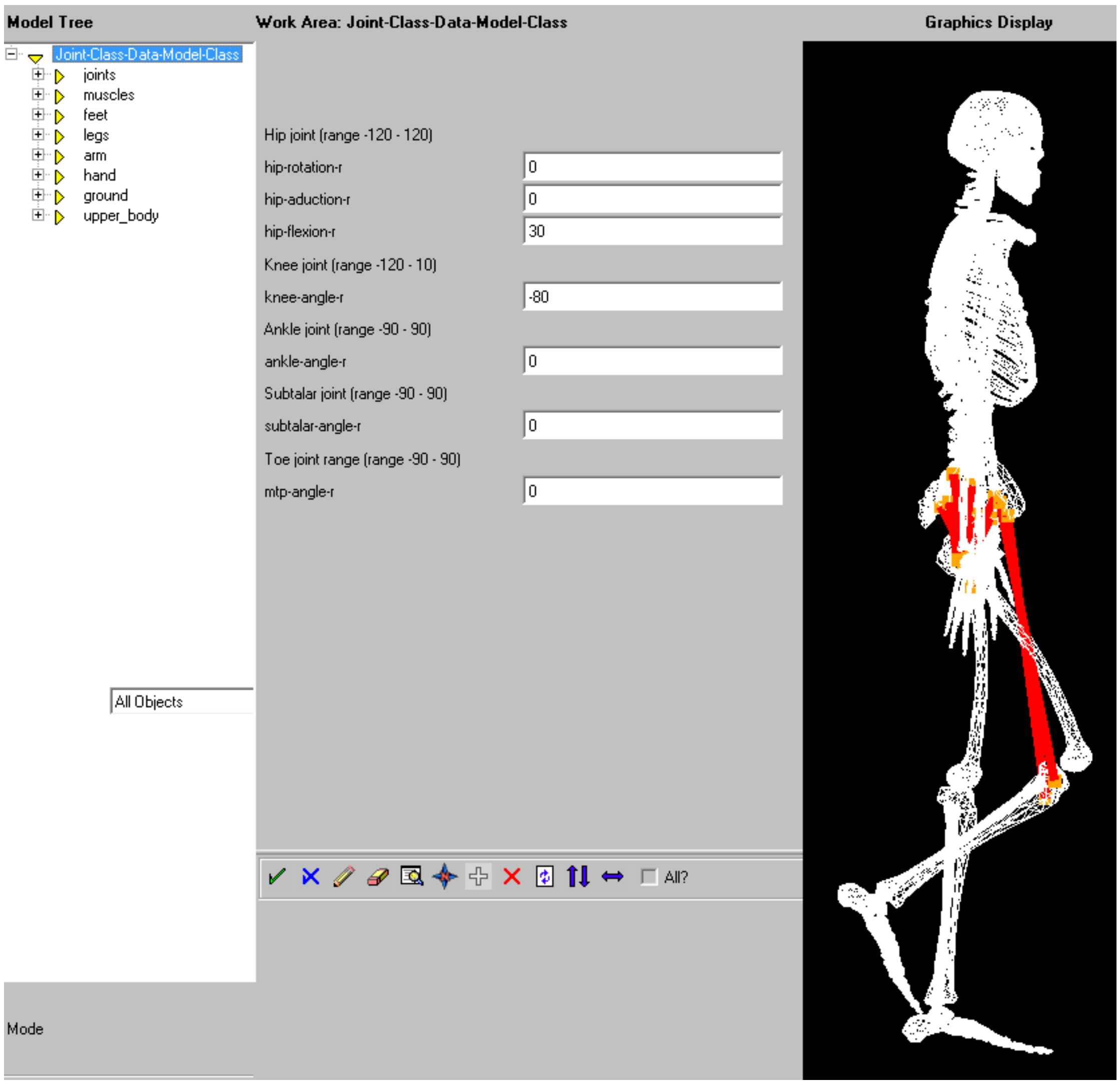


Figure 5.7: The gait model without arms in Fedem



Hip joint (range -120 - 120)

hip-rotation-r 0

hip-aduction-r 0

hip-flexion-r 30

Knee joint (range -120 - 10)

knee-angle-r -80

Ankle joint (range -90 - 90)

ankle-angle-r 0

Subtalar joint (range -90 - 90)

subtalar-angle-r 0

Toe joint range (range -90 - 90)

mtp-angle-r 0

Figure 5.8: AML knee-joint investigation with the new joint-center

Chapter 6

Discussion

6.1 Modelling

RQ1: Can a finite element model created in a Knowledge Based Engineering (KBE) framework improve musculoskeletal modeling?

Considering projects such as the NIH-founded finite element model of the foot [1] and the demu2neck [2] [1] founded by the 7FP (Seventh Framework Program, European Union research and development funding program) there is a potential for finite element method models of the musculoskeletal system. They state that there is a need for modelling the muscle and motion interaction, as well as the tissue and bone deformations.

In chapter 5 a finite element model of the bones and a muscle group is presented. It is difficult to get an impression of whether this model could be the start of a model that could be validated for simulation purposes. Nevertheless, what has been shown is the ability of the AML language to use available surface data to generate joints and finite element bodies. One of the challenges when performing musculoskeletal modeling is exactly this translation. The process of changing between the OpenSim modeling software to use the results in a CAD model for finite element methods is time-consuming.

Despite of this, finite element models of the whole body are not new. Full body models have been created to be used in simulations such as car crashes to investigate car safety [14]. The models are created to investigate high impact on the body. However, muscular activity along with the joint motion is not accounted for for in these simulations.

The most used muscle model is the hill-type model which is modeled as a spring

[1]http://cordis.europa.eu/project/rcn/95638$_e n.html$

in parallel with a counteractive element, and in series with another spring [9]. One of the geometry inputs of this model is the length of the muscle tendon along with physical variables such as velocity and subject-specific variables.

In this thesis work the modelling of the positioning of the muscle's starting and ending points has been demonstrated in AML. The rest of the muscles are available in the OpenSim model file. Adding them to the AML file is a matter of writing the muscle-class in the same structure as presented in chapter 4. However, there are some modelling issues that should be discussed before further implementation.

Firstly, the path of the muscle is important for the force generating capability of the muscle. In OpenSim the path is modelled with via-points and a wrapping function, as presented in chapter 2.7. The wrapping function accounts for the movement of the muscle due to motion and other tissues. If the hill type model is chosen a way to model the muscle path have to be found. The actual muscle modelling needs to be investigated in terms of how the model should contribute to the simulation. If the simulation is going to be stable, the passive muscles have to give some kind of feedback in the simulation.

Another but much smaller issue is writing the code so that all the bodies are inheriting from the analysis class automatically, including all the muscle points.The muscle points that "belong" to each body need to be saved in a list in so they can be saved as a RBE2 in the body mesh. This is necessary for the RBE2 nodes to be included in the bdf file generated. A generic method for the body-part-class could be created to avoid manually selecting all the muscle points, which coordinates are fixed to the body.

The AML model should also have the ability to be scaled. In the chapter 4.8 a method for scaling the geometry was presented, which could be useful in further development. However this scaling is not going to accommodate scaling individual skeletal differences in the skeleton. As stated in chapter 2.3, the bones do not grow evenly. In OpenSim the models are scaled with scale factors similar to the scale factor used in the scaling example. Since the geometry is only used for visualization purposes in OpenSim they do not need to account for whether the bone geometry scaling will influence the bone properties.

One of the main advantages of using finite element bodies is the possibility to model the joint force distributions. This is important when modeling the skeletal to research skeletal diseases. To accommodate for this, RBE2's have been created to connect the joints, and for muscle attachments, see **Figure 5.5** and **Figure 5.6**. By using the RBE2, the force on the joint from muscles will be distributed into multiple nodes in the mesh. However, the added stiffness of the RBE2 will force the body to hold its shape, while adding a RBE3 instead could possibly give more realistic deformation of the joint connection. In this thesis work the positioning of the dependent nodes has the intention of exploring methods rather than representing the actual realistic pressure distribution in the joint.

Force and pressure distribution could be done by modeling the elastic tissue in the joints, as presented in chapter 2.5. This would require more extensive knowledge about the material, and the complex material properties would depend upon state-of-the-art bio-mechanical tissue modeling. Another possible way of doing this is allowing some translation in the joint type “free joint” controlled by high stiffness spring. This simplification would be much easier to model and is the reason why free joints are added instead of ball joints is the AML model. However, the success of this would depend on what level of detail the model is going to be used for.

Another issue faced with when merging the OpenSim framework with finite element modeling is that the joints in OpenSim define the relative motion between the bodies realistically, but do not represent the actual joint center. As a result of this, some of the RBE2s connecting the bodies in the AML models go through other bodies before connecting to the mesh. One example of this is in the foot of the model where one of the joints is defined in the most distant point of the heel, see **figure 6.1**.

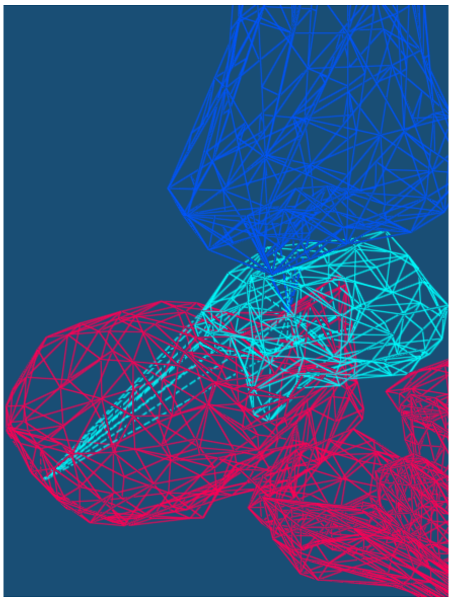


Figure 6.1: Heel caption from Fedem

To improve the model it is needed to establish whether a more accurate joint center representation needs to be used, or whether the OpenSim representation of the joint center position is realistic.

6.2 Simulations

The second research question is as follows:

RQ2: How can Fedem be used to perform musculoskeletal simulations?

In the thesis work the possibility of performing all simulations in Fedem without additional software has been investigated. Fedem is a validated software for performing multibody dynamics simulations. The challenge when using Fedem to carry out musculoskeletal modeling is to get the muscle output that can drive the simulations. In OpenSim this is done by inverse kinematics, inverse dynamics and static optimization, see chapter 2.6. In summary, these tools work by knowing the motion from experimental input and working backwards to establish which muscles are activated to produce the known motion.

From experience in doing OpenSim simulations, adding additional actuators in the joints is necessary. The added to the OpenSim model to account for residuals and additional muscle strength if the muscles cannot contribute enough. Similarly, the added actuator in the joint will account for the passive forces such as tendons in the model. In OpenSim there are possibilities to perform simulations with feedback loop to account for the instabilities in the model. The ideal option would be to model the passive forces in the model in a way that enables the tendons or tissue to have a feedback response to the motion in order to avoid the instability. One way of dealing with this issue could be to add some stiffness to the joints that could resemblance the passive tissue.

One way of carrying out the inverse kinematics type simulation in Fedem is to constrain the model and hence force it to follow the desired path. This is a major challenge due to the large variations in possible ways of moving from one position to another.

6.3 EU projects

Horizon2020 is the biggest EU research and innovation program. Funding can be achieved by creating proposals in response to the proposal calls. As part of the thesis work, some investigation has been carried out into whether there could be any proposals that would match a potential development of an AML musculoskeletal model. To get a proposal approved it is important that it fits the proposal call, but the proposed project must also have a health or economic impact for the EU.

One topic that could potentially be a good match is the "Personalised computer models and in-silico systems for well-being" (SC1-PM-17-2017) which is a research and innovation action call.

A summary of the topic description is ”Proposals should aim at the development of new integrative dynamic computer-models and simulation systems of acceptable validity, with the potential to being reused, build on open service platforms and with application in well-being, health and disease.” Furthermore, the specific challenge focuses on reducing healthcare needs, by working on well-being, prevention or rehabilitation. It is also stated in the scope of the topic that the “proposals will focus on multi-disciplinary research in medicine, social sciences and humanities(SSH) and information and communications technology (ICT) and should take advantage when relevant of existing large databases”.

There are several qualities with the AML musculoskeletal system that makes it applicable to the topic. The possibility to automate the processes associated with CAD modelling, makes it possible to save time during the modelling process.

Based on the Norwegian population health report (Folkehelserapporten)[15] from 2014 there are several clinical issues that could possibly benefit from more knowledge about the musculoskeletal system.

Challenges related to the growing proportion percentage of older people result in an increase in health issues related to old age. One example of such disease is osteoporosis, which can cause bone fractures. A population that to a lesser degree is in physical activity will contribute to an increase in musculoskeletal diseases in the future. Another contributor to the social challenges is the increase of obesity in the population, which again will cause skeletal problems.

Along with mental health, musculoskeletal diseases are the most common reason for sickness and disability benefits by NAV[2] [16]. Even if the musculoskeletal diseases do not often lead to death, the health and economic repercussions are huge [17].

To have a strong proposal there should be some exemplification of how the AML musculoskeletal system would contribute to the challenges mentioned above. Feasible research fields could be rehabilitation, design of medical devices, orthopedics and ergonomics.

One possible issue surrounding the probability of achieving funding is the open service platform requirement, which is evidently not covered by current licenses of the software. Hence, alternative software needs to be found, or some collaboration plan has to be made.

[2]The Norwegian Labour and Welfare Administration

Chapter 7

Conclusions

A conceptual model of the musculoskeletal model has been created in AML. The model takes advantages of the mesh analysis tools that are implemented in the AML software. This could be a starting point for future development of musculoskeletal modeling or just a showcase of the possibilities of the AML.

The model is based on the theory behind the OpenSim software and the geometry in the available models. Whereas some of the knowledge gained from OpenSim has been extremely useful, other parts have not been applicable to the AML model.

A simulation framework in Fedem has been explored and discussed. No simulations tests have been carried out to validate the model for simulations, but a Fedem model file has been created to continue the work in Fedem.

Chapter 8

Further work

Suggestions for further work are as follows: One main decision point is what muscle model should be used for moving forward. If the hill-muscle model is chosen, it has to be investigated how this can be represented in Fedem and it could be written to the fmm file from the AML application.

The next step is for the AML application to write the fmm file automatically. This is mainly an AML programing task only. All information required is saved in the model, it just needs to be obtained by a method or a class to write the file. The coordinates and the nodes are easily available. However, the file contains other Fedem-specific information that it is important to get right for the file to run in Fedem.

Scaling is an essential feature to the success of the application. If the model is decided to be scaled the same way as in OpenSim, some research have to be carried out to see how this will represent realistic bone properties. Furthermore, the mesh used has to be refined. State-of-the-art bone meshing standards need to be investigated to decide how to represent the different properties of the bone.

In AML the mesh attributes as element size could benefit from more automated mesh methods that could account for mesh refinement in sensitive areas such as muscle origins and insertion points. The independent node should not share the same dependent nodes, and this might be an issue where the muscle points are laying close to each other.

The pressure distribution in the joint is an essential part of the model and thus needs further research to create a realistic model. Tests should be conducted and validated against existing validated models. Depending on what the model is used for, the simplest scenario would be if the model was “good enough” without adding additional tissues in the joint. Finding new joint centre to get a more realistic joint and RBE2 should also be researched.

The issue of the floating geometry needs to be solved, either by creating rigid connections between them or by representing the body geometry in a different way. One option could be to add more tissue holding the bones together as seen in the finite foot model mentioned earlier, which had most of the tissues represented. This foot model is available to download and is in the iges format, for which AML has classes and methods for importing.

To know the success of the AML model it would require some sort of validation. Since the model is based on the OpenSim models, one way of starting the validation could be to run the forward dynamics in Fedem. By using muscle activation input computed using OpenSim in Fedem, the results should be similar to the results from a forward dynamics simulation in OpenSim.

To be independent of other software, some alternative to inverse kinematics and static optimization to find the muscle activation pattern should be further investigated in Fedem. The simulation output, which would be the muscle activation, could be compared to the muscle output from the simulations in OpenSim.

Lastly, it would be an advantage if the development focused on an area to model, possibly based on a clinical case. By doing this, one would avoid dealing with multiple models, and it would be easier to decide upon the level of detail necessary.

Bibliography

[1] Jason P Halloran, Ahmet Erdemir, and Antonie J van den Bogert. “Adaptive surrogate modeling for efficient coupling of musculoskeletal control and tissue deformation models”. In: *Journal of biomechanical engineering* 131.1 (2009), p. 011014.

[2] S Howley, D Bonneau, and B Fréchède. “A framework towards personalisation and active muscle integration in a 3D finite-element neck model for orthopaedic applications”. In: *Computer methods in biomechanics and biomedical engineering* 17.sup1 (2014), pp. 74–75.

[3] Olav Sand et al. *Menneskets fysiologi.* Gyldendal akademisk, 2014.

[4] Ahmed A Shabana. *Dynamics of multibody systems.* Cambridge university press, 2013.

[5] Daniel A Jacobs and Kenneth J Waldron. “Modeling Inelastic Collisions With the Hunt–Crossley Model Using the Energetic Coefficient of Restitution”. In: *Journal of Computational and Nonlinear Dynamics* 10.2 (2015), p. 021001.

[6] Jason P Halloran et al. “Comparison of deformable and elastic foundation finite element simulations for predicting knee replacement mechanics”. In: *Journal of biomechanical engineering* 127.5 (2005), pp. 813–818.

[7] Benjamin J Fregly, Yanhong Bei, and Mark E Sylvester. “Experimental evaluation of an elastic foundation model to predict contact pressures in knee replacements”. In: *Journal of biomechanics* 36.11 (2003), pp. 1659–1668.

[8] MSC Nastran. *MSC Nastran 2016 Linear Static Analysis User’s Guide.* 2016. URL: `https://simcompanion.mscsoftware.com/resources/sites/MSC/content/`.

[9] Martha Risnes. *Project thesis, KBE for Human Body Modeling and Simulation.* 2015.

[10] Wisnes AR. *Lærebok i biomekanikk.* Cappelen Damm Akademisk, 2013.

[11] Alexander G Bruno, Mary L Bouxsein, and Dennis E Anderson. “Development and validation of a musculoskeletal model of the fully articulated thoracolumbar spine and rib cage”. In: *Journal of biomechanical engineering* 137.8 (2015), p. 081003.

[12] Katherine RS Holzbaur, Wendy M Murray, and Scott L Delp. “A model of the upper extremity for simulating musculoskeletal surgery and analyzing neuromuscular control”. In: *Annals of biomedical engineering* 33.6 (2005), pp. 829–840.

[13] Anita N Vasavada, Siping Li, and Scott L Delp. “Influence of Muscle Morphometry and Moment Arms on the Moment-Generating Capacity of Human Neck Muscles”. In: *Spine* 23.4 (1998), pp. 412–422.

[14] FS Gayzik et al. “Development of a full body CAD dataset for computational modeling: a multi-modality approach”. In: *Annals of biomedical engineering* 39.10 (2011), pp. 2568–2583.

[15] Camilla Stoltenberg. “Folkehelserapporten 2014. Helsetilstanden i Norge”. In: (2015), p. 270. URL: `http://hdl.handle.net/11250/277374`.

[16] NAV. *Legemeldte sykefraværstilfeller 4 kv 2005-2014. Diagnose og kjønn. Antall.* URL: `https://www.nav.no/no/NAV+og+samfunn/Statistikk/Sykefravar+-+statistikk/Sykefravar`.

[17] Even Lærum et al. “Et muskel-og skjelettregnskap”. In: *Forekomst og kostnader knyttet til skader, sykdommer og plager i muskel-og skjelettsystemet [A Musculosceletal Accounting. Prevalence and Expenses Associated with Injuries, Diseases and Ailments of the Musculoskeletal System]* (2013).

Appendix A

Fedem files

A.1 Fedem .fmm file

Listing A.1: Caption of femur link and free joint from .fmm Fedem file

```
LINK
{
BASE_FTL_FILE = "femurgait.ftl";
BASE_ID = 22;
BUOYANCY = false;
B_MATRIX_FILE = "femurgait_B.fmx";
CACHED_CHECK_SUM = 1189717222 17902;
CENTER_OF_GRAVITY =
CART_X_Y_Z −0.06512261 −0.27253481 0.10048375
EUL_Z_Y_X 0.00000000 −0.00000000 0.00000000;
CENTER_OF_GRAVITY_POS_REF = 2 FcLINK;
CENTER_OF_GRAVITY_ROT_REF = 2 FcLINK;
COLOR = 0 1 1;
CONDENSE_OUT_COG = false;
COORDINATE_SYSTEM =
1.00000000 0.00000000 0.00000000 0.00000000
0.00000000 1.00000000 0.00000000 0.00000000
0.00000000 0.00000000 1.00000000 0.00000000;
DESCR = "femurgait";
EXPAND_MODE_SHAPES = true;
E_MATRIX_FILE = "femurgait_E.fmx";
FACTORIZE_MASS_MX_EIGENSOLV = true;
GEN_PART_STIFF_TYPE = DEFAULT_RIGID;
GEN_ROT_STIFF = 1000000000;
GEN_TRANS_STIFF = 1000000000;
G_MATRIX_FILE = "femurgait_G.fmx";
ID = 2;
IMPORTED_REDUCED_MATRICES = false;
INERTIA_REFERENCE = POS_CG_ROT_CS;
```

```
LINE_COLOR = 1 1 1;
LINK_CENTRIP_CORRECTION = MODEL_DEFAULT;
LINK_CS_POS_ALGORITHM = MODEL_DEFAULT;
LINK_RSD = <"femurgait",1,"fedem_reducer.fco","fedem_reducer.fop",
"fedem_reducer.res","femurgait. ftl ","femurgait_1. frs ","femurgait_B.fmx",
"femurgait_E.fmx","femurgait_G.fmx","femurgait_M.fmx","femurgait_S.fmx",
"femurgait_SAM.fsm">;
LOCATION3D_DATA = CART_X_Y_Z EUL_Z_Y_X;
L_MATRIX_FILE = "femurgait_L.fmx";
MASS = 0.00146718069806;
MASS_INERTIA = 3.68619832412e−05 2.13504724978e−06 3.67336836719e−05
    1.77639644289e−06 1.17176608381e−07 −2.03741157074e−06;
MASS_PROP_DAMP = 0;
MASS_SCALE = 1;
MESH_QUALITY = 2;
MESH_TYPE = FULL;
MINIMUM_MESH_SIZE = 0;
MODEL_TYPE = OFF;
M_MATRIX_FILE = "femurgait_M.fmx";
NUM_EIGVALS_CALC = 0;
NUM_GEN_MODES = 12;
ORIGINAL_FE_FILE = "nastran−interface\femurgait.bdf";
OUTLINE_ANGLE_THRESHOLD = 0.785398163397;
OVERRIDE_LINK_CHECKSUM = false;
POLYS_ON_POINTS_OFF = true;
RAM_USAGE_LEVEL = FULL_FE;
RECOVERY_MATRIX_SAVE_PRECISION = DOUBLE_PRECISION;
REDUCED_FTL_FILE = "femurgait.ftl";
SAM_DATA_FILE = "femurgait_SAM.fsm";
SHININESS = 0.8;
STIFFNESS_SCALE = 1;
STIF_PROP_DAMP = 0;
SUPPRESS_IN_SOLVER = false;
S_MATRIX_FILE = "femurgait_S.fmx";
TOL_EIGENVAL = 1e−08;
TOL_FACTORIZE = 1e−12;
TRANSPARENCY = 0;
USE_CONSISTENT_MASS_MATRIX = false;
USE_EXTERNAL_RESULT_FILE = false;
USE_GENERIC_DATA = false;
USE_MASS_CALCULATION = true;
VISUALIZE3D = false;
}

TRIAD
{
BASE_ID = 25;
COORDINATE_SYSTEM =
1.00000000 0.00000000 0.00000000 −0.07070000
```

```
0.00000000 1.00000000 0.00000000 −0.06610000
0.00000000 0.00000000 1.00000000 0.08350000;
FE_NODE_NO = 777;
ID = 1;
LOCAL_DIRECTIONS = GLOBAL;
LOCATION3D_DATA = CART_X_Y_Z EUL_Z_Y_X;
NDOFS = 6;
OWNER_LINK = 2;
}

FREE_JOINT
{
BASE_ID = 36;
COORDINATE_SYSTEM =
1.00000000 0.00000000 0.00000000 −0.07070000
0.00000000 1.00000000 0.00000000 −0.06610000
0.00000000 0.00000000 1.00000000 0.08350000;
FRICTION_DOF = 0;
ID = 1;
LOCATION3D_DATA = CART_X_Y_Z EUL_Z_Y_X;
MASTER_TRIAD = 1;
MOVE_MASTER_TRIAD_ALONG = true;
MOVE_SLAVE_TRIAD_ALONG = false;
ROT_FORMULATION = FOLLOWER_AXIS;
ROT_SEQUENCE = ZYX;
ROT_SPRING_CPL = NONE;
SLAVE_TRIAD = 2;
TRAN_SPRING_CPL = NONE;
VAR_QUADRANTS = 0 0 0;
X_ROT_STATUS = FIXED;
X_TRANS_STATUS = FIXED;
Y_ROT_STATUS = FIXED;
Y_TRANS_STATUS = FIXED;
Z_ROT_STATUS = FREE;
Z_TRANS_STATUS = FIXED;
}
```

A.2 Femur fedem .bdf file

Listing A.2: Caption of RBE2 in the femur .bdf file

```
GRID*              778                         −0.0965         −0.0719
*                0.1362
$ Multipoint Constraints
RBE2,1,776,123456,386,362,387,766,411,
,760,765,361,385,412
$ Multipoint Constraints
RBE2,2,777,123456,109,104,113,105,110,
,108,88,114,100,106
```

```
$ Multipoint Constraints
RBE2,3,778,123456,478,485,230,231,484,
,474,208,207,248,471
```

Appendix B

Matlab and python scripts

B.1 Matlab script for body seperation

Listing B.1: Matlab script for separating bodies in one vtp-file. The script can be further developed by separating the elements with different body number (delnr) and print this to new files.

```
fileID = fopen('foot_connectivity.dat');
C = textscan(fileID,'%f32 %f32 %f32 %f32');
fclose(fileID);
celldisp(C);

en=C{1};
to=C{2};
tre=C{3};
fire=C{4};
mat=zeros(length(en),1); %matrix investigated to find witch elements that belongs to witch
    body.

delnr=1;

for i=1:length(en);
  element=[en(i)+1,to(i)+1,tre(i)+1, fire(i)+1];

  for j=1:4;
       if isnan(element(j))==0;
          if mat(element(j)) ==0;
              mat(element(j))= delnr;
          else
          delnr=mat(element(j));
               if isnan(en(i))==0;
                   mat(en(i)+1)=delnr;
              end
               if  isnan(to(i))==0;
              mat(to(i)+1)=delnr;
              end
               if  isnan(tre(i))==0;
              mat(tre(i)+1)=delnr;
              end
               if  isnan( fire (i))==0;
              mat(fire(i)+1)=delnr;
              end
         end
       end
  end
delnr=delnr+1;
end
```

B.2 Matlab script for spine investigation

```
x_spline_points=[−2.0944 −1.74533 −1.39626 −1.0472 −0.698132 −0.349066 −0.174533
     0.197344 0.337395 0.490178 1.52146 2.0944;−0.0032 0.00179 0.00411 0.0041 0.00212
     −0.001 −0.0031 −0.005227 −0.005435 −0.005574 −0.005435 −0.00525];
y_spline_points=[−2.0944 −1.22173 −0.523599 −0.349066 −0.174533 0.159149 2.0944;
     −0.4226 −0.4082 −0.399 −0.3976 −0.3966 −0.395264 −0.396];


%curvex=fnplt(cscvn(x_spline_points));
%curvey=fnplt(cscvn(y_spline_points));
%plot(curvex(2,:),curvex(2,:),'o')
%axis equal

steg=240;
dx=4.5742/steg;
x=0:dx:(steg*dx);
fy=cscvn(y_spline_points);
ypoints=fnval(fy,x);


dx=6.5193/steg;
x=0:dx:(steg*dx);
fx=cscvn(x_spline_points);
xpoints=fnval(fx,x);

rad=−(2/3)*pi:(pi/18):(pi/18);
kordinater=[];

j=1;

for  i=1:steg;
```

```
        if  j==15;
            break;

       elseif  abs(xpoints(1,i)−rad(j))<0.009;
            kordinater(1,j)= rad(j);
            kordinater(2,j)= xpoints(2,i);
            j=j+1;
      end
end

j=1;

for  i=1:steg;
        if  j==15;
            break;

       elseif  abs(ypoints(1,i)−rad(j))<0.009;
            kordinater(3,j)= ypoints(2,i);
            j=j+1;
      end
end

j=1;
%for investigation  of knee angle change vs. change of angle
for  j=2:14;
   deltax= kordinater(2,j)−kordinater(2,(j−1));
   deltay=kordinater(3,j)−kordinater(3,(j−1));
   hyp=((deltax)^2+(deltay)^2)^0.5;
   deltavinkel=acos(deltax/hyp);
   kordinater(4,j)= deltavinkel∗(180/pi);

end


j=1;
for  j=3:14
    kordinater(5,j)=(kordinater(4,j)  − kordinater(4,j−1));
end

plot(kordinater (2,:) ,kordinater (3,:) ,'o')
axis  equal
hold  on

x=kordinater(2,:);  y=kordinater(3,:);
% circlefit  function
[xc,yc,R,a]  =  circfit (x,y);

plot(xc,yc,'g∗')
hold  on
```

```
theta = linspace(0,2*pi);
x = R*cos(theta) + xc;
y = R*sin(theta) + yc;
plot(x,y,'c')

 title ('Knee translation spline')
xlabel('[m] translation in x-diection')
ylabel('[m] translation in y-direction')


% file printing in case of cam joint use

%fileID = fopen('kordinater.txt','w');
%for j=1:14;
%fprintf(fileID,'%6.2f %12.8f %12.8f\r\n',kordinater(1,j),kordinater(2,j),kordinater(3,j));
%end
%fclose(fileID);

%fileID = fopen('globalekordinater.txt','w');
%for j=1:14;
%fprintf(fileID,'%6.2f %12.8f %12.8f
    %12.8f\r\n',kordinater(1,j),kordinater(2,j)-0.0707,kordinater(3,j)-0.0661,0.0835);
%end
%fclose(fileID);
```

B.3 Python joint class XML to AML parser

```
import xml.etree.ElementTree as ET
from os import path

#input path
path_input ={
    'gait':'C:\OpenSimWorkspace\Models\Gait2392_Simbody\gait2392_simbody.osim',
'thoracolumbar':'C:\OpenSimWorkspace\Models\Thoracolumbar_OSIM_V1
\Thoracolumbar_Spine_With_RibCage.osim'}

#input: gait or thoracolumbar
model= 'gait'

tree = ET.parse(path_input[model])
root = tree.getroot()

def write_class(class_name, cont):
    with open(path.relpath(class_name + '.aml'), 'w') as f:
        f.write(cont)
    print 'Class: "' + class_name + '" written to AML file.'
    return 'done'
```

```
aml_header = '(in-package :aml)\n\n'

 joint_class  = "(ground-joint :class  'coordinate-system-class\n" +\
                               'origin  ( list   0 0 0 )\n' +\
                               ')\n\n'

for body in root.findall('.//Body'):
    for location_in_parent in body.findall('.//location_in_parent'):
        for orientation_in_parent in body.findall('.//orientation_in_parent'):
            for parent_body in body.findall('.//parent_body'):
                x, y, z =orientation_in_parent.text.split()
                 joint_class  += '(' + body.attrib['name'] + '-joint  :class
                     \'coordinate-system-class\n' + \
                "orientation  ( list  (rotate (radians-to-degrees " + x + " ) :x-axis)(rotate
                     (radians-to-degrees " + y + " ) :y-axis)(rotate (radians-to-degrees "
                     + z + " ) :z-axis)(translate (list " + location_in_parent.text + " )))\n"
                     + \
                'reference-coordinate-system ^^' + parent_body.text + '-joint \n'+\
                'display? ^^display-coord-systems?\n' +\
                'length ^^coordinate-system-length\n' +\
                ')\n\n'

 class_definition  ='(define-class joint-class\n' + \
                'inherit-from (\n' + \
                'object\n' + \
                ')\n' + \
                ':properties  (\n' + \
                'display-coord-systems? (default nil)\n' + \
                'coordinate-system-length 0.1\n' + \
                ')\n' + \
                ':subobjects  (\n' + \
                 joint_class + \
                ')\n' + \
                ')'

aml_code = aml_header + \
          class_definition

 write_class(' joint_class_gait ', aml_code)
```

B.4 Python body classes XML to AML parser

```
import xml.etree.ElementTree as ET
from os import path

#input path
path_input ={
    'gait':'C:\OpenSimWorkspace\Models\Gait2392_Simbody\gait2392_simbody.osim',
'thoracolumbar':'C:\OpenSimWorkspace\Models\Thoracolumbar_OSIM_V1
\Thoracolumbar_Spine_With_RibCage.osim'}

#input: gait or thoracolumbar
model= 'gait'

tree = ET.parse(path_input[model])
root = tree.getroot()

#functions to help making body_structure
def getbodies():
    geometry_list_helper=[]
    for body in root.findall('.//Body'):
            geometry_list_helper.append(body.get('name'))
    return geometry_list_helper

#function that prints body name and corresponding geometry files
def geometryfiles ():
    for body in root.findall('.//Body'):
        print '   ______________________   '
        print body.get('name')
        for geometry_file in body.findall('.//geometry_file'):
            geometry_file = geometry_file.text
            if geometry_file is not None:
                print geometry_file

#input: treestructure of bodies Thoracolumbar
ground=['ground', 'sacrum', 'pelvis']
lumbar=['lumbar3', 'lumbar4', 'lumbar3', 'lumbar2', 'lumbar1', 'lumbar5']
thoracic=['thoracic12', 'thoracic11', 'thoracic10', 'thoracic9', 'thoracic8', 'thoracic7',
    'thoracic6', 'thoracic5', 'thoracic4', 'thoracic3', 'thoracic2', 'thoracic1']
rib=['sternum','rib12_R', 'rib11_R', 'rib10_R', 'rib9_R', 'rib8_R', 'rib7_R', 'rib6_R',
    'rib5_R', 'rib4_R', 'rib3_R', 'rib2_R', 'rib1_R', 'rib12_L', 'rib11_L', 'rib10_L',
    'rib9_L', 'rib8_L', 'rib7_L', 'rib6_L', 'rib5_L', 'rib4_L', 'rib3_L', 'rib2_L',
    'rib1_L',]
abd=[ 'Abdomen','Abd_L_L1', 'Abd_L_L2', 'Abd_L_L3', 'Abd_L_L4', 'Abd_L_L5', 'Abd_R_L1',
    'Abd_R_L2', 'Abd_R_L3', 'Abd_R_L4', 'Abd_R_L5']
arm=['clavicle_R', 'scapula_R', 'humerus_R', 'ulna_R', 'radius_R',' clavicle_L ',
    'scapula_L', 'humerus_L', 'ulna_L', 'radius_L']
hand=['hand_R', 'hand_L']
headneck=['head_neck']

body_structure_thoracolumbar={'ground':ground, 'lumbar':lumbar,'thoracic':thoracic,
    'rib':rib, 'abd':abd, 'arm':arm,'hand':hand,'headneck':headneck }

#input: treestructure of bodies gait2392
ground=['pelvis']
upper_body=['torso']
```

```
feet=['talus_r', 'calcn_r', 'toes_r', 'talus_l', 'calcn_l', 'toes_l']
legs=['femur_r', 'tibia_r', 'femur_l', 'tibia_l']

body_structure_gait2392={'ground':ground, 'upper_body':upper_body,'feet':feet, 'legs':legs }

body_structure={'gait':body_structure_gait2392,
    'thoracolumbar':body_structure_thoracolumbar }
body_structure= body_structure[model]

#function for writing sub-geometry-class
def write_sub_geometry_class(structure, part, root):

    def write_class(class_name, cont):
        with open(path.relpath(class_name + '.aml'), 'a') as f:
            f.write(cont)
            print 'Class: "' + part + '" written to AML file.'

    sub_geometry_class=" "
     reference_object_list =" "
    for body in root.findall('.//Body'):
        if body.get('name') in structure[part]:
            reference_object_list  +='' + body.attrib['name'] + '-reference-joint (default
                nil)\n'
            for DisplayGeometry in body.findall('.//DisplayGeometry'):
                for geometry_file in DisplayGeometry.findall('.//geometry_file'):
                    geometry_file = geometry_file.text
                    if geometry_file is not None:
                        geometry_file= geometry_file.replace(".vtp","",1)
                        for  transform in  DisplayGeometry.findall('.//transform'):
                            rx, ry, rz, x, y, z =transform.text.split()
                            sub_geometry_class += "(" + geometry_file + " :class ' " +
                                geometry_file + "-web-surface \n" + \
                            'orientation (list (rotate (radians-to-degrees ' + rx + ' )
                                :x-axis)(rotate (radians-to-degrees ' + ry + ' )
                                :y-axis)(rotate (radians-to-degrees ' + rz +' )
                                :z-axis)(translate (list '+ x +' '+ y +' '+ z +'))) \n' + \
                            'reference-coordinate-system ^^' + body.attrib['name'] +
                                '-reference-joint \n' +\
                            ')\n'

     class_definition  = '(define-class '+ part +'-sub-geometry-class \n' + \
                        ':inherit-from (\n' + \
                        'object\n' + \
                        ')\n' + \
                        ' :properties (\n' + \
                         reference_object_list  +\
                        ')\n' + \
                        ':subobjects (\n' + \
                        sub_geometry_class + \
                    ')\n' + \
                    ')\n'

    aml_code = class_definition

    write_class ('sub−geometry_class_gait' , aml_code) #aml_code is the string created from
        functions defined to fit aml classes

#function for writing body−class
def write_body_class(structure, root):

    def write_class (class_name, cont):
        with open(path.relpath(class_name + '.aml'), 'a') as f:
            f.write(cont)
            print 'Class: body−class written to AML file.'

    class_definition = '(define−class body−class \n' + \
                    ':inherit−from (\n' + \
                    'object\n' + \
                    ')\n' + \
                    ' :properties (\n' + \
                    ')\n' + \
                    ':subobjects (\n' +\
                    "(joints: class 'joint−class \n" +\
                    ')\n'


    aml_code = class_definition

    write_class (' body_class_gait ', aml_code)

    geometry_list = " "
    for part in structure:
        geometry_list = "(" + part + " :class '" + part +"−sub−geometry−class \n"
        reference_object_list = " "
        for body in root. findall ('.//Body'):
            if body.get('name') in structure[part]:
                reference_object_list +='' + body.attrib['name'] + '−reference−joint (the
                    ' + body.attrib['name'] + '−joint (:from ^^joints))\n'



        class_definition = geometry_list +\
                        reference_object_list +\
                        ')\n'


        aml_code = class_definition
        write_class('body_class_gait', aml_code)


#write to file. Remeber to comment out if you dont want to keep writing to file
write_body_class(body_structure, root)

#for body_part in body_structure:
    #write_sub_geometry_class(body_structure, body_part, root)
```

B.5 Python surface geometry class XML to .dat file parser

```
import xml.etree.ElementTree as ET
from os import path


#C:\OpenSimWorkspace\Models\Gait2392_Simbody\gait2392_simbody.osim
#C:\OpenSimWorkspace\Models\Thoracolumbar_OSIM_V1\
#Thoracolumbar_Spine_With_RibCage.osim

tree = ET.parse('C:\OpenSimWorkspace\Models
\Thoracolumbar_OSIM_V1\Thoracolumbar_Spine_With_RibCage.osim')
root = tree.getroot()

# Print to shell

for body in root.findall('.//Body'):
    print '  ------------------------  '
    print body.get('name')
    for geometry_file in body.findall('.//geometry_file'):
        geometry_file = geometry_file.text
        print geometry_file




# Write to AML file (first level: Body parts)


def write_class(class_name, cont):
    with open(path.relpath(class_name + '.aml'), 'a') as f:
        f.write(cont)
    print 'Class: "' + class_name + '" written to AML file.'
    return 'done'

aml_header = '(in-package :aml)\n\n'

body_web_surface_class=''
```

```
for body in root.findall('.//Body'):
    for geometry_file in body.findall('.//geometry_file'):
        geometry_file = geometry_file.text
        if geometry_file is not None:
            geometry_file= geometry_file.replace(".vtp","",1)
            body_web_surface_class += '(define-class '+ geometry_file + '-web-surface \n '
                 +\
                 ':inherit-from(web-surface-object) \n' +\
                 ' :properties ( \n' +\
                 r'nodes-file "C:\\PytonWorkspace\\' + geometry_file +'_points.dat" \n' +\
                 r'con-file  "C:\\PytonWorkspace\\' + geometry_file +'_connectivity.dat"
                      \n' +\
                 'cleanup? nil \n' +\
                 'method 2 \n' +\
                 ')\n' +\
            ')\n\n'

aml_code = aml_header + \
            body_web_surface_class
write_class('web_surface_class', aml_code)
```

Appendix C

AML source code

C.1 Body-part-class from code refinement

Listing C.1: Code refinement classes of the body-part-class

```
(define-class body-part-tagging-class
   :inherit-from (
      tagging-object
   )
   :properties  (
      max-elem-size (default nil)
      min-elem-size (default nil)
      tag-dimensions '(0 1 2 3)
      tag-attributes ( list  ^^max-elem-size ^^min-elem-size 1 0.1 0 20.0 1.0e-5)
   )
)


(define-class body-part-websurface-class
   :inherit-from (
      web-surface-object

   )
   :properties  (
      cleanup? nil
      method (default 2)
      geometry-file-name (default (object-name ^self))
      nodes-file  (format nil "~a~a~a" "C:\\PytonWorkspace\\"(write-to-string
          ^geometry-file-name) "_points.dat" )
      con-file (format nil "~a~a~a" "C:\\PytonWorkspace\\"(write-to-string
          ^geometry-file-name) "_connectivity.dat" )
   )
)
```

```
(define−class body−part−class
   :inherit−from (
      body−part−tagging−class
      body−part−websurface−class
   )
   :properties (
      body−translation−list (default '(0 0 0))
      body−rotation−list (default '(0 0 0))
      orientation ( list
         (rotate (radians−to−degrees (first ^body−rotation−list)) :x−axis)
         (rotate (radians−to−degrees (second ^body−rotation−list) ) :y−axis)
         (rotate (radians−to−degrees (third ^body−rotation−list)) :z−axis)
         (translate ^body−translation−list)
      )
   )
)

(define−class arm−sub−geometry−class
   :inherit−from (
      object
   )
   :properties (
      clavicle_R−reference−joint (default nil)
      scapula_R−reference−joint (default nil)
      humerus_R−reference−joint (default nil)
      ulna_R−reference−joint (default nil)
      radius_R−reference−joint (default nil)
      clavicle_L−reference−joint (default nil)
      scapula_L−reference−joint (default nil)
      humerus_L−reference−joint (default nil)
      ulna_L−reference−joint (default nil)
      radius_L−reference−joint (default nil)
   )
   :subobjects (
      ( clavicle :class 'body−part−class
         reference−coordinate−system ^^clavicle_R−reference−joint
      )
      (scapula :class 'body−part−class
         reference−coordinate−system ^^scapula_R−reference−joint
      )
      (humerus :class 'body−part−class
         reference−coordinate−system ^^humerus_R−reference−joint
      )

   )
)
```

C.2 Meshing and analyse classes

Listing C.2: Classes for initiating the meshing and analysis

```
(define−class RBE2−node−properties−class
   :inherit−from (object)
   :properties (
      RBE2−origin (default nil)
      tri−mesh−entities (the surface−elements−query (:from ^tri−mesh))
      node−mesh−entities (the nodes−query (:from ^tri−mesh))
   )
   :subobjects (
      (RBE2−independent−node :class '(mesh−node−class object)
         mesh−database−object        ^^mesh−db
         color                        'blue
         line−width                   10
         coordinates           ( first  (the position (:from  ^^RBE2−origin)))
      )

      (interface−sphere  :class  'sphere−object
         diameter 0.01
         reference−object  ^^RBE2−origin
      )

      (RBE2−dependent−nodes :class 'mesh−query−nodes−from−interface−class ;;; <−−
          INTERFACE PLANE
         mesh−database−object          ^^mesh−db
         interface−object              ^^interface−sphere
         tolerance                     10000 ;; Max allowable distance from plane to
              nodes...
         quantity                      10 ;; −> get all nodes within tolerance
         subset−mesh−query−object−list (list ^^tri−mesh−entities)
         color                          'cyan
         line−width                     6
      )
   )
)

(define−method get−independent−node RBE2−node−properties−class()
   !RBE2−independent−node
)

(define−method get−dependent−nodes RBE2−node−properties−class()
   !RBE2−dependent−nodes
)
(define−class RBE2−rigid−node−class
   :inherit−from (analysis−rigid−body−element−type−1−class)
   :properties (
      independent−node−query−object (get−independent−node ^self)
```

```
        dependent−nodes−query−object (get−dependent−nodes ^self)
        dependent−degrees−of−freedom−list '(1 2 3 4 5 6)
        create−new−independent−node?  nil ; ;; Node already exists in part mesh.
    )
    :subobjects (
    )
)

(define−class body−analysis−class
    :inherit−from (analysis−model−class)
    :properties (
        property−set−objects−list (list ^2D−property−set)
        mesh−model−object (default nil)
        analysis−type (default nil)
        materials−list ( list 'steel)
        node−set−objects−list ( list ^node−set)
        element−set−2d−objects−list (list ^2d−elements)
        rigid−body−element−objects−list (list ^body−slave−RBE2 ^body−master−RBE2
              ^muscle−RBE2) ; ^body−master−RBE2 ^muscle−RBE2)
        material−catalog−object ^material−catalog

        selected−body (default nil)
        node−mesh−entities (default nil)
    )
    :subobjects (
        (material−catalog :class 'material−catalog−class
        )
        (node−set :class 'analysis−node−set−class
            test (get−independent−node−coordinates (^^mesh−model−object))
            testb (get−independent−muscle−node−coordinates
                  (get−muscle−mesh−model−object ^^mesh−model−object))
            query−objects−list (append
                ( list ^^node−mesh−entities)
                ( list (get−independent−node (^body−master−RBE2))) ;uncomment if body do
                     not have master joint
                ( list (get−independent−node (^body−slave−RBE2)))
                ( list (get−independent−node (^muscle−RBE2))) ;uncomment if body do not
                     have muscle point
            )
        )

        (2d−elements :class 'analysis−element−set−2d−type−1−class
            query−objects−list ( list ^^tri−mesh−entities)
            property−set−object ^^2D−property−set
        )
        (2D−property−set :class 'analysis−property−set−2d−type−1−class
            material−catalog−object ^^material−catalog
            material−name            "Steel"
            thickness                0.1
        )
        (body−master−RBE2 :class '(RBE2−node−properties−class RBE2−rigid−node−class)
             ;uncomment if body do not have master joint
           RBE2−origin ^^master−joint−reference−object
        )
        (body−slave−RBE2 :class '(RBE2−node−properties−class RBE2−rigid−node−class)
           RBE2−origin ^^slave−joint−reference−object
        )
        (muscle−RBE2 :class '(RBE2−node−properties−class RBE2−rigid−node−class)
             ;uncomment if body do not have muscle point
           RBE2−origin ^^selected−muscle
        )
        (nastran−interface  :class  'nastran−analysis−class
           analysis−model−object ^superior
           body−name (format nil "~{~a~a~}" (list (append (object−name
                ^^selected−body))'gait.bdf))
           nastran−file−name ^body−name
           nastran−version (nth 2 '(:nei−nastran :msc−nastran :nx−nastran))
        )

    )
)
(define−method my−object−selection−method body−part−tagging−class ()
    self  ;; returns the object  itself
)

(define−method my−muscle−point−selection−method muscle−point−properties−class ()
    self  ;; returns the object  itself
)
;; This is the class where the object selection  is needed
(define−class object−selection−class
    :inherit−from(data−model−node−mixin )
    :properties (
        (selected−body :class  'object−selection−property−class
            label "Select Body Object"
            formula nil
            object−selection−method 'my−object−selection−method
        )
        (selected−muscle :class  'object−selection−property−class
            label "Select Muscle Point"
            allowed−classes−list 'muscle−point−properties−class
            formula nil
            object−selection−method 'my−muscle−point−selection−method
        )
    )
)

(define−class body−mesh−analysis
    :inherit−from (object−selection−class)
```

```
   :properties (
      selected-body (default (the model-manager body-mesh-analysis body ground pelvis
           ))
      selected-muscle (default (the model-manager body-mesh-analysis body muscles
           r-hip-abd glut_med2_r origin-point))
      slave-joint-reference-object (the slave-RBE2-joint (:from ^selected-body))
      master-joint-reference-object (the master-RBE2-joint (:from ^selected-body))

      tri-mesh-entities (the surface-elements-query (:from ^tri-mesh))
      node-mesh-entities (the nodes-query (:from ^tri-mesh))
   )
   :subobjects (
      (body :class 'body-class
      )
      (mesh-db :class 'mesh-database-class
      )
      (tri-mesh :class 'paver-mesh-class
         element-shape :tri
         mesh-database-object ^^mesh-db
         object-to-mesh ^^selected-body
         color 'white
      )
      (part-mesh-2d-query :class 'mesh-elements-2d-query-class
         tagged-object-list ^^selected-body
         mesh-object ^^tri-mesh
         color 'magenta
      )

      (analysis :class 'body-analysis-class
         mesh-model-object ^^tri-mesh
         analysis-type 'linear-static

      )
   )
)
```

C.3 Muscle classes

Listing C.3: Muscle classes

```
(define-class muscle-point-properties-class
   :inherit-from (point-object)
   :properties (
      point-coord (default nil)
      body-muscle-ref (default nil)
      orientation (translate ^point-coord)
      reference-coordinate-system ^body-muscle-ref
      color "orange"
      line-width 10
```

```
    )
)
(define-method get-body-muscle-rbe-list muscle-point-properties-class()
)

(define-class muscle-properties-class
    :inherit-from (object)
    :properties(
        origin-point-coordinates (default nil)
        insertion-point-coordinates (default nil)
        origin-point-reference-object (default nil)
        insertion-point-reference-object (default nil)
    )
    :subobjects(

        (origin-point :class 'muscle-point-properties-class
            point-coord ^^origin-point-coordinates
            body-muscle-ref ^^origin-point-reference-object
            global-coord (convert-coords (^^origin-point) '(0 0 0) :from :local :to :global)
                  ;^point-coord if orientation is used
        )

        (insertion-point :class 'muscle-point-properties-class
            point-coord ^^insertion-point-coordinates
            body-muscle-ref ^^insertion-point-reference-object
            global-coord (convert-coords (^^insertion-point) '(0 0 0) :from :local :to
                  :global) ;^point-coord
        )

        (muscle-line-visualization :class 'line-object
            point1 (the global-coord (:from ^^origin-point))
            point2 (the global-coord (:from ^^insertion-point))
            color "red"
            line-width                          15
        )
    )
)

(define-method get-orgin-point muscle-properties-class ()
    !origin-point
)
(define-method get-insertion-point muscle-properties-class ()
    !insertion-point
)

(define-class R-hip-abd-muscle-group-class
    :inherit-from (object)
    :properties(
        pelvis-muscle-ref (default nil)
    femur_r-muscle-ref (default nil)
    tibia_r-muscle-ref (default  nil)
)
:subobjects(

    (glut_max1_r  :class  'muscle-properties-class
        ; originally  4 points  defining  the muscle line
        origin-point-coordinates '(-0.1195 0.0612 0.07)
        insertion-point-coordinates '(-0.0277 -0.0566 0.047)
        origin-point-reference-object ^^pelvis-muscle-ref
        insertion-point-reference-object ^^femur_r-muscle-ref
        color  'yellow
    )
    (glut_med1_r  :class  'muscle-properties-class
        origin-point-coordinates '(-0.0408 0.0304 0.1209)
        insertion-point-coordinates '(-0.0218 -0.0117 0.0555)
        origin-point-reference-object ^^pelvis-muscle-ref
        insertion-point-reference-object ^^femur_r-muscle-ref
    )
    (glut_med2_r  :class  'muscle-properties-class
        origin-point-coordinates '(-0.0855 0.0445 0.0766)
        insertion-point-coordinates '(-0.0258 -0.0058 0.0527)
        origin-point-reference-object ^^pelvis-muscle-ref
        insertion-point-reference-object ^^femur_r-muscle-ref
    )
    (glut_med3_r  :class  'muscle-properties-class
        origin-point-coordinates '( -0.1223 0.0105 0.0648)
        insertion-point-coordinates '(-0.0309 -0.0047 0.0518)
        origin-point-reference-object ^^pelvis-muscle-ref
        insertion-point-reference-object ^^femur_r-muscle-ref
    )
    (glut_min1_r  :class  'muscle-properties-class
        origin-point-coordinates '( -0.0467 -0.008 0.1056)
        insertion-point-coordinates '(-0.0072 -0.0104 0.056)
        origin-point-reference-object ^^pelvis-muscle-ref
        insertion-point-reference-object ^^femur_r-muscle-ref
    )
    (glut_min2_r  :class  'muscle-properties-class
        origin-point-coordinates '(-0.0633 -0.0065 0.0991)
        insertion-point-coordinates '(-0.0096 -0.0104 0.056)
        origin-point-reference-object ^^pelvis-muscle-ref
        insertion-point-reference-object ^^femur_r-muscle-ref
    )
    (glut_min3_r  :class  'muscle-properties-class
        origin-point-coordinates '(-0.0834 -0.0063 0.0856)
        insertion-point-coordinates '(-0.0135 -0.0083 0.055)
        origin-point-reference-object ^^pelvis-muscle-ref
        insertion-point-reference-object ^^femur_r-muscle-ref
    )
```

```
        (peri_r  :class  'muscle−properties−class ;originally 3 points
           origin−point−coordinates '(−0.1396 0.0003 0.0235)
           insertion−point−coordinates '(−0.0148 −0.0036 0.0437)
           origin−point−reference−object ^^pelvis−muscle−ref
           insertion−point−reference−object ^^femur_r−muscle−ref
        )
        (sar_r  :class  'muscle−properties−class ;originally 5 points
           origin−point−coordinates '(−0.0153 −0.0013 0.1242)
           insertion−point−coordinates '(0.0243 −0.084 −0.0252)
           origin−point−reference−object ^^pelvis−muscle−ref
           insertion−point−reference−object ^^tibia_r−muscle−ref
        )
        ( tfl_r  :class  'muscle−properties−class ;originally 4 points
           origin−point−coordinates '(−0.0311 0.0214 0.1241)
           insertion−point−coordinates '(0.006 −0.0487 0.0297)
           origin−point−reference−object ^^pelvis−muscle−ref
           insertion−point−reference−object ^^tibia_r−muscle−ref
        )
    )
)

(define−class muscle−class
    :inherit−from(object)
    :properties (

    )
    :subobjects(
        (R−hip−abd :class 'R−hip−abd−muscle−group−class

        )
    )
)
```

C.4 Body-class

Listing C.4: Classes for structuring the bodies joint and muscle into one human body class

```
(define−class thoracolumbar−sub−geometry−class
    :inherit−from (
        object
    )
    :subobjects(
        (headneck :class  'headneck−sub−geometry−class
        )

        (thoracic  :class  'thoracic−sub−geometry−class
        )

        (lumbar :class  'lumbar-sub-geometry-class
        )
        (ribs   :class  'rib-gait-sub-geometry-class
        )
    )
)
(define-class osim-model-data-model-class
    :inherit-from ( data-model-node-mixin )
    :properties  (
        (osim-model-type :class 'option-property-class
            label "Model type"
            mode 'menu
            ;formula :inherit-formula
            options-list  '(gait2392 thoracolumbar)
            labels-list  '("Gait2392" "Thoracolumbar")
        )
    )
)
(define-class body-class
    :inherit-from (
        object
        ;osim-model-data-model-class
    )
    :properties  (
        ;options are gait2392 thoracolumbar
        osim-model-type (default 'gait2392)
        ;osim-model-type (default 'thoracolumbar)

        talus_r-reference-joint  (the  talus_r-joint  (:from ^joints))
        calcn_r-reference-joint  (the  calcn_r-joint  (:from ^joints))
        toes_r-reference-joint  (the  toes_r-joint  (:from ^joints))
        talus_l-reference-joint  (the  talus_l-joint  (:from ^joints))
        calcn_l-reference-joint  (the  calcn_l-joint  (:from ^joints))
        toes_l-reference-joint  (the  toes_l-joint  (:from ^joints))
        torso-reference-joint (the torso-joint (:from ^joints))
        femur_r-reference-joint (the femur_r-joint (:from ^joints))
        tibia_r-reference-joint  (the  tibia_r-joint  (:from ^joints))
        femur_l-reference-joint (the femur_l-joint (:from ^joints))
        tibia_l-reference-joint  (the  tibia_l-joint  (:from ^joints))
        pelvis_gait-reference-joint  (the  pelvis_gait-joint  (:from ^joints))

        knee-RBE2-reference-joint (the knee-RBE2-joint (:from ^joints))
        l-knee-RBE2-reference-joint (the l-knee-RBE2-joint (:from ^joints))

        thoracic12-reference-joint (the thoracic12-joint (:from ^joints))
        thoracic11-reference-joint (the thoracic11-joint (:from ^joints))
        thoracic10-reference-joint (the thoracic10-joint (:from ^joints))
        thoracic9-reference-joint (the thoracic9-joint (:from ^joints))
        thoracic8-reference-joint (the thoracic8-joint (:from ^joints))
    thoracic7−reference−joint (the thoracic7−joint (:from ^joints))
    thoracic6−reference−joint (the thoracic6−joint (:from ^joints))
    thoracic5−reference−joint (the thoracic5−joint (:from ^joints))
    thoracic4−reference−joint (the thoracic4−joint (:from ^joints))
    thoracic3−reference−joint (the thoracic3−joint (:from ^joints))
    thoracic2−reference−joint (the thoracic2−joint (:from ^joints))
    thoracic1−reference−joint (the thoracic1−joint (:from ^joints))

    head_neck−reference−joint (the head_neck−joint (:from ^joints))
    hand_R−reference−joint (the hand_R−joint (:from ^joints))
    hand_L−reference−joint (the hand_L−joint (:from ^joints))

    lumbar5−reference−joint (the lumbar5−joint (:from ^joints))
    lumbar4−reference−joint (the lumbar4−joint (:from ^joints))
    lumbar3−reference−joint (the lumbar3−joint (:from ^joints))
    lumbar2−reference−joint (the lumbar2−joint (:from ^joints))
    lumbar1−reference−joint (the lumbar1−joint (:from ^joints))

    clavicle_R−reference−joint (the clavicle_R−joint( :from ^joints))
    scapula_R−reference−joint (the scapula_R−joint (:from ^joints))
    humerus_R−reference−joint (the humerus_R−joint(:from ^joints))
    ulna_R−reference−joint (the ulna_R−joint (:from ^joints))
    radius_R−reference−joint (the radius_R−joint (:from ^joints))
    clavicle_L−reference−joint (the  clavicle_L−joint (:from ^joints))
    scapula_L−reference−joint (the scapula_L−joint (:from ^joints))
    humerus_L−reference−joint (the humerus_L−joint(:from ^joints))
    ulna_L−reference−joint (the  ulna_L−joint (:from ^joints))
    radius_L−reference−joint (the radius_L−joint(:from ^joints))

    ground−reference−joint (the ground−joint (:from ^joints))
    sacrum−reference−joint (the sacrum−joint (:from ^joints))
    pelvis−reference−joint (the pelvis−joint (:from ^joints))
)
:subobjects (
    (joints  :class  '(gait−joint−class joint−class)
    )

    (muscles :class  'muscle−class
        pelvis−muscle−ref (the pelvis_gait−joint (:from ^^joints))
        femur_r−muscle−ref (the femur_r−joint (:from ^^joints))
        tibia_r−muscle−ref (the tibia_r−joint (:from ^^joints))
    )

    (feet  :class  'feet−sub−geometry−class
    )


    (legs  :class  'legs−sub−geometry−class
    )
```

```
        (arm :class  'arm-sub-geometry-class
        )

        (hand :class  'hand-sub-geometry-class
        )

        (ground :class   (case !osim-model-type
                (gait2392   'gait-ground-sub-geometry-class)
                (thoracolumbar 'ground-sub-geometry-class)
            )
        )

        (upper_body :class (case !osim-model-type
                (gait2392 'upper_body-sub-geometry-class)
                (thoracolumbar 'thoracolumbar-sub-geometry-class)
            )

        )
    )
)
```

C.5 Femur scaling example

Listing C.5: Generic scaling class with femur example

```
(in-package :aml)

(define-class femur-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\femur_points.dat"
        con-file   "C:\\PytonWorkspace\\femur_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class body-part-class
    :inherit-from (
        femur-web-surface
        object
    )
    :properties (
    )
)

(define-class scale-property-class
   :inherit−from (scale−object)
   :properties (
      source−object ^scale−object
      scale−factor (default 2.0)
      color (default "yellow")
   )

   :subobjects (
      (scale−object :class 'body−part−class

      )
   )
)

(define−class scale−example−class
   :inherit−from (object)
   :properties (

   )

   :subobjects (
      (scale−referance−cordinate−system :class 'coordinate−system−class
         origin ( list  0 0 0 )
         display? nil
      )
      (femur−original :class 'body−part−class
         color "yellow"
         orientation ( list (translate ( list 0.5 0 0)))
         reference−coordinate−system ^^scale−referance−cordinate−system

      )
      (femur−scale−factor2 :class 'scale−property−class
         color "orange"
         orientation ( list (translate ( list 1 0 0)))
         reference−coordinate−system ^^scale−referance−cordinate−system
      )
      (femur−scale−factor4:class 'scale−property−class
         color "red"
         scale−factor 4
         orientation ( list (translate  ( list 1.5 0 0)))
         reference−coordinate−system ^^scale−referance−cordinate−system
      )
   )
)
```

C.6 Thoracolumbar joint classes

Listing C.6: Joint-classes from thoracolumbar model

```
(define−class joint−coordinate−system−class
   :inherit−from (
      coordinate−system−class
   )
   :properties (
       display? nil
      length 0.1
      coordinate−system−length 0.1
      body−translation−list (default '(0 0 0))
      body−rotation−list (default '(0 0 0))
      orientation (default
         ( list
            (rotate (radians−to−degrees (first ^body−rotation−list)) :x−axis)
            (rotate (radians−to−degrees (second ^body−rotation−list) ) :y−axis)
            (rotate (radians−to−degrees (third ^body−rotation−list)) :z−axis)
            (translate ^body−translation−list)
         )
      )

   )
)


(define−class joint−class
   :inherit−from (
      object
   )
   :properties (
   )
   :subobjects (
      (ground−joint :class 'joint−coordinate−system−class
         origin ( list  0 0 0 )
      )

      (sacrum−joint :class 'joint−coordinate−system−class
         reference−coordinate−system ^^ground−joint
      )

      (pelvis−joint  :class 'joint−coordinate−system−class
         body−translation−list '(0 0.049 0 )
         reference−coordinate−system ^^sacrum−joint
      )

      (Abdomen−joint :class 'joint−coordinate−system−class
         body−translation−list '(−0.078 0.071 −0 )
         reference−coordinate−system ^^sacrum−joint
      )

      (lumbar5−joint :class 'joint−coordinate−system−class
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.36652 )
            :z−axis)(translate (list −0.098 0.071 0 )))
      reference−coordinate−system ^^sacrum−joint
  )

  (lumbar4−joint :class 'joint−coordinate−system−class
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0.31416 )
            :z−axis)(translate ( list 0 0.039581 0 )))
      reference−coordinate−system ^^lumbar5−joint
  )

  (lumbar3−joint :class 'joint−coordinate−system−class
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0.20944 )
            :z−axis)(translate ( list 0 0.036802 0 )))
      reference−coordinate−system ^^lumbar4−joint
  )

  (lumbar2−joint :class 'joint−coordinate−system−class
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0.12217 )
            :z−axis)(translate ( list 0 0.036764 0 )))
      reference−coordinate−system ^^lumbar3−joint
  )

  (lumbar1−joint :class 'joint−coordinate−system−class
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0.10472 )
            :z−axis)(translate ( list 0 0.034054 0 )))
      reference−coordinate−system ^^lumbar2−joint
  )

  (thoracic12−joint :class 'joint−coordinate−system−class
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.034907 )
            :z−axis)(translate (list 0 0.035497 0 )))
      reference−coordinate−system ^^lumbar1−joint
  )

  (thoracic11−joint :class 'joint−coordinate−system−class
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.069813 )
            :z−axis)(translate (list 0 0.033837 0 )))
      reference−coordinate−system ^^thoracic12−joint
  )

  (thoracic10−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.087266 )
        :z−axis)(translate (list 0 0.029544 0 )))
    reference−coordinate−system ^^thoracic11−joint
)

(thoracic9−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.069813 )
        :z−axis)(translate (list 0 0.029 0 )))
    reference−coordinate−system ^^thoracic10−joint
)

(thoracic8−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.087266 )
        :z−axis)(translate (list 0 0.025135 0 )))
    reference−coordinate−system ^^thoracic9−joint
)

(thoracic7−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.10472 )
        :z−axis)(translate (list 0 0.024615 0 )))
    reference−coordinate−system ^^thoracic8−joint
)

(thoracic6−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.10472 )
        :z−axis)(translate (list 0 0.025934 0 )))
    reference−coordinate−system ^^thoracic7−joint
)

(thoracic5−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.10472 )
        :z−axis)(translate (list 0 0.025672 0 )))
    reference−coordinate−system ^^thoracic6−joint
)

(thoracic4−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.10472 )
        :z−axis)(translate (list 0 0.026267 0 )))
    reference−coordinate−system ^^thoracic5−joint
)

(thoracic3−joint :class 'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.069813 )
         :z−axis)(translate (list  0  0.024475 0 )))
    reference−coordinate−system ^^thoracic4−joint
)

(thoracic2−joint   :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.05236 )
         :z−axis)(translate (list  0  0.02221  0  )))
    reference−coordinate−system ^^thoracic3−joint
)

(thoracic1−joint   :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.017453 )
         :z−axis)(translate (list  0  0.020271  0 )))
    reference−coordinate−system ^^thoracic2−joint
)

(head_neck−joint  :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0.5236 )
         :z−axis)(translate ( list  −0.00684 0.020271 0 )))
    reference−coordinate−system ^^thoracic1−joint
)

(rib12_R−joint  :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 1.5708 ) :x−axis)(rotate
         (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.2748 )
         :z−axis)(translate ( list  −0.022046 0.022058 0.021423 )))
    reference−coordinate−system ^^thoracic12−joint
)

(rib11_R−joint  :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 1.5708 ) :x−axis)(rotate
         (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.34175 )
         :z−axis)(translate ( list  −0.01598 0.0188 0.01679 )))
    reference−coordinate−system ^^thoracic11−joint
)

(rib10_R−joint  :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 1.5708 ) :x−axis)(rotate
         (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.40034 )
         :z−axis)(translate ( list  −0.014883 0.029 0.01702 )))
    reference−coordinate−system ^^thoracic10−joint
)

(rib9_R−joint  :class  'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.45172 )
          :z−axis)(translate (list −0.011703 0.025135 0.014682 )))
    reference−coordinate−system ^^thoracic9−joint
)

(rib8_R−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.49691 )
          :z−axis)(translate (list −0.010496 0.024615 0.014582 )))
    reference−coordinate−system ^^thoracic8−joint
)

(rib7_R−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.53682 )
          :z−axis)(translate (list −0.010069 0.025934 0.015672 )))
    reference−coordinate−system ^^thoracic7−joint
)

(rib6_R−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.57221 )
          :z−axis)(translate (list −0.0088519 0.025672 0.015663 )))
    reference−coordinate−system ^^thoracic6−joint
)

(rib5_R−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.60373 )
          :z−axis)(translate (list −0.0078849 0.026267 0.016164 )))
    reference−coordinate−system ^^thoracic5−joint
)

(rib4_R−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.63193 )
          :z−axis)(translate (list −0.0061241 0.024475 0.014919 )))
    reference−coordinate−system ^^thoracic4−joint
)

(rib3_R−joint :class 'joint−coordinate−system−class
    orientation ( list (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.65727 )
          :z−axis)(translate (list −0.0044462 0.02221 0.013345 )))
    reference−coordinate−system ^^thoracic3−joint
)

(rib2_R−joint :class 'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.68013 )
          :z−axis)(translate ( list  −0.0030697 0.020271 0.011998 )))
    reference−coordinate−system ^^thoracic2−joint
)

(rib1_R−joint   :class   'joint−coordinate−system−class
    ;;  original  orientation
    ;orientation  ( list  (rotate  (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees 0.70085 )
          :z−axis)(translate ( list  −0.0033483 0.019498 0.018755 )))
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0) :y−axis)(rotate (radians−to−degrees 0.5 )
          :z−axis)( translate  ( list  −0.0033483 0.019498 0.018755 )))
    reference−coordinate−system ^^thoracic1−joint
)

(rib12_L−joint   :class   'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.2748 )
          :z−axis)(translate (list  −0.022046 0.022058 −0.021423 )))
    reference−coordinate−system ^^thoracic12−joint
)

(rib11_L−joint   :class   'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.34175 )
          :z−axis)(translate (list  −0.01598 0.0188 −0.01679 )))
    reference−coordinate−system ^^thoracic11−joint
)

(rib10_L−joint   :class   'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.40034 )
          :z−axis)(translate (list  −0.014883 0.029 −0.01702 )))
    reference−coordinate−system ^^thoracic10−joint
)

(rib9_L−joint   :class   'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.45172 )
          :z−axis)(translate (list  −0.011703 0.025135 −0.014682 )))
    reference−coordinate−system ^^thoracic9−joint
)

(rib8_L−joint   :class   'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 1.5708 ) :x−axis)(rotate
          (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.49691 )
          :z−axis)(translate (list  −0.010496 0.024615 −0.014582 )))
    reference−coordinate−system ^^thoracic8−joint
)

(rib7_L−joint  :class  'joint−coordinate−system−class
    orientation ( list  (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
         (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.53682 )
         :z−axis)(translate (list −0.010069 0.025934 −0.015672 )))
    reference−coordinate−system ^^thoracic7−joint
)

(rib6_L−joint  :class  'joint−coordinate−system−class
    orientation ( list  (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
         (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.57221 )
         :z−axis)(translate (list −0.0088519 0.025672 −0.015663 )))
    reference−coordinate−system ^^thoracic6−joint
)

(rib5_L−joint  :class  'joint−coordinate−system−class
    orientation ( list  (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
         (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.60373 )
         :z−axis)(translate (list −0.0078849 0.026267 −0.016164 )))
    reference−coordinate−system ^^thoracic5−joint
)

(rib4_L−joint  :class  'joint−coordinate−system−class
    orientation ( list  (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
         (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.63193 )
         :z−axis)(translate (list −0.0061241 0.024475 −0.014919 )))
    reference−coordinate−system ^^thoracic4−joint
)

(rib3_L−joint  :class  'joint−coordinate−system−class
    orientation ( list  (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
         (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.65727 )
         :z−axis)(translate (list −0.0044462 0.02221 −0.013345 )))
    reference−coordinate−system ^^thoracic3−joint
)

(rib2_L−joint  :class  'joint−coordinate−system−class
    orientation ( list  (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
         (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.68013 )
         :z−axis)(translate (list −0.0030697 0.020271 −0.011998 )))
    reference−coordinate−system ^^thoracic2−joint
)

(rib1_L−joint  :class  'joint−coordinate−system−class
    orientation ( list  (rotate (radians−to−degrees 1.5708 ) :x−axis)(rotate
         (radians−to−degrees 3.1416 ) :y−axis)(rotate (radians−to−degrees −0.70085 )
         :z−axis)(translate (list −0.0033483 0.019498 −0.018755 )))
    reference−coordinate−system ^^thoracic1−joint
)

(sternum−joint  :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  0.0684394468871297 −0.0547530538394528
          0.0147941171846263 )))
    reference−coordinate−system ^^rib1_R−joint
)

(clavicle_R−joint   :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees −0.2618 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate (list  0 0 0 )))
    reference−coordinate−system ^^sternum−joint
)

(scapula_R−joint  :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0.34907 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  −0.01433 0.02007 0.13554 )))
    reference−coordinate−system ^^clavicle_R−joint
)

(humerus_R−joint :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees −0.087266 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate (list −0.009595 −0.034 0.009 )))
    reference−coordinate−system ^^scapula_R−joint
)

(ulna_R−joint  :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  0.0061  −0.2904 −0.0123 )))
    reference−coordinate−system ^^humerus_R−joint
)

(radius_R−joint  :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  0.0004  −0.011503 0.019999 )))
    reference−coordinate−system ^^ulna_R−joint
)

(hand_R−joint :class  'joint−coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
```

```
            :z−axis)(translate ( list  0.018  −0.242 0.025 )))
        reference−coordinate−system ^^radius_R−joint
    )

    ( clavicle_L−joint    :class  'joint−coordinate−system−class
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0.2618 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0 0  −0.0670985961566419 )))
        reference−coordinate−system ^^sternum−joint
    )

    (scapula_L−joint   :class  'joint−coordinate−system−class
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees  −0.34907 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate (list  −0.01433 0.02007 −0.13554 )))
        reference−coordinate−system ^^clavicle_L−joint
    )

    (humerus_L−joint  :class  'joint−coordinate−system−class
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0.087266 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  −0.009595 −0.034 −0.009 )))
        reference−coordinate−system ^^scapula_L−joint
    )

    (ulna_L−joint   :class  'joint−coordinate−system−class
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0.0061  −0.2904 0.0123 )))
        reference−coordinate−system ^^humerus_L−joint
    )

    (radius_L−joint   :class  'joint−coordinate−system−class
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0.0004  −0.011503  −0.019999 )))
        reference−coordinate−system ^^ulna_L−joint
    )

    (hand_L−joint  :class  'joint−coordinate−system−class
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0.018  −0.242  −0.025 )))
        reference−coordinate−system ^^radius_L−joint
    )

    (Abd_L_L1−joint  :class  'joint−coordinate−system−class
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.38397 )
            :z−axis)(translate (list 0 0 0 )))
        reference−coordinate−system ^^lumbar1−joint
    )

    (Abd_L_L2−joint :class 'joint−coordinate−system−class
        orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.27925 )
            :z−axis)(translate (list 0 0 0 )))
        reference−coordinate−system ^^lumbar2−joint
    )

    (Abd_L_L3−joint :class 'joint−coordinate−system−class
        orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.15708 )
            :z−axis)(translate (list 0 0 0 )))
        reference−coordinate−system ^^lumbar3−joint
    )

    (Abd_L_L4−joint :class 'joint−coordinate−system−class
        orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0.05236 )
            :z−axis)(translate ( list 0 0 0 )))
        reference−coordinate−system ^^lumbar4−joint
    )

    (Abd_L_L5−joint :class 'joint−coordinate−system−class
        orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0.36652 )
            :z−axis)(translate ( list 0 0 0 )))
        reference−coordinate−system ^^lumbar5−joint
    )

    (Abd_R_L1−joint :class 'joint−coordinate−system−class
        orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.38397 )
            :z−axis)(translate (list 0 0 0 )))
        reference−coordinate−system ^^lumbar1−joint
    )

    (Abd_R_L2−joint :class 'joint−coordinate−system−class
        orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.27925 )
            :z−axis)(translate (list 0 0 0 )))
        reference−coordinate−system ^^lumbar2−joint
    )

    (Abd_R_L3−joint :class 'joint−coordinate−system−class
        orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0.15708 )
```

```
              :z−axis)(translate (list 0 0 0 )))
          reference−coordinate−system ^^lumbar3−joint
        )

        (Abd_R_L4−joint :class 'joint−coordinate−system−class
          orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0.05236 )
              :z−axis)(translate ( list  0 0 0 )))
          reference−coordinate−system ^^lumbar4−joint
        )

        (Abd_R_L5−joint :class 'joint−coordinate−system−class
          orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0.36652 )
              :z−axis)(translate ( list  0 0 0 )))
          reference−coordinate−system ^^lumbar5−joint
        )
    )
)
```

C.7 Joint classes from gait model

Listing C.7: Joint-class from gait model

```
;Approximation to the SimSpline function from OpenSim
(defun get−knee−joint−translation(knee−angle)

    (case knee−angle
        (10 ( list (rotate knee−angle :z−axis)(translate ( list  −0.00515263 −0.39523564 0 ))))
        (0  ( list (rotate knee−angle :z−axis)(translate ( list  −0.00436810 −0.39571497 0 ))))
        (−10 (list (rotate knee−angle :z−axis)(translate ( list −0.00308293 −0.39663410 0 ))))
        (−20 (list (rotate knee−angle :z−axis)(translate ( list  −0.00103582 −0.39763316 0 ))))
        (−30 (list (rotate knee−angle :z−axis)(translate ( list  0.00073763  −0.39898467 0 ))))
        (−40 (list (rotate knee−angle :z−axis)(translate ( list   0.00212033  −0.40106284 0 ))))
        (−50 (list (rotate knee−angle :z−axis)(translate ( list  0.00333546  −0.40335643 0 ))))
        (−60 (list (rotate knee−angle :z−axis)(translate ( list  0.00411215  −0.40554767 0 ))))
        (−70 (list (rotate knee−angle :z−axis)(translate ( list  0.00436737  −0.40820478 0 ))))
        (−80 (list (rotate knee−angle :z−axis)(translate ( list  0.00408543  −0.41099947 0 ))))
        (−90 (list (rotate knee−angle :z−axis)(translate ( list  0.00408543  −0.41099947 0 ))))
        (−100 (list (rotate knee−angle :z−axis)(translate ( list  0.00183353  −0.41655753 0 ))))
        (−110 (list (rotate knee−angle :z−axis)(translate ( list   −0.00045820 −0.41956626 0
              ))))
        (−120 (list (rotate knee−angle :z−axis)(translate ( list   −0.00320000 −0.42260000 0
              ))))
    )
)

(define−class gait−joint−class
:inherit-from (
    object
)
:properties (
    display-coord-systems? (default nil)
    coordinate-system-length 0.1

    hip-aduction-r (default 0)
    hip-rotation-r (default 0)
    hip-flexion-r (default 30)
    knee-angle-r (default -80)
    knee-angle-l (default 0)
    ankle-angle-r (default 0)
    subtalar-angle-r (default 0)
    mtp-angle-r (default 0)
)
:subobjects (
    (ground_gait-joint  :class  'coordinate-system-class
        origin ( list   0 0 0 )
        display? ^^display-coord-systems?
        length ^^coordinate-system-length
    )

    ( pelvis_gait-joint   :class  'coordinate-system-class
        orientation ( list  (rotate (radians-to-degrees 0 ) :x-axis)(rotate
              (radians-to-degrees 0 ) :y-axis)(rotate (radians-to-degrees 0 )
              :z-axis)(translate ( list   0 0 0 )))
        reference-coordinate-system ^^ground_gait-joint
        display? ^^display-coord-systems?
        length ^^coordinate-system-length
    )

    ; spline investigation-joint
    (knee-RBE2-joint :class 'coordinate-system-class
        orientation ( list  (rotate (radians-to-degrees ^^knee-angle-r) :z-axis)(translate
              (list -0.0105 -0.4091 0)))
        reference-coordinate-system ^^femur_r-joint
        display? ^^display-coord-systems?
        length ^^coordinate-system-length
    )
    ; spline investigation-joint
    (l-knee-RBE2-joint :class 'coordinate-system-class
        orientation ( list  (translate ( list  -0.0105 -0.4091 0)))
        reference-coordinate-system ^^femur_l-joint
        display? ^^display-coord-systems?
        length ^^coordinate-system-length
    )

    (femur_r-joint :class  'coordinate-system-class
    orientation  ( list
        (rotate ^^hip−aduction−r :x−axis)(rotate ^^hip−rotation−r :y−axis) (rotate
              ^^hip−flexion−r :z−axis )(translate (list  −0.0707 −0.0661 0.0835 ))
        )
    reference−coordinate−system ^^pelvis_gait−joint
    display? ^^display−coord−systems?
    )

; Knee joint
( tibia_r−joint   :class  'coordinate−system−class
    orientation  (get−knee−joint−translation ^^knee−angle−r)
    ; orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  get−knee−joint−translation ^^knee−angle−r)))
    reference−coordinate−system ^^femur_r−joint
    display? ^^display−coord−systems?
    length ^^coordinate−system−length

)

; Ankle joint  orientation
( talus_r−joint   :class  'coordinate−system−class
    orientation  ( list
        ( list  (rotate ^^ankle−angle−r '(−0.10 −0.17 0.9799))(translate (list  0 −0.43 0
              )))
        )
    reference−coordinate−system ^^tibia_r−joint
    display? ^^display−coord−systems?
    length ^^coordinate−system−length
)

; Subtalar joint  orientation
(calcn_r−joint   :class  'coordinate−system−class
    orientation  ( list
        ( list  (rotate ^^subtalar−angle−r '(0.78 0.6 −0.18))(translate ( list   −0.04877
              −0.04195 0.00792 )))
        )
    reference−coordinate−system ^^talus_r−joint
    display? ^^display−coord−systems?
    length ^^coordinate−system−length
)

;; Toe joint  orientation
(toes_r−joint   :class  'coordinate−system−class
    orientation  ( list
        ( list  (rotate ^^mtp−angle−r '(−0.58 0 0.8))(translate (list  0.1788 −0.002
              0.00108 )))
        )
    reference−coordinate−system ^^calcn_r−joint
    display? ^^display−coord−systems?
    length ^^coordinate−system−length
)

(femur_l−joint   :class  'coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list    −0.0707 −0.0661 −0.0835 )))
    reference−coordinate−system ^^pelvis_gait−joint
    display? ^^display−coord−systems?
    length ^^coordinate−system−length
)

( tibia_l−joint    :class  'coordinate−system−class
    ; orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list    0  0  0  )))
    orientation  (get−knee−joint−translation ^^knee−angle−l)
    reference−coordinate−system ^^femur_l−joint
    display? ^^display−coord−systems?
    length ^^coordinate−system−length
)

( talus_l−joint    :class  'coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list    0  −0.43 0 )))
    reference−coordinate−system ^^tibia_l−joint
    display? ^^display−coord−systems?
    length ^^coordinate−system−length
)

( calcn_l−joint    :class  'coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list    −0.04877 −0.04195 −0.00792 )))
    reference−coordinate−system ^^talus_l−joint
    display? ^^display−coord−systems?
    length ^^coordinate−system−length
)

( toes_l−joint    :class   'coordinate−system−class
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list    0.1788 −0.002 −0.00108 )))
    reference−coordinate−system ^^calcn_l−joint
    display? ^^display−coord−systems?
    length ^^coordinate−system−length
)
```

```
        (torso−joint   :class  'coordinate−system−class
            orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                  :z−axis)(translate ( list   −0.1007 0.0815 0 )))
            reference−coordinate−system ^^pelvis_gait−joint
            display? ^^display−coord−systems?
            length ^^coordinate−system−length
        )

    )
)
```

C.8 Gait2392 body classes

Listing C.8: Body classes from gait model

```
(define−class body−part−tagging−class
    :inherit−from (
        tagging−object
    )
    :properties  (
        max−elem−size (default nil)
        min−elem−size (default nil)
        tag−dimensions '(0 1 2 3)
        tag−attributes ( list  ^^max−elem−size ^^min−elem−size 1 0.1 0 20.0 1.0e−5)
        reference−coordinate−system (default nil)
        slave−RBE2−joint ^reference−coordinate−system
        master−RBE2−joint (default nil)
    )
)

(define−class feet−sub−geometry−class
    :inherit−from (
        object
    )
    :properties  (
        max−elem−size 0.012
        min−elem−size 0.003

        talus_r−reference−joint  (default  nil )
        calcn_r−reference−joint  (default  nil )
        toes_r−reference−joint  (default  nil )
        talus_l−reference−joint  (default  nil )
        calcn_l−reference−joint  (default  nil )
        toes_l−reference−joint  (default  nil )
    )
    :subobjects  (
        (talus  :class  '(body−part−tagging−class talus−web−surface)
          orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
               (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
               :z−axis)(translate (list 0 0 0)))
          reference−coordinate−system ^^talus_r−reference−joint
          master−RBE2−joint ^^calcn_r−reference−joint
       )
       ;calcanus
       (foot  :class  '(body−part−tagging−class foot−web−surface)
          orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
               (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
               :z−axis)(translate (list 0 0 0)))
          reference−coordinate−system ^^calcn_r−reference−joint
          master−RBE2−joint ^^toes_r−reference−joint
       )
       ;toe
       (bofoot  :class  '(body−part−tagging−class bofoot−web−surface)
          orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
               (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
               :z−axis)(translate (list 0 0 0)))
          reference−coordinate−system ^^toes_r−reference−joint
       )
       ( l_talus   :class  '(body−part−tagging−class l_talus−web−surface)
          orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
               (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
               :z−axis)(translate (list 0 0 0)))
          reference−coordinate−system ^^talus_l−reference−joint
          master−RBE2−joint ^^calcn_l−reference−joint
       )
       ( l_foot   :class  '(body−part−tagging−class l_foot−web−surface)
          orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
               (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
               :z−axis)(translate (list 0 0 0)))
          reference−coordinate−system ^^calcn_l−reference−joint
          master−RBE2−joint ^^toes_l−reference−joint
       )
       (l_bofoot   :class  '(body−part−tagging−class l_bofoot−web−surface)
          orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
               (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
               :z−axis)(translate (list 0 0 0)))
          reference−coordinate−system ^^toes_l−reference−joint
       )
    )
    )

; rib sub−geometry−class for full body−model
(define−class rib−gait−sub−geometry−class
   :inherit−from (
       object
   )
```

```
    :properties (
    max−elem−size 0.012
        min−elem−size 0.003

        torso−reference−joint (default nil)
    )
    :subobjects (
        (hat_ribs :class '(body−part−tagging−class hat_ribs−web−surface)
            orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                  :z−axis)(translate (list 0 0 0)))
            reference−coordinate−system ^^torso−reference−joint
        )
    )
)

(define−class upper_body−sub−geometry−class
    :inherit−from (
        object
    )
    :properties (
        max−elem−size 0.012
        min−elem−size 0.003

        torso−reference−joint (default nil)
    )
    :subobjects (
        (hat_spine :class '(body−part−tagging−class hat_spine−web−surface)
            orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                  :z−axis)(translate (list 0 0 0)))
            reference−coordinate−system ^^torso−reference−joint
        )
        (hat_jaw :class '(body−part−tagging−class hat_jaw−web−surface)
            orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                  :z−axis)(translate (list 0 0 0)))
            reference−coordinate−system ^^torso−reference−joint
        )
        (hat_skull :class '(body−part−tagging−class hat_skull−web−surface)
            orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                  :z−axis)(translate (list 0 0 0)))
            reference−coordinate−system ^^torso−reference−joint
        )
        (hat_ribs :class '(body−part−tagging−class hat_ribs−web−surface)
            orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                  :z−axis)(translate (list 0 0 0)))
            reference−coordinate−system ^^torso−reference−joint
        )
    )
)

(define−class legs−sub−geometry−class
    :inherit−from (
        object
    )
    :properties (
        max−elem−size 0.012
        min−elem−size 0.003

        l−knee−RBE2−reference−joint (default nil)
        knee−RBE2−reference−joint (default nil)
        femur_r−reference−joint (default nil)
        tibia_r−reference−joint (default nil)
        femur_l−reference−joint (default nil)
        tibia_l−reference−joint (default nil)
        talus_r−reference−joint (default nil)
    )
    :subobjects (
        (femur :class '(body−part−tagging−class femur−web−surface)
            orientation (list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                  :z−axis)(translate (list 0 0 0)))
            reference−coordinate−system ^^femur_r−reference−joint
            master−RBE2−joint ^^knee−RBE2−reference−joint
        )
        (tibia :class '(body−part−tagging−class tibia−web−surface)
            orientation (list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                  :z−axis)(translate (list 0 0 0)))
            ;reference−coordinate−system ^^knee−RBE2−joint
            reference−coordinate−system ^^tibia_r−reference−joint
            slave−RBE2−joint ^^knee−RBE2−reference−joint
            master−RBE2−joint ^^talus_r−reference−joint
        )
        (fibula :class '(body−part−tagging−class fibula−web−surface)
            orientation (list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                  :z−axis)(translate (list 0 0 0)))
            reference−coordinate−system ^^tibia_r−reference−joint

        )
        (l_femur :class '( body−part−tagging−class l_femur−web−surface)
            orientation (list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                  :z−axis)(translate (list 0 0 0)))
```

```
          reference−coordinate−system ^^femur_l−reference−joint
          master−RBE2−joint ^^l−knee−RBE2−reference−joint
       )
       ( l_tibia   :class  '(  body−part−tagging−class l_tibia−web−surface)
          orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
               (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
               :z−axis)(translate (list 0 0 0)))
          reference−coordinate−system ^^tibia_l−reference−joint
          slave−RBE2−joint ^^l−knee−RBE2−reference−joint
          master−RBE2−joint ^^talus_l−reference−joint
       )
       ( l_fibula   :class  '( body−part−tagging−class l_fibula−web−surface)
          orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
               (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
               :z−axis)(translate (list 0 0 0)))
          reference−coordinate−system ^^tibia_l−reference−joint
          slave−RBE2−joint ^^knee−RBE2−reference−joint
          master−RBE2−joint ^^talus_l−reference−joint
       )
   )
)
(define−class gait−ground−sub−geometry−class
   :inherit−from (
       object
   )

   :properties  (
       max−elem−size 0.012
       min−elem−size 0.003

       pelvis_gait−reference−joint  (default  nil)
       femur_r−reference−joint (default  nil)

       torso−reference−joint (default  nil)
       femur_l−reference−joint (default  nil)
   )
   :subobjects (
       (sacrum :class  '(body−part−tagging−class sacrum−web−surface)
          orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
               (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
               :z−axis)(translate (list 0 0 0)))
          reference−coordinate−system ^^pelvis_gait−reference−joint
          master−RBE2−joint ^^torso−reference−joint
       )
       ( pelvis  :class  '(body−part−tagging−class pelvis−web−surface)
          orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
               (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
               :z−axis)(translate (list 0 0 0)))
          reference−coordinate−system ^^pelvis_gait−reference−joint
```

```
            master−RBE2−joint ^^femur_r−reference−joint
        )
        ( l_pelvis   :class   '(body−part−tagging−class l_pelvis−web−surface)
            orientation  ( list   (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                  :z−axis)(translate (list 0 0 0)))
            reference−coordinate−system ^^pelvis_gait−reference−joint
            master−RBE2−joint ^^femur_l−reference−joint
        )
    )
)
```

C.9 Thoracolumbar body classes

Listing C.9: Body classes from thoracolumbar model

```

(define−class headneck−sub−geometry−class
    :inherit−from (
        object
    )
    :properties  (
        head_neck−reference−joint (default nil)
        color  (default  "white")
    )
    :subobjects (
        (rotatedcerv7  :class   '  rotatedcerv7−web−surface
            orientation  ( list   (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                  :z−axis)(translate ( list   0  0  0)))
            reference−coordinate−system ^^head_neck−reference−joint
        )
        (cerv6  :class   '  cerv6−web−surface
            orientation  ( list   (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                  :z−axis)(translate ( list   0.0038 0.0164 0)))
            reference−coordinate−system ^^head_neck−reference−joint
        )
        (cerv5  :class   '  cerv5−web−surface
            orientation  ( list   (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                  :z−axis)(translate ( list   0.008  0.0343 0)))
            reference−coordinate−system ^^head_neck−reference−joint
        )
        (cerv4  :class   '  cerv4−web−surface
            orientation  ( list   (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                  :z−axis)(translate ( list   0.0114 0.0535 0)))
            reference−coordinate−system ^^head_neck−reference−joint
        )
        (cerv3 :class ' cerv3−web−surface
            orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                :z−axis)(translate ( list 0.014 0.0689 0)))
            reference−coordinate−system ^^head_neck−reference−joint
        )
        (cerv2 :class ' cerv2−web−surface
            orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                :z−axis)(translate ( list 0.0139 0.0852 0)))
            reference−coordinate−system ^^head_neck−reference−joint
        )
        (cerv1 :class ' cerv1−web−surface
            orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                :z−axis)(translate ( list −0.0217 0.103 0)))
            reference−coordinate−system ^^head_neck−reference−joint
        )
        (jaw :class ' jaw−web−surface
            orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                :z−axis)(translate ( list 0.0216 0.1179 0)))
            reference−coordinate−system ^^head_neck−reference−joint
        )
        ( skull :class ' skull−web−surface
            orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                :z−axis)(translate ( list 0.0216 0.1179 0)))
            reference−coordinate−system ^^head_neck−reference−joint
        )
    )
)
(define−class thoracic−sub−geometry−class
    :inherit−from (
        object
    )
    :properties (
        color (default "white")

        thoracic12−reference−joint (default nil)
        thoracic11−reference−joint (default nil)
        thoracic10−reference−joint (default nil)
        thoracic9−reference−joint (default nil)
        thoracic8−reference−joint (default nil)
        thoracic7−reference−joint (default nil)
        thoracic6−reference−joint (default nil)
        thoracic5−reference−joint (default nil)
        thoracic4−reference−joint (default nil)
    thoracic3−reference−joint (default  nil)
    thoracic2−reference−joint (default  nil)
    thoracic1−reference−joint (default  nil)
  )
  :subobjects (
    (thoracic12_s  :class  ' thoracic12_s−web−surface
      orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  0 0 0)))
      reference−coordinate−system ^^thoracic12−reference−joint
    )
    (thoracic11_s  :class  ' thoracic11_s−web−surface
      orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  0 0 0)))
      reference−coordinate−system ^^thoracic11−reference−joint
    )
    (thoracic10_s  :class  ' thoracic10_s−web−surface
      orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  0 0 0)))
      reference−coordinate−system ^^thoracic10−reference−joint
    )
    (thoracic9_s  :class  ' thoracic9_s−web−surface
      orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  0 0 0)))
      reference−coordinate−system ^^thoracic9−reference−joint
    )
    (thoracic8_s  :class  ' thoracic8_s−web−surface
      orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  0 0 0)))
      reference−coordinate−system ^^thoracic8−reference−joint
    )
    (thoracic7_s  :class  ' thoracic7_s−web−surface
      orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  0 0 0)))
      reference−coordinate−system ^^thoracic7−reference−joint
    )
    (thoracic6_s  :class  ' thoracic6_s−web−surface
      orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list  0 0 0)))
      reference−coordinate−system ^^thoracic6−reference−joint
    )
    (thoracic5_s  :class  ' thoracic5_s−web−surface
      orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
              (radians-to-degrees 0 ) :y-axis)(rotate (radians-to-degrees 0 )
              :z-axis)(translate ( list  0 0 0)))
         reference-coordinate-system ^^thoracic5-reference-joint
      )
      (thoracic4_s  :class  ' thoracic4_s-web-surface
         orientation  ( list  (rotate  (radians-to-degrees 0 ) :x-axis)(rotate
              (radians-to-degrees 0 ) :y-axis)(rotate (radians-to-degrees 0 )
              :z-axis)(translate ( list  0 0 0)))
         reference-coordinate-system ^^thoracic4-reference-joint
      )
      (thoracic3_s  :class  ' thoracic3_s-web-surface
         orientation  ( list  (rotate  (radians-to-degrees 0 ) :x-axis)(rotate
              (radians-to-degrees 0 ) :y-axis)(rotate (radians-to-degrees 0 )
              :z-axis)(translate ( list  0 0 0)))
         reference-coordinate-system ^^thoracic3-reference-joint
      )
      (thoracic2_s  :class  ' thoracic2_s-web-surface
         orientation  ( list  (rotate  (radians-to-degrees 0 ) :x-axis)(rotate
              (radians-to-degrees 0 ) :y-axis)(rotate (radians-to-degrees 0 )
              :z-axis)(translate ( list  0 0 0)))
         reference-coordinate-system ^^thoracic2-reference-joint
      )
      (thoracic1_s  :class  ' thoracic1_s-web-surface
         orientation  ( list  (rotate  (radians-to-degrees 0 ) :x-axis)(rotate
              (radians-to-degrees 0 ) :y-axis)(rotate (radians-to-degrees 0 )
              :z-axis)(translate ( list  0 0 0)))
         reference-coordinate-system ^^thoracic1-reference-joint
      )
   )
)
(define-class abd-sub-geometry-class
   :inherit-from (
      object
   )
   :properties  (
      Abdomen-reference-joint (default nil)
      Abd_L_L1-reference-joint (default nil)
      Abd_L_L2-reference-joint (default nil)
      Abd_L_L3-reference-joint (default nil)
      Abd_L_L4-reference-joint (default nil)
      Abd_L_L5-reference-joint (default nil)
      Abd_R_L1-reference-joint (default nil)
      Abd_R_L2-reference-joint (default nil)
      Abd_R_L3-reference-joint (default nil)
      Abd_R_L4-reference-joint (default nil)
      Abd_R_L5-reference-joint (default nil)
   )
   :subobjects  (
   )
```

```
)
(define−class hand−sub−geometry−class
  :inherit−from (
    object
  )
  :properties (
    hand_R−reference−joint (default nil)
    hand_L−reference−joint (default nil)
  )
  :subobjects (
    (pisiform :class ' pisiform−web−surface
      orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
          :z−axis)(translate (list −0.013388 −0.009886 −0.010593)))
      reference−coordinate−system ^^hand_R−reference−joint
    )
    (lunate :class ' lunate−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^hand_R−reference−joint
    )
    (scaphoid :class ' scaphoid−web−surface
      orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
          :z−axis)(translate (list 0.012345 −0.004464 −0.001254)))
      reference−coordinate−system ^^hand_R−reference−joint
    )
    (triquetrum :class ' triquetrum−web−surface
      orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
          :z−axis)(translate (list −0.010784 −0.007499 −0.001289)))
      reference−coordinate−system ^^hand_R−reference−joint
    )
    (hamate :class ' hamate−web−surface
      orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
          :z−axis)(translate (list −0.006977 −0.017549 0.001577)))
      reference−coordinate−system ^^hand_R−reference−joint
    )
    (capitate :class ' capitate−web−surface
      orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
          :z−axis)(translate (list 0.003992 −0.015054 0.002327)))
      reference−coordinate−system ^^hand_R−reference−joint
    )
    (trapezoid :class ' trapezoid−web−surface
      orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
```

```
            :z−axis)(translate (list 0.013135 −0.019116 −0.000137)))
        reference−coordinate−system ^^hand_R−reference−joint
    )
    (trapezium :class ' trapezium−web−surface
        orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
            :z−axis)(translate (list 0.019285 −0.019623 −0.007981)))
        reference−coordinate−system ^^hand_R−reference−joint
    )
    (1mc :class ' 1mc−web−surface
        orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
            :z−axis)(translate (list 0.026485 −0.025023 −0.010481)))
        reference−coordinate−system ^^hand_R−reference−joint
    )
    (2mc :class ' 2mc−web−surface
        orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
            :z−axis)(translate (list 0.018677 −0.052674 0.007359)))
        reference−coordinate−system ^^hand_R−reference−joint
    )
    (3mc :class ' 3mc−web−surface
        orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
            :z−axis)(translate (list 0.004469 −0.054293 0.009704)))
        reference−coordinate−system ^^hand_R−reference−joint
    )
    (4mc :class ' 4mc−web−surface
        orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
            :z−axis)(translate (list −0.008054 −0.055573 0.00584)))
        reference−coordinate−system ^^hand_R−reference−joint
    )
    (5mc :class ' 5mc−web−surface
        orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
            :z−axis)(translate (list −0.017904 −0.049737 −0.001891)))
        reference−coordinate−system ^^hand_R−reference−joint
    )
    (thumbprox :class ' thumbprox−web−surface
        orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
            :z−axis)(translate (list 0.042985 −0.054223 −0.023181)))
        reference−coordinate−system ^^hand_R−reference−joint
    )
    (thumbdist :class ' thumbdist−web−surface
        orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
            :z−axis)(translate (list 0.056985 −0.080123 −0.033281)))
    reference−coordinate−system ˆˆhand_R−reference−joint
  )
  (2proxph :class ' 2proxph−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list 0.022178 −0.080917 0.010979)))
    reference−coordinate−system ˆˆhand_R−reference−joint
  )
  (2midph :class ' 2midph−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list 0.029695 −0.12219 0.018305)))
    reference−coordinate−system ˆˆhand_R−reference−joint
  )
  (2distph :class ' 2distph−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list 0.033028 −0.14708 0.019525)))
    reference−coordinate−system ˆˆhand_R−reference−joint
  )
  (3proxph :class ' 3proxph−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list 0.004709 −0.080583 0.011482)))
    reference−coordinate−system ˆˆhand_R−reference−joint
  )
  (3midph :class ' 3midph−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list 0.006359 −0.12479 0.017712)))
    reference−coordinate−system ˆˆhand_R−reference−joint
  )
  (3distph :class ' 3distph−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list 0.007724 −0.15384 0.019666)))
    reference−coordinate−system ˆˆhand_R−reference−joint
  )
  (4proxph :class ' 4proxph−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list −0.009884 −0.079262 0.005667)))
    reference−coordinate−system ˆˆhand_R−reference−joint
  )
  (4midph :class ' 4midph−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list −0.013412 −0.11952 0.007012)))
    reference−coordinate−system ˆˆhand_R−reference−joint
```

```
)
(4distph  :class  ' 4distph−web−surface
    orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list −0.015729 −0.14431 0.007575)))
    reference−coordinate−system ^^hand_R−reference−joint
)
(5proxph :class  ' 5proxph−web−surface
    orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list −0.019501 −0.071168 −0.003387)))
    reference−coordinate−system ^^hand_R−reference−joint
)
(5midph :class  ' 5midph−web−surface
    orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list −0.02478 −0.10673 −0.006266)))
    reference−coordinate−system ^^hand_R−reference−joint
)
(5distph  :class  ' 5distph−web−surface
    orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list −0.027651 −0.12741 −0.008509)))
    reference−coordinate−system ^^hand_R−reference−joint
)
( pisiform_l   :class  ' pisiform_l−web−surface
    orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list −0.013388 −0.009886 0.010593)))
    reference−coordinate−system ^^hand_L−reference−joint
)
( lunate_l   :class  ' lunate_l−web−surface
    orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
         :z−axis)(translate ( list  0 0 0)))
    reference−coordinate−system ^^hand_L−reference−joint
)
(scaphoid_l  :class  ' scaphoid_l−web−surface
    orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list 0.012345 −0.004464 0.001254)))
    reference−coordinate−system ^^hand_L−reference−joint
)
(triquetrum_l  :class  ' triquetrum_l−web−surface
    orientation  ( list  (rotate  (radians−to−degrees −0 ) :x−axis)(rotate
         (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
         :z−axis)(translate (list −0.010784 −0.007499 0.001289)))
    reference−coordinate−system ^^hand_L−reference−joint
)
```

```
(hamate_l :class ' hamate_l−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list −0.006977 −0.017549 −0.001577)))
    reference−coordinate−system ^^hand_L−reference−joint
)
( capitate_l :class ' capitate_l−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.003992 −0.015054 −0.002327)))
    reference−coordinate−system ^^hand_L−reference−joint
)
( trapezoid_l :class ' trapezoid_l−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.013135 −0.019116 0.000137)))
    reference−coordinate−system ^^hand_L−reference−joint
)
(trapezium_l :class ' trapezium_l−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.019285 −0.019623 0.007981)))
    reference−coordinate−system ^^hand_L−reference−joint
)
(1mc_l :class ' 1mc_l−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.026485 −0.025023 0.010481)))
    reference−coordinate−system ^^hand_L−reference−joint
)
(2mc_l :class ' 2mc_l−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.018677 −0.052674 −0.007359)))
    reference−coordinate−system ^^hand_L−reference−joint
)
(3mc_l :class ' 3mc_l−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.004469 −0.054293 −0.009704)))
    reference−coordinate−system ^^hand_L−reference−joint
)
(4mc_l :class ' 4mc_l−web−surface
    orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list −0.008054 −0.055573 −0.00584)))
    reference−coordinate−system ^^hand_L−reference−joint
)
(5mc_l :class ' 5mc_l−web−surface
   orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list −0.017904 −0.049737 0.001891)))
   reference−coordinate−system ^^hand_L−reference−joint
)
(thumbprox_l :class ' thumbprox_l−web−surface
   orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.042985 −0.054223 0.023181)))
   reference−coordinate−system ^^hand_L−reference−joint
)
(thumbdist_l :class ' thumbdist_l−web−surface
   orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.056985 −0.080123 0.033281)))
   reference−coordinate−system ^^hand_L−reference−joint
)
(2proxph_l :class ' 2proxph_l−web−surface
   orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.022178 −0.080917 −0.010979)))
   reference−coordinate−system ^^hand_L−reference−joint
)
(2midph_l :class ' 2midph_l−web−surface
   orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.029695 −0.12219 −0.018305)))
   reference−coordinate−system ^^hand_L−reference−joint
)
(2distph_l :class ' 2distph_l−web−surface
   orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.033028 −0.14708 −0.019525)))
   reference−coordinate−system ^^hand_L−reference−joint
)
(3proxph_l :class ' 3proxph_l−web−surface
   orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.004709 −0.080583 −0.011482)))
   reference−coordinate−system ^^hand_L−reference−joint
)
(3midph_l :class ' 3midph_l−web−surface
   orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
        (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
        :z−axis)(translate (list 0.006359 −0.12479 −0.017712)))
   reference−coordinate−system ^^hand_L−reference−joint
)
(3distph_l :class ' 3distph_l−web−surface
   orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                :z−axis)(translate (list 0.007724 −0.15384 −0.019666)))
           reference−coordinate−system ^^hand_L−reference−joint
       )
       (4proxph_l :class ' 4proxph_l−web−surface
           orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                :z−axis)(translate (list −0.009884 −0.079262 −0.005667)))
           reference−coordinate−system ^^hand_L−reference−joint
       )
       (4midph_l :class ' 4midph_l−web−surface
           orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                :z−axis)(translate (list −0.013412 −0.11952 −0.007012)))
           reference−coordinate−system ^^hand_L−reference−joint
       )
       (4distph_l :class ' 4distph_l−web−surface
           orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                :z−axis)(translate (list −0.015729 −0.14431 −0.007575)))
           reference−coordinate−system ^^hand_L−reference−joint
       )
       (5proxph_l :class ' 5proxph_l−web−surface
           orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                :z−axis)(translate (list −0.019501 −0.071168 0.003387)))
           reference−coordinate−system ^^hand_L−reference−joint
       )
       (5midph_l :class ' 5midph_l−web−surface
           orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                :z−axis)(translate (list −0.02478 −0.10673 0.006266)))
           reference−coordinate−system ^^hand_L−reference−joint
       )
       (5distph_l :class ' 5distph_l−web−surface
           orientation ( list (rotate (radians−to−degrees −0 ) :x−axis)(rotate
                (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees −0 )
                :z−axis)(translate (list −0.027651 −0.12741 0.008509)))
           reference−coordinate−system ^^hand_L−reference−joint
       )
   )
)
(define−class lumbar−sub−geometry−class
   :inherit−from (
       object
   )
   :properties (
       color (default "white")
```

```
      lumbar5−reference−joint (default nil)
      lumbar4−reference−joint (default nil)
      lumbar3−reference−joint (default nil)
      lumbar2−reference−joint (default nil)
      lumbar1−reference−joint (default nil)
   )
   :subobjects (
      (lumbar5 :class ' lumbar5−web−surface
         orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list 0 0 0)))
         reference−coordinate−system ^^lumbar5−reference−joint
      )
      (lumbar4 :class ' lumbar4−web−surface
         orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list 0 0 0)))
         reference−coordinate−system ^^lumbar4−reference−joint
      )
      (lumbar3 :class ' lumbar3−web−surface
         orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list 0 0 0)))
         reference−coordinate−system ^^lumbar3−reference−joint
      )
      (lumbar2 :class ' lumbar2−web−surface
         orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list 0 0 0)))
         reference−coordinate−system ^^lumbar2−reference−joint
      )
      (lumbar1 :class ' lumbar1−web−surface
         orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list 0 0 0)))
         reference−coordinate−system ^^lumbar1−reference−joint
      )
   )
)
(define−class rib−sub−geometry−class
   :inherit−from (
      object
   )
   :properties (
      rib12_R−reference−joint (default nil)
      rib11_R−reference−joint (default nil)
      rib10_R−reference−joint (default nil)
      rib9_R−reference−joint (default nil)
      rib8_R−reference−joint (default nil)
```

```
    rib7_R−reference−joint (default nil)
    rib6_R−reference−joint (default nil)
    rib5_R−reference−joint (default nil)
    rib4_R−reference−joint (default nil)
    rib3_R−reference−joint (default nil)
    rib2_R−reference−joint (default nil)
    rib1_R−reference−joint (default nil)
    rib12_L−reference−joint (default nil)
    rib11_L−reference−joint (default nil)
    rib10_L−reference−joint (default nil)
    rib9_L−reference−joint (default nil)
    rib8_L−reference−joint (default nil)
    rib7_L−reference−joint (default nil)
    rib6_L−reference−joint (default nil)
    rib5_L−reference−joint (default nil)
    rib4_L−reference−joint (default nil)
    rib3_L−reference−joint (default nil)
    rib2_L−reference−joint (default nil)
    rib1_L−reference−joint (default nil)
    sternum−reference−joint (default nil)
  )
  :subobjects (
    (Rib12R :class ' Rib12R−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
           (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
           :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib12_R−reference−joint
    )
    (Rib11R :class ' Rib11R−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
           (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
           :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib11_R−reference−joint
    )
    (Rib10R :class ' Rib10R−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
           (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
           :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib10_R−reference−joint
    )
    (Rib9R :class ' Rib9R−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
           (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
           :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib9_R−reference−joint
    )
    (Rib8R :class ' Rib8R−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
           (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
```

```
            :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^rib8_R−reference−joint
    )
    (Rib7R :class ' Rib7R−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^rib7_R−reference−joint
    )
    (Rib6R :class ' Rib6R−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^rib6_R−reference−joint
    )
    (Rib5R :class ' Rib5R−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^rib5_R−reference−joint
    )
    (Rib4R :class ' Rib4R−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^rib4_R−reference−joint
    )
    (Rib3R :class ' Rib3R−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^rib3_R−reference−joint
    )
    (Rib2R :class ' Rib2R−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^rib2_R−reference−joint
    )
    (Rib1R :class ' Rib1R−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^rib1_R−reference−joint
    )
    (Rib12L :class ' Rib12L−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
            (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
            :z−axis)(translate ( list  0 0 0)))
```

```
      reference−coordinate−system ^^rib12_L−reference−joint
   )
   (Rib11L :class ' Rib11L−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib11_L−reference−joint
   )
   (Rib10L :class ' Rib10L−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib10_L−reference−joint
   )
   (Rib9L :class ' Rib9L−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib9_L−reference−joint
   )
   (Rib8L :class ' Rib8L−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib8_L−reference−joint
   )
   (Rib7L :class ' Rib7L−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib7_L−reference−joint
   )
   (Rib6L :class ' Rib6L−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib6_L−reference−joint
   )
   (Rib5L :class ' Rib5L−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib5_L−reference−joint
   )
   (Rib4L :class ' Rib4L−web−surface
      orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
          (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
          :z−axis)(translate ( list 0 0 0)))
      reference−coordinate−system ^^rib4_L−reference−joint
```

```
        )
        (Rib3L :class ' Rib3L−web−surface
            orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                  :z−axis)(translate ( list 0 0 0)))
            reference−coordinate−system ^^rib3_L−reference−joint
        )
        (Rib2L :class ' Rib2L−web−surface
            orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                  :z−axis)(translate ( list 0 0 0)))
            reference−coordinate−system ^^rib2_L−reference−joint
        )
        (Rib1L :class ' Rib1L−web−surface
            orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                  :z−axis)(translate ( list 0 0 0)))
            reference−coordinate−system ^^rib1_L−reference−joint
        )
        (Sternum :class ' Sternum−web−surface
            orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                  :z−axis)(translate ( list 0 0 0)))
            reference−coordinate−system ^^sternum−reference−joint
        )
    )
)
(define−class arm−sub−geometry−class
    :inherit−from (
        object
    )
    :properties (
        clavicle_R−reference−joint (default nil)
        scapula_R−reference−joint (default nil)
        humerus_R−reference−joint (default nil)
        ulna_R−reference−joint (default nil)
        radius_R−reference−joint (default nil)
        clavicle_L−reference−joint (default nil)
        scapula_L−reference−joint (default nil)
        humerus_L−reference−joint (default nil)
        ulna_L−reference−joint (default nil)
        radius_L−reference−joint (default nil)
    )
    :subobjects (
        ( clavicle :class ' clavicle−web−surface
            orientation ( list (rotate (radians−to−degrees 0 ) :x−axis)(rotate
                  (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
                  :z−axis)(translate ( list 0 0 0)))
            reference−coordinate−system ^^clavicle_R−reference−joint
    )
    (scapula  :class  ' scapula−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^scapula_R−reference−joint
    )
    (humerus :class  ' humerus−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^humerus_R−reference−joint
    )
    (ulna  :class  ' ulna−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^ulna_R−reference−joint
    )
    (radius  :class  ' radius−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^radius_R−reference−joint
    )
    ( clavicle_l   :class  ' clavicle_l−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^clavicle_L−reference−joint
    )
    (scapula_l  :class  ' scapula_l−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^scapula_L−reference−joint
    )
    (humerus_l :class  ' humerus_l−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^humerus_L−reference−joint
    )
    (ulna_l  :class  ' ulna_l−web−surface
        orientation  ( list  (rotate  (radians−to−degrees 0 ) :x−axis)(rotate
              (radians−to−degrees 0 ) :y−axis)(rotate (radians−to−degrees 0 )
              :z−axis)(translate ( list  0 0 0)))
        reference−coordinate−system ^^ulna_L−reference−joint
    )
```

```
        ( radius_l  :class  ' radius_l-web-surface
            orientation  ( list  (rotate  (radians-to-degrees 0 ) :x-axis)(rotate
                 (radians-to-degrees 0 ) :y-axis)(rotate (radians-to-degrees 0 )
                 :z-axis)(translate ( list  0 0 0)))
            reference-coordinate-system ^^radius_L-reference-joint
        )
    )
)
(define-class ground-sub-geometry-class
    :inherit-from (
        object
    )
    :properties  (
        ground-reference-joint (default nil)
        sacrum-reference-joint (default nil)
        pelvis-reference-joint  (default  nil)
    )
    :subobjects  (
        (sacrum :class  ' sacrum-web-surface
            orientation  ( list  (rotate  (radians-to-degrees 0 ) :x-axis)(rotate
                 (radians-to-degrees 0 ) :y-axis)(rotate (radians-to-degrees 0 )
                 :z-axis)(translate ( list  0 0 0)))
            reference-coordinate-system ^^sacrum-reference-joint
        )
        ( pelvis_rv   :class  ' pelvis_rv-web-surface
            orientation  ( list  (rotate  (radians-to-degrees 0 ) :x-axis)(rotate
                 (radians-to-degrees 0 ) :y-axis)(rotate (radians-to-degrees 0 )
                 :z-axis)(translate ( list  0 0 0)))
            reference-coordinate-system ^^pelvis-reference-joint
        )
        ( pelvis_lv   :class  ' pelvis_lv-web-surface
            orientation  ( list  (rotate  (radians-to-degrees 0 ) :x-axis)(rotate
                 (radians-to-degrees 0 ) :y-axis)(rotate (radians-to-degrees 0 )
                 :z-axis)(translate ( list  0 0 0)))
            reference-coordinate-system ^^pelvis-reference-joint
        )
    )
)
```

C.10 Data model class

Listing C.10: Movement data-model-mixin

```
(define-class joint-class-data-model-class
    :inherit-from (data-model-node-mixin body-class)
    :properties  (
        property-objects-list
        ( list
            "Hip joint  (range -120 - 120)"
```

```
          ( list  (the  superior  hip−rotation−r self)  '(automatic−apply? t)
          )
          ( list  (the  superior  hip−aduction−r self) '(automatic−apply? t)
          )
          ( list  (the  superior  hip−flexion−r self)  '(automatic−apply? t)
          )
          "Knee joint (range −120 − 10)"
          ( list  (the  superior  knee−angle−r self) '(automatic−apply? t)
          )
          "Ankle joint  (range −90 − 90)"
          ( list  (the  superior  ankle−angle−r self) '(automatic−apply? t)
          )
          "Subtalar joint  (range −90 − 90)"
          ( list  (the  superior  subtalar−angle−r self)  '(automatic−apply? t)
          )
          "Toe joint range (range −90 − 90)"
          ( list  (the  mtp−angle−r self) '(automatic−apply? t)
          )
       )
       (hip−flexion−r :class  'editable−data−property−class
          label "hip−flexion−r"
          formula :inherit−formula
       )
       (hip−rotation−r :class  'editable−data−property−class
          label "hip−rotation−r"
          formula :inherit−formula
       )
       (hip−aduction−r :class 'editable−data−property−class
          label "hip−aduction−r"
          formula :inherit−formula
       )
       (knee−angle−r :class 'editable−data−property−class
          label "knee−angle−r"
          formula :inherit−formula
       )
       (ankle−angle−r :class 'editable−data−property−class
          label "ankle−angle−r"
          formula :inherit−formula
       )
       (subtalar−angle−r :class  'editable−data−property−class
          label "subtalar−angle−r"
          formula :inherit−formula
       )
       (mtp−angle−r :class 'editable−data−property−class
          label "mtp−angle−r"
          formula :inherit−formula
       )
   )
)
```

C.11 Web-surface-classes

Listing C.11: Web-surface-classes

```
(define-class treadmill-web-surface
   :inherit-from(web-surface-object)
    :properties (
      nodes-file "C:\\PytonWorkspace\\treadmill_points.dat"
      con-file  "C:\\PytonWorkspace\\treadmill_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define-class sacrum-web-surface
   :inherit-from(web-surface-object)
    :properties (
      nodes-file "C:\\PytonWorkspace\\sacrum_points.dat"
      con-file  "C:\\PytonWorkspace\\sacrum_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define-class pelvis-web-surface
   :inherit-from(web-surface-object)
    :properties (
      nodes-file "C:\\PytonWorkspace\\pelvis_points.dat"
      con-file  "C:\\PytonWorkspace\\pelvis_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define-class l_pelvis-web-surface
   :inherit-from(web-surface-object)
    :properties (
      nodes-file "C:\\PytonWorkspace\\l_pelvis_points.dat"
      con-file  "C:\\PytonWorkspace\\l_pelvis_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define-class femur-web-surface
   :inherit-from(web-surface-object)
    :properties (
      nodes-file "C:\\PytonWorkspace\\femur_points.dat"
      con-file  "C:\\PytonWorkspace\\femur_connectivity.dat"
```

```
        cleanup? nil
        method 2
    )
)

(define-class tibia-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\tibia_points.dat"
        con-file   "C:\\PytonWorkspace\\tibia_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class fibula-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\fibula_points.dat"
        con-file   "C:\\PytonWorkspace\\fibula_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class talus-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\talus_points.dat"
        con-file   "C:\\PytonWorkspace\\talus_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class foot-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\foot_points.dat"
        con-file   "C:\\PytonWorkspace\\foot_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class bofoot-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\bofoot_points.dat"
        con−file   "C:\\PytonWorkspace\\bofoot_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class l_femur−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\l_femur_points.dat"
        con−file   "C:\\PytonWorkspace\\l_femur_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class l_tibia−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\l_tibia_points.dat"
        con−file   "C:\\PytonWorkspace\\l_tibia_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class l_fibula−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\l_fibula_points.dat"
        con−file   "C:\\PytonWorkspace\\l_fibula_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class l_talus−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\l_talus_points.dat"
        con−file   "C:\\PytonWorkspace\\l_talus_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class l_foot−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes-file "C:\\PytonWorkspace\\l_foot_points.dat"
        con-file   "C:\\PytonWorkspace\\l_foot_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class l_bofoot-web-surface
    :inherit-from(web-surface-object)
    :properties  (
        nodes-file "C:\\PytonWorkspace\\l_bofoot_points.dat"
        con-file   "C:\\PytonWorkspace\\l_bofoot_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class hat_spine-web-surface
    :inherit-from(web-surface-object)
    :properties  (
        nodes-file "C:\\PytonWorkspace\\hat_spine_points.dat"
        con-file   "C:\\PytonWorkspace\\hat_spine_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class hat_jaw-web-surface
    :inherit-from(web-surface-object)
    :properties  (
        nodes-file "C:\\PytonWorkspace\\hat_jaw_points.dat"
        con-file   "C:\\PytonWorkspace\\hat_jaw_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class hat_skull-web-surface
    :inherit-from(web-surface-object)
    :properties  (
        nodes-file "C:\\PytonWorkspace\\hat_skull_points.dat"
        con-file   "C:\\PytonWorkspace\\hat_skull_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class hat_ribs-web-surface
    :inherit-from(web-surface-object)
    :properties  (
        nodes−file ”C:\\PytonWorkspace\\hat_ribs_points.dat”
        con−file   ”C:\\PytonWorkspace\\hat_ribs_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(in−package :aml)

(define−class sacrum−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file ”C:\\PytonWorkspace\\sacrum_points.dat”
        con−file   ”C:\\PytonWorkspace\\sacrum_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class pelvis_rv−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file ”C:\\PytonWorkspace\\pelvis_rv_points.dat”
        con−file   ”C:\\PytonWorkspace\\pelvis_rv_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class pelvis_lv−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file ”C:\\PytonWorkspace\\pelvis_lv_points.dat”
        con−file   ”C:\\PytonWorkspace\\pelvis_lv_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class lumbar5−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file ”C:\\PytonWorkspace\\lumbar5_points.dat”
        con−file   ”C:\\PytonWorkspace\\lumbar5_connectivity.dat”
        cleanup? nil
        method 2
    )
)
```

```
(define−class lumbar4−web−surface
   :inherit−from(web−surface−object)
   :properties  (
      nodes−file "C:\\PytonWorkspace\\lumbar4_points.dat"
      con−file   "C:\\PytonWorkspace\\lumbar4_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class lumbar3−web−surface
   :inherit−from(web−surface−object)
   :properties  (
      nodes−file "C:\\PytonWorkspace\\lumbar3_points.dat"
      con−file   "C:\\PytonWorkspace\\lumbar3_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class lumbar2−web−surface
   :inherit−from(web−surface−object)
   :properties  (
      nodes−file "C:\\PytonWorkspace\\lumbar2_points.dat"
      con−file   "C:\\PytonWorkspace\\lumbar2_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class lumbar1−web−surface
   :inherit−from(web−surface−object)
   :properties  (
      nodes−file "C:\\PytonWorkspace\\lumbar1_points.dat"
      con−file   "C:\\PytonWorkspace\\lumbar1_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class thoracic12_s−web−surface
   :inherit−from(web−surface−object)
   :properties  (
      nodes−file "C:\\PytonWorkspace\\thoracic12_s_points.dat"
      con−file   "C:\\PytonWorkspace\\thoracic12_s_connectivity.dat"
      cleanup? nil
      method 2
   )
```

```
)

(define−class thoracic11_s−web−surface
   :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\thoracic11_s_points.dat"
        con−file   "C:\\PytonWorkspace\\thoracic11_s_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class thoracic10_s−web−surface
   :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\thoracic10_s_points.dat"
        con−file   "C:\\PytonWorkspace\\thoracic10_s_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class thoracic9_s−web−surface
   :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\thoracic9_s_points.dat"
        con−file   "C:\\PytonWorkspace\\thoracic9_s_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class thoracic8_s−web−surface
   :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\thoracic8_s_points.dat"
        con−file   "C:\\PytonWorkspace\\thoracic8_s_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class thoracic7_s−web−surface
   :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\thoracic7_s_points.dat"
        con−file   "C:\\PytonWorkspace\\thoracic7_s_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class thoracic6_s-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\thoracic6_s_points.dat"
        con-file   "C:\\PytonWorkspace\\thoracic6_s_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class thoracic5_s-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\thoracic5_s_points.dat"
        con-file   "C:\\PytonWorkspace\\thoracic5_s_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class thoracic4_s-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\thoracic4_s_points.dat"
        con-file   "C:\\PytonWorkspace\\thoracic4_s_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class thoracic3_s-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\thoracic3_s_points.dat"
        con-file   "C:\\PytonWorkspace\\thoracic3_s_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class thoracic2_s-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\thoracic2_s_points.dat"
        con-file   "C:\\PytonWorkspace\\thoracic2_s_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class thoracic1_s−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\thoracic1_s_points.dat"
        con−file  "C:\\PytonWorkspace\\thoracic1_s_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class rotatedcerv7−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\rotatedcerv7_points.dat"
        con−file  "C:\\PytonWorkspace\\rotatedcerv7_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class cerv6−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\cerv6_points.dat"
        con−file  "C:\\PytonWorkspace\\cerv6_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class cerv5−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\cerv5_points.dat"
        con−file  "C:\\PytonWorkspace\\cerv5_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class cerv4−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\cerv4_points.dat"
        con−file  "C:\\PytonWorkspace\\cerv4_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class cerv3−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\cerv3_points.dat"
        con−file   "C:\\PytonWorkspace\\cerv3_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class cerv2−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\cerv2_points.dat"
        con−file   "C:\\PytonWorkspace\\cerv2_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class cerv1−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\cerv1_points.dat"
        con−file   "C:\\PytonWorkspace\\cerv1_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class jaw−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\jaw_points.dat"
        con−file   "C:\\PytonWorkspace\\jaw_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class skull−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\skull_points.dat"
        con−file  "C:\\PytonWorkspace\\skull_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class Rib12R−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\Rib12R_points.dat"
        con−file  "C:\\PytonWorkspace\\Rib12R_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class Rib11R−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\Rib11R_points.dat"
        con−file  "C:\\PytonWorkspace\\Rib11R_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class Rib10R−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\Rib10R_points.dat"
        con−file  "C:\\PytonWorkspace\\Rib10R_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class Rib9R−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\Rib9R_points.dat"
        con−file  "C:\\PytonWorkspace\\Rib9R_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class Rib8R−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\Rib8R_points.dat"
        con−file   "C:\\PytonWorkspace\\Rib8R_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class Rib7R−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\Rib7R_points.dat"
        con−file   "C:\\PytonWorkspace\\Rib7R_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class Rib6R−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\Rib6R_points.dat"
        con−file   "C:\\PytonWorkspace\\Rib6R_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class Rib5R−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\Rib5R_points.dat"
        con−file   "C:\\PytonWorkspace\\Rib5R_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class Rib4R−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\Rib4R_points.dat"
        con−file   "C:\\PytonWorkspace\\Rib4R_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class Rib3R−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file ”C:\\PytonWorkspace\\Rib3R_points.dat”
        con−file   ”C:\\PytonWorkspace\\Rib3R_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class Rib2R−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file ”C:\\PytonWorkspace\\Rib2R_points.dat”
        con−file   ”C:\\PytonWorkspace\\Rib2R_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class Rib1R−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file ”C:\\PytonWorkspace\\Rib1R_points.dat”
        con−file   ”C:\\PytonWorkspace\\Rib1R_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class Rib12L−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file ”C:\\PytonWorkspace\\Rib12L_points.dat”
        con−file   ”C:\\PytonWorkspace\\Rib12L_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class Rib11L−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file ”C:\\PytonWorkspace\\Rib11L_points.dat”
        con−file   ”C:\\PytonWorkspace\\Rib11L_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class Rib10L−web−surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\Rib10L_points.dat"
        con-file  "C:\\PytonWorkspace\\Rib10L_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class Rib9L-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\Rib9L_points.dat"
        con-file  "C:\\PytonWorkspace\\Rib9L_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class Rib8L-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\Rib8L_points.dat"
        con-file  "C:\\PytonWorkspace\\Rib8L_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class Rib7L-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\Rib7L_points.dat"
        con-file  "C:\\PytonWorkspace\\Rib7L_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class Rib6L-web-surface
    :inherit-from(web-surface-object)
    :properties (
        nodes-file "C:\\PytonWorkspace\\Rib6L_points.dat"
        con-file  "C:\\PytonWorkspace\\Rib6L_connectivity.dat"
        cleanup? nil
        method 2
    )
)
```

```
(define−class Rib5L−web−surface
   :inherit−from(web−surface−object)
   :properties  (
      nodes−file "C:\\PytonWorkspace\\Rib5L_points.dat"
      con−file   "C:\\PytonWorkspace\\Rib5L_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class Rib4L−web−surface
   :inherit−from(web−surface−object)
   :properties  (
      nodes−file "C:\\PytonWorkspace\\Rib4L_points.dat"
      con−file   "C:\\PytonWorkspace\\Rib4L_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class Rib3L−web−surface
   :inherit−from(web−surface−object)
   :properties  (
      nodes−file "C:\\PytonWorkspace\\Rib3L_points.dat"
      con−file   "C:\\PytonWorkspace\\Rib3L_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class Rib2L−web−surface
   :inherit−from(web−surface−object)
   :properties  (
      nodes−file "C:\\PytonWorkspace\\Rib2L_points.dat"
      con−file   "C:\\PytonWorkspace\\Rib2L_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class Rib1L−web−surface
   :inherit−from(web−surface−object)
   :properties  (
      nodes−file "C:\\PytonWorkspace\\Rib1L_points.dat"
      con−file   "C:\\PytonWorkspace\\Rib1L_connectivity.dat"
      cleanup? nil
      method 2
   )
)
```

```
(define−class Sternum−web−surface
   :inherit−from(web−surface−object)
    :properties (
      nodes−file "C:\\PytonWorkspace\\Sternum_points.dat"
      con−file   "C:\\PytonWorkspace\\Sternum_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class clavicle−web−surface
   :inherit−from(web−surface−object)
    :properties (
      nodes−file "C:\\PytonWorkspace\\clavicle_points.dat"
      con−file   "C:\\PytonWorkspace\\clavicle_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class scapula−web−surface
   :inherit−from(web−surface−object)
    :properties (
      nodes−file "C:\\PytonWorkspace\\scapula_points.dat"
      con−file   "C:\\PytonWorkspace\\scapula_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class humerus−web−surface
   :inherit−from(web−surface−object)
    :properties (
      nodes−file "C:\\PytonWorkspace\\humerus_points.dat"
      con−file   "C:\\PytonWorkspace\\humerus_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class ulna−web−surface
   :inherit−from(web−surface−object)
    :properties (
      nodes−file "C:\\PytonWorkspace\\ulna_points.dat"
      con−file   "C:\\PytonWorkspace\\ulna_connectivity.dat"
      cleanup? nil
      method 2
   )
```

```
)

(define−class radius−web−surface
   :inherit−from(web−surface−object)
   :properties (
      nodes−file "C:\\PytonWorkspace\\radius_points.dat"
      con−file  "C:\\PytonWorkspace\\radius_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class pisiform−web−surface
   :inherit−from(web−surface−object)
   :properties (
      nodes−file "C:\\PytonWorkspace\\pisiform_points.dat"
      con−file  "C:\\PytonWorkspace\\pisiform_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class lunate−web−surface
   :inherit−from(web−surface−object)
   :properties (
      nodes−file "C:\\PytonWorkspace\\lunate_points.dat"
      con−file  "C:\\PytonWorkspace\\lunate_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class scaphoid−web−surface
   :inherit−from(web−surface−object)
   :properties (
      nodes−file "C:\\PytonWorkspace\\scaphoid_points.dat"
      con−file  "C:\\PytonWorkspace\\scaphoid_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class triquetrum−web−surface
   :inherit−from(web−surface−object)
   :properties (
      nodes−file "C:\\PytonWorkspace\\triquetrum_points.dat"
      con−file  "C:\\PytonWorkspace\\triquetrum_connectivity.dat"
      cleanup? nil
      method 2
```

```
    )
)

(define−class hamate−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\hamate_points.dat"
        con−file  "C:\\PytonWorkspace\\hamate_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class capitate−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\capitate_points.dat"
        con−file  "C:\\PytonWorkspace\\capitate_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class trapezoid−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\trapezoid_points.dat"
        con−file  "C:\\PytonWorkspace\\trapezoid_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class trapezium−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\trapezium_points.dat"
        con−file  "C:\\PytonWorkspace\\trapezium_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 1mc−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\1mc_points.dat"
        con−file  "C:\\PytonWorkspace\\1mc_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 2mc−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\2mc_points.dat"
        con−file   "C:\\PytonWorkspace\\2mc_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 3mc−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\3mc_points.dat"
        con−file   "C:\\PytonWorkspace\\3mc_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 4mc−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\4mc_points.dat"
        con−file   "C:\\PytonWorkspace\\4mc_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 5mc−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\5mc_points.dat"
        con−file   "C:\\PytonWorkspace\\5mc_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class thumbprox−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes−file "C:\\PytonWorkspace\\thumbprox_points.dat"
        con−file   "C:\\PytonWorkspace\\thumbprox_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class thumbdist−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file ”C:\\PytonWorkspace\\thumbdist_points.dat”
        con−file   ”C:\\PytonWorkspace\\thumbdist_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class 2proxph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file ”C:\\PytonWorkspace\\2proxph_points.dat”
        con−file   ”C:\\PytonWorkspace\\2proxph_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class 2midph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file ”C:\\PytonWorkspace\\2midph_points.dat”
        con−file   ”C:\\PytonWorkspace\\2midph_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class 2distph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file ”C:\\PytonWorkspace\\2distph_points.dat”
        con−file   ”C:\\PytonWorkspace\\2distph_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class 3proxph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file ”C:\\PytonWorkspace\\3proxph_points.dat”
        con−file  ”C:\\PytonWorkspace\\3proxph_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class 3midph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file ”C:\\PytonWorkspace\\3midph_points.dat”
        con−file  ”C:\\PytonWorkspace\\3midph_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class 3distph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file ”C:\\PytonWorkspace\\3distph_points.dat”
        con−file  ”C:\\PytonWorkspace\\3distph_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class 4proxph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file ”C:\\PytonWorkspace\\4proxph_points.dat”
        con−file  ”C:\\PytonWorkspace\\4proxph_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class 4midph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file ”C:\\PytonWorkspace\\4midph_points.dat”
        con−file  ”C:\\PytonWorkspace\\4midph_connectivity.dat”
        cleanup? nil
        method 2
    )
)

(define−class 4distph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\4distph_points.dat"
        con−file   "C:\\PytonWorkspace\\4distph_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 5proxph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\5proxph_points.dat"
        con−file   "C:\\PytonWorkspace\\5proxph_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 5midph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\5midph_points.dat"
        con−file   "C:\\PytonWorkspace\\5midph_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 5distph−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\5distph_points.dat"
        con−file   "C:\\PytonWorkspace\\5distph_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class clavicle_l−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\clavicle_l_points.dat"
        con−file   "C:\\PytonWorkspace\\clavicle_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class scapula_l−web−surface
    :inherit−from(web−surface−object)
    :properties  (
        nodes-file "C:\\PytonWorkspace\\scapula_l_points.dat"
        con-file   "C:\\PytonWorkspace\\scapula_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class humerus_l-web-surface
    :inherit-from(web-surface-object)
    :properties  (
        nodes-file "C:\\PytonWorkspace\\humerus_l_points.dat"
        con-file   "C:\\PytonWorkspace\\humerus_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class ulna_l-web-surface
    :inherit-from(web-surface-object)
    :properties  (
        nodes-file "C:\\PytonWorkspace\\ulna_l_points.dat"
        con-file   "C:\\PytonWorkspace\\ulna_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class radius_l-web-surface
    :inherit-from(web-surface-object)
    :properties  (
        nodes-file "C:\\PytonWorkspace\\radius_l_points.dat"
        con-file   "C:\\PytonWorkspace\\radius_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class pisiform_l-web-surface
    :inherit-from(web-surface-object)
    :properties  (
        nodes-file "C:\\PytonWorkspace\\pisiform_l_points.dat"
        con-file   "C:\\PytonWorkspace\\pisiform_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define-class lunate_l-web-surface
   :inherit-from(web-surface-object)
   :properties  (
      nodes-file "C:\\PytonWorkspace\\lunate_l_points.dat"
      con-file   "C:\\PytonWorkspace\\lunate_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define-class scaphoid_l-web-surface
   :inherit-from(web-surface-object)
   :properties  (
      nodes-file "C:\\PytonWorkspace\\scaphoid_l_points.dat"
      con-file   "C:\\PytonWorkspace\\scaphoid_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define-class triquetrum_l-web-surface
   :inherit-from(web-surface-object)
   :properties  (
      nodes-file "C:\\PytonWorkspace\\triquetrum_l_points.dat"
      con-file   "C:\\PytonWorkspace\\triquetrum_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define-class hamate_l-web-surface
   :inherit-from(web-surface-object)
   :properties  (
      nodes-file "C:\\PytonWorkspace\\hamate_l_points.dat"
      con-file   "C:\\PytonWorkspace\\hamate_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define-class capitate_l-web-surface
   :inherit-from(web-surface-object)
   :properties  (
      nodes-file "C:\\PytonWorkspace\\capitate_l_points.dat"
      con-file   "C:\\PytonWorkspace\\capitate_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)
```

```
(define-class trapezoid_l-web-surface
   :inherit-from(web-surface-object)
    :properties  (
       nodes-file ”C:\\PytonWorkspace\\trapezoid_l_points.dat”
       con-file   ”C:\\PytonWorkspace\\trapezoid_l_connectivity.dat”
       cleanup? nil
       method 2
   )
)

(define-class trapezium_l-web-surface
   :inherit-from(web-surface-object)
    :properties  (
       nodes-file ”C:\\PytonWorkspace\\trapezium_l_points.dat”
       con-file   ”C:\\PytonWorkspace\\trapezium_l_connectivity.dat”
       cleanup? nil
       method 2
   )
)

(define-class 1mc_l-web-surface
   :inherit-from(web-surface-object)
    :properties  (
       nodes-file ”C:\\PytonWorkspace\\1mc_l_points.dat”
       con-file   ”C:\\PytonWorkspace\\1mc_l_connectivity.dat”
       cleanup? nil
       method 2
   )
)

(define-class 2mc_l-web-surface
   :inherit-from(web-surface-object)
    :properties  (
       nodes-file ”C:\\PytonWorkspace\\2mc_l_points.dat”
       con-file   ”C:\\PytonWorkspace\\2mc_l_connectivity.dat”
       cleanup? nil
       method 2
   )
)

(define-class 3mc_l-web-surface
   :inherit-from(web-surface-object)
    :properties  (
       nodes-file ”C:\\PytonWorkspace\\3mc_l_points.dat”
       con-file   ”C:\\PytonWorkspace\\3mc_l_connectivity.dat”
       cleanup? nil
       method 2
   )
)
```

```
(define−class 4mc_l−web−surface
   :inherit−from(web−surface−object)
   :properties (
      nodes−file "C:\\PytonWorkspace\\4mc_l_points.dat"
      con−file   "C:\\PytonWorkspace\\4mc_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class 5mc_l−web−surface
   :inherit−from(web−surface−object)
   :properties (
      nodes−file "C:\\PytonWorkspace\\5mc_l_points.dat"
      con−file   "C:\\PytonWorkspace\\5mc_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class thumbprox_l−web−surface
   :inherit−from(web−surface−object)
   :properties (
      nodes−file "C:\\PytonWorkspace\\thumbprox_l_points.dat"
      con−file   "C:\\PytonWorkspace\\thumbprox_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class thumbdist_l−web−surface
   :inherit−from(web−surface−object)
   :properties (
      nodes−file "C:\\PytonWorkspace\\thumbdist_l_points.dat"
      con−file   "C:\\PytonWorkspace\\thumbdist_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class 2proxph_l−web−surface
   :inherit−from(web−surface−object)
   :properties (
      nodes−file "C:\\PytonWorkspace\\2proxph_l_points.dat"
      con−file   "C:\\PytonWorkspace\\2proxph_l_connectivity.dat"
      cleanup? nil
      method 2
   )
```

```
)

(define−class 2midph_l−web−surface
   :inherit−from(web−surface−object)
    :properties  (
      nodes−file "C:\\PytonWorkspace\\2midph_l_points.dat"
      con−file   "C:\\PytonWorkspace\\2midph_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class 2distph_l−web−surface
   :inherit−from(web−surface−object)
    :properties  (
      nodes−file "C:\\PytonWorkspace\\2distph_l_points.dat"
      con−file   "C:\\PytonWorkspace\\2distph_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class 3proxph_l−web−surface
   :inherit−from(web−surface−object)
    :properties  (
      nodes−file "C:\\PytonWorkspace\\3proxph_l_points.dat"
      con−file   "C:\\PytonWorkspace\\3proxph_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class 3midph_l−web−surface
   :inherit−from(web−surface−object)
    :properties  (
      nodes−file "C:\\PytonWorkspace\\3midph_l_points.dat"
      con−file   "C:\\PytonWorkspace\\3midph_l_connectivity.dat"
      cleanup? nil
      method 2
   )
)

(define−class 3distph_l−web−surface
   :inherit−from(web−surface−object)
    :properties  (
      nodes−file "C:\\PytonWorkspace\\3distph_l_points.dat"
      con−file   "C:\\PytonWorkspace\\3distph_l_connectivity.dat"
      cleanup? nil
      method 2
    )
)

(define−class 4proxph_l−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\4proxph_l_points.dat"
        con−file   "C:\\PytonWorkspace\\4proxph_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 4midph_l−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\4midph_l_points.dat"
        con−file   "C:\\PytonWorkspace\\4midph_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 4distph_l−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\4distph_l_points.dat"
        con−file   "C:\\PytonWorkspace\\4distph_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 5proxph_l−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\5proxph_l_points.dat"
        con−file   "C:\\PytonWorkspace\\5proxph_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 5midph_l−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\5midph_l_points.dat"
        con−file   "C:\\PytonWorkspace\\5midph_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)

(define−class 5distph_l−web−surface
    :inherit−from(web−surface−object)
    :properties (
        nodes−file "C:\\PytonWorkspace\\5distph_l_points.dat"
        con−file   "C:\\PytonWorkspace\\5distph_l_connectivity.dat"
        cleanup? nil
        method 2
    )
)
```