FACULTY
OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 8
30614 Plzeň
Czech Republic

Computer muscle modelling

State of the Art and Concept of PhD. Thesis

Ing. Martin Červenka

Technical Report No. DCSE/TR-2022-03
20^{th} of August, 2022
Distribution: public

Technical Report No. DCSE/TR-2022-03
20^{th} of August, 2022

Computer muscle modelling

State of the Art and Concept of PhD. Thesis

Ing. Martin Červenka

Abstract

Nowadays, computer muscle modelling plays a more and more important role in the threatment of various musculoskeletal diseases. For example, to find the appropriate artificial joint replacement, physicians should know the anatomy and dynamic properties of the specific patient. Various models and algorithms help physicians. In this report, state-of-the-art methods are discussed. Then, a competitive modified position-based dynamics approach is presented. The speed/quality ratio of the proposed method is adjustable and achieves certain results over a specified period.

This work was supported by the projects SGS-2019-016 of the Czech Ministry of Education, GA 17-05534S of the Czech Science Foundation and partially by the project IDEG-2021-027 of the West Bohemia University.

Copies of this report are available on
`http://www.kiv.zcu.cz/en/research/publications/`
or by surface mail on request sent to the following address:

University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 8
30614 Plzeň
Czech Republic

Copyright © 2022 University of West Bohemia, Czech Republic

Acknowledgements

I greatly appreciate the long-term help and support of my supervisor *doc. Ing. Josef Kohout, Ph.D.* I am als very greatful for all of the advice that *prof. Ing. Vaclav Skala, CSc* provided, especially for the discussion on radial-basis functions. My thanks also go to my partner and family, without whose indirect help this work would not have been possible.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Muscle modelling pipeline . . . 2

2 Data aquisition **5**
- 2.1 Muscle-tendon units . . . 5
- 2.2 Muscle attachments . . . 6
- 2.3 Non-invasive methods . . . 7
- 2.4 Invasive methods . . . 8
- 2.5 Physiological signals . . . 9
- 2.6 Data registration . . . 10

3 Estimation techniques **12**
- 3.1 Constant and piecewise linear estimation . . . 13
- 3.2 Bezier curve . . . 13
- 3.3 Catmull-Rom spline . . . 14
- 3.4 Discrette Fourier transform . . . 14
- 3.5 Radial basis functions . . . 15
  - 3.5.1 Centre point distribution . . . 16
  - 3.5.2 Polynomial extension . . . 17

4 Finite element method **19**
- 4.1 Problem formulation in strong form . . . 19
  - 4.1.1 Laplace equation . . . 19
  - 4.1.2 Poisson equation . . . 20
  - 4.1.3 Second-order partial differential equation . . . 20
- 4.2 Boundary condition . . . 21
- 4.3 Weak formulation . . . 22
- 4.4 One-dimensional problem . . . 22
  - 4.4.1 Discretization . . . 22
  - 4.4.2 Triangular basis . . . 23
- 4.5 Multidimensional triangular basis . . . 24
  - 4.5.1 2D triangular basis . . . 24
- 4.6 Examples . . . 26

4.6.1 One-dimensional problem 26
4.6.2 Two-dimensional problem 29
4.6.3 Surface model problems 33

5 Existing methods **35**
5.1 Hill-type model 36
5.2 Via-points 37
5.3 Wrapping obstacles 38
5.4 Finite element method 39
5.5 Other optimalization problems 41
5.5.1 Mass-spring system 41
5.5.2 ARAP 43
5.5.3 PBD 48

6 Experiments and Results **58**
6.1 PBD results 58
6.1.1 Collision detection and response 58
6.1.2 Muscle decomposition 59
6.1.3 Artificial data 62
6.1.4 Iliacus 63
6.1.5 Gluteus maximus 64
6.1.6 Other muscles 65
6.1.7 Quantitative tests 65
6.1.8 Fibre length 69
6.1.9 Speed 69
6.2 Preliminary ARAP Results 71

7 Conclusion & Future Work **74**
7.1 The ambitious goal 74
7.2 ARAP & PBD 76

A Publications **77**

B Other activities **79**

Notation

a, b, c	scalar variables (italics)
f, g, h	scalar functions (parameters not specified)
x, y, z	cartesian coordinates of a point/vector in 3D
$f(x, y), g(x, y), h(x, y)$	scalar functions with two parameters
$\mathbf{f}(x), \mathbf{g}(x), \mathbf{h}(x)$	vector functions with single parameter
$\mathbf{a}, \mathbf{b}, \mathbf{c}$	vector variables (italics, bold)
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	matrix variables (italics, bold, uppercase)
D_f	domain of function f
H_f	codomain of function f
$\times$	cartesian/cross product
$\cdot$	dot product
$\mathbb{R}, \mathbb{C}$	real, complex numbers
$\lvert\mathbf{A}\rvert$	determinant of matrix $\mathbf{A}$
$\lvert\lvert\mathbf{a}\rvert\rvert = \lvert\lvert\mathbf{a}\rvert\rvert_2$	L2 (euclidean) norm of vector $\mathbf{a}$
$\ldots, \vdots, \cdots, \ddots$	pattern follows

Chapter 1

Introduction

Modern technology and rapid scientific and industrial progress are leading us to live faster and less healthy lives. The consequence is that psychical stress is all around us, and our physical body suffers. Fortunately, modern healthcare is improving daily, perfecting old and new techniques, treatments, technologies, tools and more.

Physical health is what this work focuses on. Tones of people go to their jobs every day to work hard, stand all day, or do other monotonic jobs, which may affect the human body in the way overused joints wear out (causing osteoarthritis). Alternatively, the bones weaken (osteoporosis) due to external influences, such as monotonic work and inappropriate diet. In order to treat and predict these diseases, physicians need to know as much as possible about the human body, especially the individual patient.

The history of studying the human body goes back to 1800 BC when our ancestors knew the basics, such as bones, muscles and internal organs approximate location of the average human. The medical field is developing rapidly and allows in-vivo exploration in particular. Combined with modern technology, a physician can model a patient in-vivo on his or her computer, which helps in making crucial treatment decisions.

The following text describes state-of-the-art methods of computerized muscle modelling, my contribution to date on this topic and some suggestions for further research.

1.1 Motivation

Osteoporosis is a disease with high prevalence [86], causing bone weakening and, subsequently, bone fractures. Osteoarthritis is another musculoskeletal disease where bone joints gradually wear out after time. In the osteoporosis case, knowing the force values applied to bones by surrounding muscles helps physicians estimate fracture predictions and suitable treatment even before the fracture happens. In osteoarthritis cases, the internal forces involved in

the bone movement must be known to choose a suitable artificial joint [65], as long as the wrong choice may lead to pain for the patient.

However, many more diseases and injuries require the modelling and predicting of musculoskeletal system movement. Another problem is patellar dislocation, "with an estimated incidence rate of 43-77 per 100,000 individuals in children and adolescents" [7]. Even in treating stroke and hemiplegic diseases, computer muscle modelling can be used [92]. All these mentioned problems (and many others) are leading researchers to develop a satisfactory model of the musculoskeletal system.

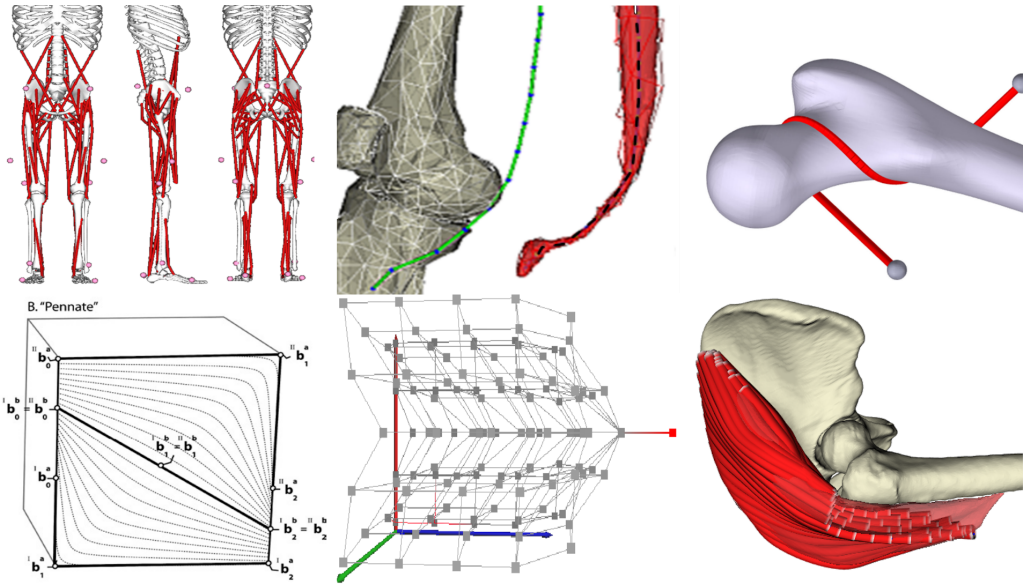

Figure 1.1: Various muscle modelling approaches. From top to bottom, from left to right: Hill-type model [50], Via-points model [39], Wrapping obstacles model [56], Finite element model [22], Mass-spring model [40], PBD model

There are currently many muscle modelling approaches, some of them are shown in Fig. 1.1. This report will describe the most significant of these in more detail.

1.2 Muscle modelling pipeline

The whole muscle modelling process consists of many steps, starting with the acquisition of suitable raw data and its subsequent transformation into a useful form. The last step is the formulation of the mathematical model, where the main problem is (among others) the definition of the muscle-bone interaction.

An example of a complex pipeline (consisting of data acquisition, model building and inverse kinematics) includes the following steps:

1. obtaining raw data of the patient at rest (contains muscles, bones, muscle attachment areas, and movement data according to chapter 2),
2. extraction and transformation of the raw data into a useful form, using:
   (a) segmentation – separation of different types of tissues (if they are distinguishable). Segmentation can be manual, semi-automatic or automatic (depends on the complexity of the segmentation),
   (b) registration – mapping of data from different measurements and modalities, see section 2.6,
   (c) approximation and interpolation – if the data is partially corrupted or some parts are missing, these methods are used. They are further described in chapter 3,
3. acquiring some of general apriori knowledge, determined by human anatomy, such as:
   (a) defines how the attachment areas will be determined (calculation based on apriori knowledge only or based on the measured muscle attachment areas from chapter 2),
   (b) defining how, for example, a bone is connected by a joint to another bone or how a muscle is connected to a set of bones (attachment areas), etc.,
   (c) defining physiological parameters of studied muscles, such as internal muscle architecture (e.g. figre arrangement: parallel, pennate, etc.), optimal (resting) length
4. creating a mathematical model that requires (these approaches are further described in chapter 5):
   (a) defining the data space (discretized or continuous),
   (b) defining the shape of the data (triangular surface mesh, tetrahedral volumetric mesh, scattered data... / surface defined by Fourier series, implicit RBF,...)
   (c) defining the interaction between muscle and bone models and thus determining whether or not a transformation of the measured data is necessary.
5. transformation back into a useful form.

The next chapter 2 discusses the data used in musculoskeletal models, and chapter 3 describes the basic mathematical procedures used for approximating and interpolating purposes in computer muscle modelling approaches. Chapter 4 describes the basic algorithm for modelling physical

phenomena, the finite element method, and chapter 5 presents existing approaches, followed by chapter 6, containing practical experiments. Conclusion and future work are described in the last chapter, 7.

Chapter 2

Data aquisition

The basis of quality computer muscle modelling success is to obtain relevant data; therefore, methods and possibilities of data acquisition are presented.

There are two categories of data, general and personalized data. Personalized data are those that have been measured on a given subject (in the case of a patient's illness or optimizing an athlete's performance). In contrast, general data is data from someone else that will not be a good match to a given patient (see [67], [43] and [71] for further reasons; the main problem is with the high variability of muscle attachment sites between subjects).

2.1 Muscle-tendon units

The human body is a very complex system, so an accurate model of it (if we could be able to build it) would be useless. Therefore, all state-of-the-art musculoskeletal models have to simplify the problem to some extent, and many phenomena are not taken into account (due to the complexity of the human body). All the musculoskeletal models described below are just a set of bones and muscles. Some methods also require models of the attachment areas or information about the direction of the muscle fibres. All models must be described by a geometric model. The muscle models described throughout the following text actually approximate MTUs (muscle-tendon units), which are made up of muscles, tendons, cartilages, aponeurosis, fats, blood vessels, etc. Muscle (MTU) and bone models are often approximated by a triangular mesh, so this would be implicitly assumed in the text unless otherwise stated. The problem with model acquisition is due to the resolution of acquisition methods discussed below.

2.2 Muscle attachments

To determine, which part of a muscle is connected to a certain part of a bone, a muscle attachment area has to be defined. The determination of the muscle attachment area can be done automatically or manually, and for the decision, some additional data may be required. If the data are not provided beforehand, there are two possibilities:

1. Fix the set of points of the muscle model that are "close enough" to the bone surface or penetrate the bone before the movement happens.

2. Obtain a dedicated set of points which defines the attachment area (e.g. boundary points [17] , scattered points over the area [46], points from boudnary curve) from a user.

In the first case, the implementation is quite simple. However, there is a problem that some muscle parts may be fixed incorrectly. The most common example is a muscle part fixed to a bone joint, which introduces further problems (forcing the muscle part into the joint etc.). The second case is more robust but requires obtaining the whole area from the boundary points. We addressed this issue in our paper [46], which works for simple cases but may fail for more curved and complicated muscle attachment area shapes, mainly where the surface bends several times. In the paper, we reached maximal accuracy of 78.74%, which may be insufficient for some applications.

In the paper, we also experimented with surface plate reconstruction from the set of point using RBF (described in section 3.5), achieving acceptable results even for more complicated boundaries.

2.3 Non-invasive methods

Noninvasive methods are methods of data acquisition that allow for the extraction of personalized data. CT, MRI and PET-based methods are the most commonly used. CT is a well-known method (invented in the early 1970s by A. M. Cormack) that allows obtaining several X-ray images combined into a 3D model (using the Radon transformation). Because there is a large difference (in Hounsfield units HU) between bones and soft tissues), X-rays images are suitable to distinguish between bones and soft tissues. Soft tissue types are almost indistinguishable due to small HU differences (bones: 200-600 HU, all soft tissue: 40-100 HU). The image of real data obtained with the CT scanner is shown in Fig. 2.1.

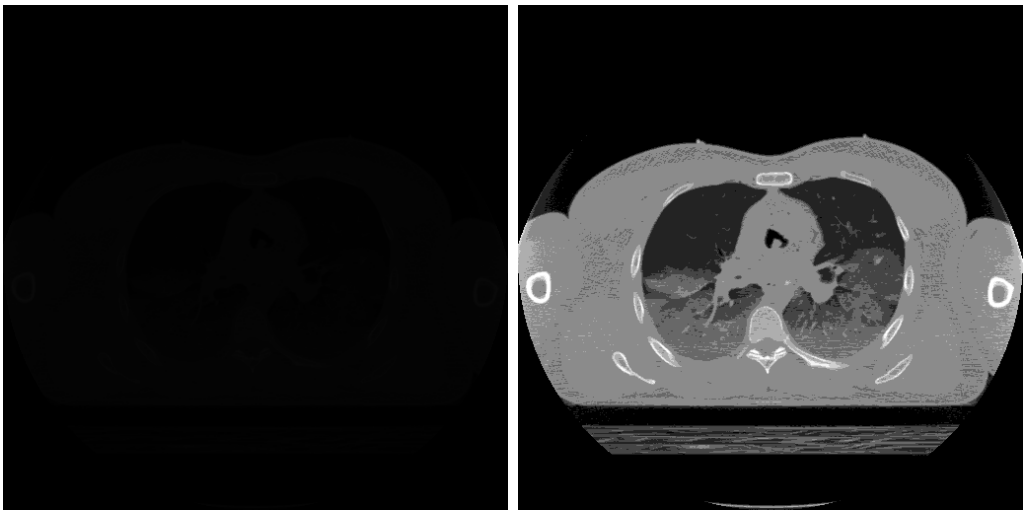

Figure 2.1: CT result. Original data [60] on the left, same data adjusted using the linear transfer function on the right.

In the early 1980s, the first MRI was installed. MRI is based on the measurement of the spin echo after the application of a strong magnetic pulse. The main advantages of MRI are that MRI does not produce ionizing radiation, and MRI also generally has better visibility of soft tissues. These advantages are strongly counteracted by a much longer acquisition time, which is unpleasant to the physicians but even more for the patients. An example of an MRI result is shown in Fig. 2.2.

More recently (in the late 1980s), diffusion-weighted imaging (DWI) was introduced as an improved version of MRI. Its basic principle is to exploit the rate of tissue water diffusion. For more detailed information, see. e.g. [68,72].

Diffusion-tensor imaging is a version of DWI where diffusion is defined using tensor arithmetics. For example, DTI acquisition has been used to determine the pennate angle of various human muscles [53], which is one of the key parameters in musculoskeletal modelling that affects the magnitude

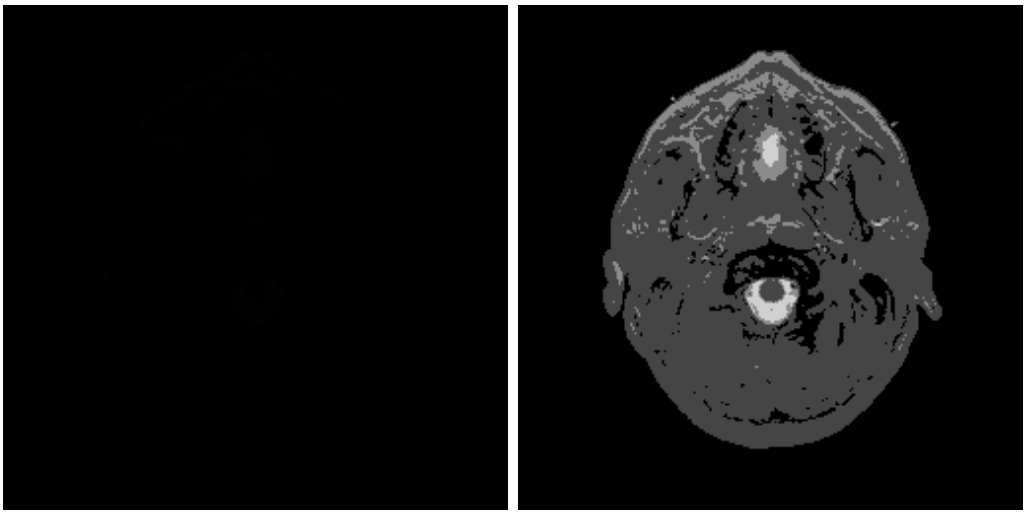

Figure 2.2: MRI result. Original data [60] on the left, The same data adjusted using the linear transfer function on the right.

of the resulting forces. However, none of the acquisition methods can determine muscle attachment areas (bounded single/multiple area(s) of the bone to where the given muscle is attached). Unfortunately, this measurement cannot be performed by a noninvasive method because the attachment areas are not visible (or very poorly visible) on imaging. An at least partially personalized approach can be used, where personalized bone and muscle models are acquired, and then the attachment areas are estimated (by an expert, a probabilistic model [29], etc.). Although individual muscles are also difficult to distinguish using imaging methods, there are techniques [63] that can do so automatically.

2.4 Invasive methods

For obvious ethical reasons, invasive methods (such as dissecting and other cutting) cannot be used directly on the patient. Therefore, experiments on the cadaver are necessary to obtain more detailed data. These experiments allow producing more accurate models with precise measurements. Moreover, some of the musculoskeletal features (which are indistinguishable on the scanning devices) can be acquired. The most problematic are muscle/tendon separation and attachment areas.

An example of obtaining the data in an invasive way is shown in Fig. 2.3. This dataset has been acquired as a part of The Visible Human Project [60]; however, there are also Visible Human Chinese [93], and Visible Human Korea [70]. Two datasets were obtained (male and female), where each subject was frozen and sliced. 256x256 MRI images were produced from the male subject, spaced by 4mm. CT scans are 512x512 images and 1mm apart. In

figure 2.3, there are the data from 70mm photographs (digitised to 4096x2700 px) also 1mm apart. The female subject's accuracy was three times better, so the distance between slices was 0.33mm.

More details about acquiring these data can be found in [80], where the Visible Human Male procedure is described.

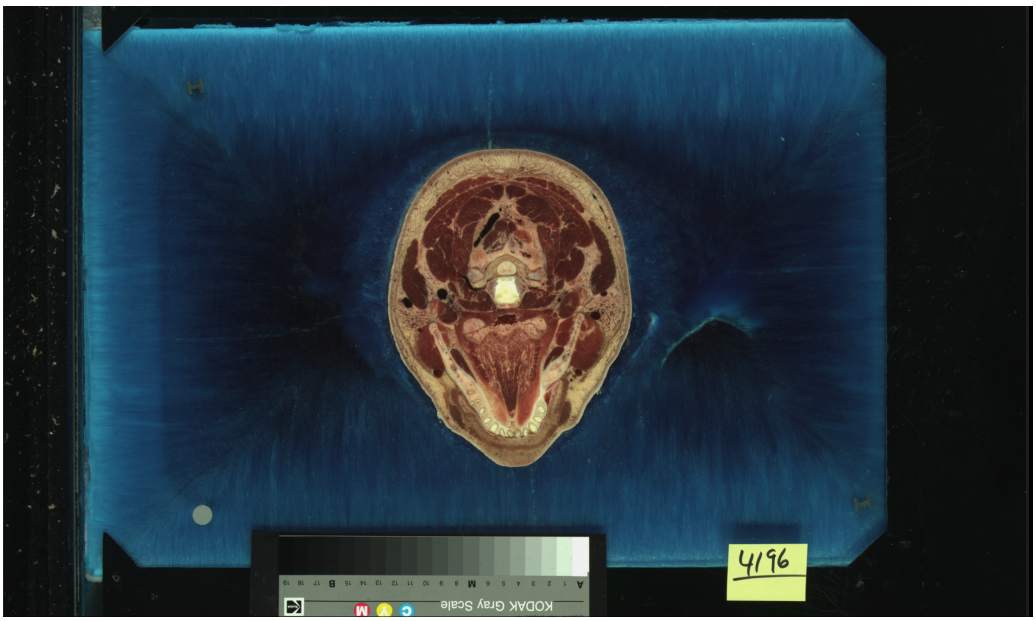

Figure 2.3: The detailed cryosection image of the Visible Human Male [80].

Fukuda et al. [29] describe the other invasive method. The hip region was dissected into individual muscles, and the attachment areas of the muscles were tracked down using an optical tracker. They build a probabilistic model using eight different cadaver specimens. However, they were outliers present in the measurement, which had to be removed manually afterwards.

A cadaveric study from Carbone et al. [12]. The study produced a dataset (TLEM 2.0 - Twente Lower Extremity Model) from cadaveric dissection. This model includes, in total, 166 muscle-tendon elements for each leg. The details about the procedure can be found in [12].

The invasive methods are also used for anatomic fibres and tendon measurements, due to the fact that these more detailed features of the muscle are complicated to measure using noninvasive methods. Lee et al. [53] describe the problem and use cadaveric data to estimate the pennate angle of a given muscle for that reason.

2.5 Physiological signals

The last set of data to consider is the physiological signals. The best example of a physiological signal is EMG (Electromyography). This diagnostic

procedure measures motor neuron activation signals responsible for muscle activation. This knowledge can be used to model a muscle movement, modelling the muscle as realistic as possible. However, the muscle is so complex and varies from person to person. The EMG is performed using a needle electrode inserted directly into the patient's muscle, and its electrical activity is recorded. Because of its invasionality, the direct personalized model is not preferred, and the EMG is measured on the skin surface instead of using a set of sensors. The skin measurement, however, is imprecise because it cannot be located directly. These sensors can be placed on the shaved and cleaned skin by sticking a patch or using rubber bands (the latter may be omitted for more intense movement measurements). If the sensors are located well, the measurements can be sufficient for some cases.

Movement data are also suitable for this task, not only because the movement can be learned from it but also because it can be verified. These data can be obtained using location sensors put on the patient. Then he/she is requested to perform some movement activities (walking, running, jumping), and the movement of various parts of his/her body is recorded.

These signals are more often used for muscle modelling using direct kinematics instead of the inverse one. Since the main focus of this report is on the inverse kinematics approaches, these data are just mentioned here, and further details are omitted. The movement data are used in further text to determine the target bone locations, from which the muscle shape can be inversely determined.

This chapter shows that the data may vary significantly. If the data do not quite correspond to each other, registration should be done (e.g. Li et al. [55], described in the next section 2.6). Nevertheless, the data are too coarse or incomplete in many cases. For these reasons, an approximation and interpolation method has to be used.

2.6 Data registration

This brief introduction to data acquisition modalities introduces the data's main issue: its diversity. Individual data have various positions, rotations, and alignments, measured in different conditions. Because of this and also according to the pipeline in 1.2, registration has to be done beforehand. The registration is mapping a set of data measured by some modality to a different set measured by a different modality or in different conditions. The main two registration classes are rigid and non-rigid registration, the first one preserves the shape and scale of the transformed object, and the latter does not. With promising results, the registration of musculoskeletal models has already been done by Zhao et al. [94]. However, the resulting data may become self-intersecting. We also dealt a similar problem, which is described in the bachelor thesis [16] (see Fig. 2.4, where 3D muscle surface and surface

muscle fibres (both measured with different modalities) have been registered using the algorithm from Li et al. [55], an elastic registration approach, based on surface plate reconstruction. The fibre models should lie on the plate, further used to produce more dense fibres on the muscle surface. Unluckily, the results were not as promising as it was at the beginning, mainly because the noisiness of the input data was transferred and even enhanced to the output fibres.

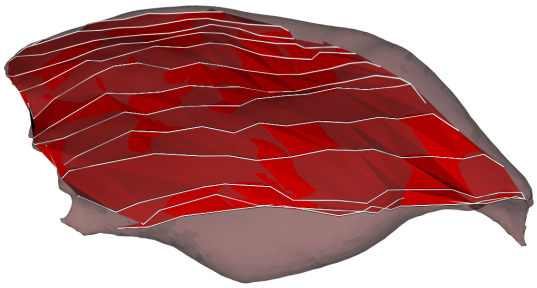

Figure 2.4: Muscle fibre model registration to the muscle model surface [16], using a modified algorithm from Li et al. [55].

Chapter 3

Estimation techniques

The next step in the muscle modelling pipeline (see section 1.2) is an approximation or interpolation[1]. A key ingredient to make a smooth model is to be able to approximate/interpolate in the part of a model (muscle, bone, muscle fibre, attachment area and many more) where no data exists. Extrapolation is not considered under the assumption that the border of the model is known, accurate and well defined.

The Bezier curves for muscle modelling purposes have been used by Delp et al. [22] followed by Kohout and Kukacka [48], but it has been shown [47] that the modelled muscle fibres may self-intersect. To solve the issue, a Catmull-Rom spline has been proposed.

Kohout and Cholt [47] proposed this approximation for the muscle fibre model to make the fibres as smooth as the Bezier curves and non-self-intersecting. Although this approach seems successful, estimation using the higher smoothness curve would probably overperform their approach.

The next approach worth mentioning here is the radial basis function (RBF) approximation and interpolation approach. The RBF has been designed to tackle scattered data approximation and interpolation problems. This approach is commonly used in many areas, e.g. image reconstruction [82], neural networks [36], surface reconstruction [69] and much more. This approach was firstly proposed by Hardy [34]. Its main advantage is that it can reach, in theory, infinite smoothness C^∞, when, e.g. Gaussian RBFs are used as a basis function, which is in contrast with Bezier or Catmull-Rom, where polynomial basis functions are used. We already used RBF for the muscle modelling purposes in our paper [46], more specifically for muscle attachment area estimation.

This section describes these estimation techniques, which are suitable for muscle modelling problems in detail. Each of the methods listed will be described in a particular dimension. However, extending the idea to higher/lower dimensions is usually possible.

[1]Word "estimation" will be used further for both approximation and interpolation to simplify the rest of the text.

3.1 Constant and piecewise linear estimation

The most simple estimation is the constant one. The idea is straightforward, and we assign to the unknown independent value the value of the closest independent value.

The piecewise linear estimation in one dimension takes into account two closest values. The intermediate values lie on the straight line going through the two closest values. Figure 3.1 shows the constant and piecewise linear estimation. The red is the constant, and the linear estimation is green.

These kinds of estimations have a considerable drawback. They are at most C^0 smooth. Moreover, they can be non-continuous. It means they are not differentiable on a nonempty finite set of points. The main issue is that this is not how the muscle works. Muscle fibres and, therefore, whole muscles are continuous and smooth. This model would not represent a physically accurate real muscle, not to mention the problem the consequence is that the resulting simulation would not look attractive to the user.

The higher dimension polynomial can be used to solve the problem of a low degree of smoothness. There will be two mentioned approaches, although there are many more to consider.

3.2 Bezier curve

Bezier curve is a curve that has two boundary points and a set of control points. The bezier curve without any control point would collapse to a linear curve. If the curve has only one control point, it is called the quadratic Bezier curve. The cubic Bezier curve has two control points. The first boundary point, together with the first control point, forms a derivative of the curve's start, similar to the last control point and the second boundary point forms a derivate of the end of the curve.

The more general recursive definition can be formulated as the bezier curve of degree n is a point-to-point linear combination of the bezier curves of degree $n-1$, one without the first boundary vertex and the other without the second boundary vertex (in the first case, The first control point becomes the first boundary point, and in the second one, the last control point becomes second boundary point). The end of the recursion is secured by the fact that the Bezier curve of degree 1 is just a vertex in space.

The definition is also the binomial distribution of all of the Bezier vertices (both border and all control) while changing the probability of success forms the bezier curve.

$$P(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & -3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P_0} \\ \mathbf{P_1} \\ \mathbf{P_2} \\ \mathbf{P_3} \end{bmatrix} \tag{3.1}$$

The cubic Bezier curve is shown in Figure 3.1 on blue. There can be seen that the curve does not go through the control point. Thus it is instead an approximation method.

3.3 Catmull-Rom spline

A Catmull-Rom spline [15] [90] is a cubic spline that interpolates all four points. Its tangent is calculated using the previous and next control vertex for both internal vertices. Tangents of the border vertices are not clearly defined. However, the vertex itself can be used instead of the missing one. The centripetal variant is defined as[2]:

$$P(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \mathbf{P_0} \\ \mathbf{P_1} \\ \mathbf{P_2} \\ \mathbf{P_3} \end{bmatrix} \tag{3.2}$$

Setting $t \in \langle 0, 1 \rangle$ will produce the Catmull-Rom spline between two middle vertice P_1 and P_2. The general Catmull-Rom spline is, however, defined with a parameter τ:

$$P(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau - 3 & 3 - 2\tau & -\tau \\ -\tau & 2 - \tau & \tau - 2 & \tau \end{bmatrix} \begin{bmatrix} \mathbf{P_0} \\ \mathbf{P_1} \\ \mathbf{P_2} \\ \mathbf{P_3} \end{bmatrix} \tag{3.3}$$

The Catmull-Rom spline overperforms Cubic bezier curves in three main properties. The Catmull-Rom curve goes through all of the control vertices. It does not need to specify first or second derivatives on any vertices. Moreover, the centripetal $\tau = 0.5$ (between uniform $\tau = 0$ and chordal $\tau = 1$) parametrization does not produce artefacts such as cups and self-intersection.

The Catmull-Rom spline is also shown in Figure 3.1, the spline in question is in pink colour.

3.4 Discrette Fourier transform

The discrete Fourier transform (DFT) is beneficial when the data are equidistantly sampled. In this case, DFT decomposes the curve into the sum of individual sinusoidal functions. In the case of interpolation, all of these function has to be added together. An approximation is made otherwise. The advantage of this estimation is that the extrapolation does not diverge (this is

[2]The definitions are in matrix form (derived from recurrent form) to be consistent with other methods.

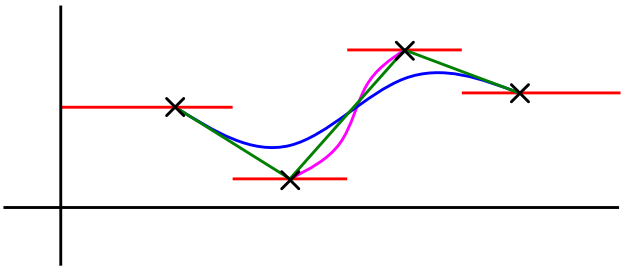

Figure 3.1: Constant (red), piecewise linear (green), Bezier (blue) and Catmull-Rom (pink) interpolation of a set of points.

common in polynomial estimation cases). The main disadvantage is that the data has to be equidistant (there is also non-uniform DFT [6], but it is more complex than DFT). The formula is derived using Euler's formula:

$$F_k = \sum_{n=0}^{N-1} P_n e^{-2\pi i k \frac{n}{N}} \tag{3.4}$$

The inverse operation is performed in the same fashion; however, the unit circle has to be traversed in reverse, and the whole result has to be normed by the number of functions.

$$P_k = \frac{1}{N} \sum_{n=0}^{N-1} F_n e^{2\pi i k \frac{n}{N}} \tag{3.5}$$

The resulting curve will be C^∞ smooth. Many medical signals (like EKG, EEG, and EMG) usually obey the equidistant distribution and have a periodical component. That is the reason it is described here. However, the model's shape usually does not obey equidistant distribution or periodicity, so other estimation methods must be considered. The promising is the radial basis function approximation and interpolation.

The extension to higher dimensions is simple: separate the vector componentwise, calculate DFT and put these components back together.

3.5 Radial basis functions

The RBF approach is formulated mathematically as:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \varphi \left(||\mathbf{x} - \mathbf{P_i}|| \right) \tag{3.6}$$

It is just a linear combination of a set of RBFs φ with centres in points $\mathbf{P_i}$, forced to interpolate the vertices $\mathbf{P_i}$ using suitable weights λ_i. The equation is nothing else than the linear system equation $\mathbf{Ax} = \mathbf{b}$, where matrix $\mathbf{A}$

consists of RBF function values, λ_i is an unknown vector and $f(\mathbf{x})$ will be filled with known values in each of the input points.

In order to approximate, a lower number of RBF may be used, so the problem will be overdetermined and has to be solved, e.g. using the least squares approach (considering some drawbacks, see, e.g. Skala and Kansa [78]).

$$f(\mathbf{x}) = \sum_{i=1}^{M} \lambda_i \varphi\left(||\mathbf{x} - \xi_\mathbf{i}||\right) \tag{3.7}$$

3.5.1 Centre point distribution

The crucial problem is to determine where the centre points ξ_i should be placed. There are some commonly used possibilities, and each of them has (as always) its advantages and drawbacks.

Grid distribution

The grid centre point distribution is where all the centre points are arranged in any form of a grid (regular, cartesian, rectilinear, curvilinear etc.) The grid setup is simple but has major drawbacks. The biggest one is that the regularities of the grid may cause ill-conditionality of the RBF matrix [20] if the shape parameter is not chosen carefully. The other is that it does not reflect the interpolated/approximated function features.

Halton distribution

The Halton distribution [33] is a quazi-random point distribution. The original version is one-dimensional and operates on $(0, 1)$ interval (which can be scaled by a constant). The element of the sequence is described as:

$$\text{Halton}_k(p) = \sum_{i=0}^{\lfloor \log_p k \rfloor} \frac{1}{p^{i+1}} \left(\left\lfloor \frac{k}{p^i} \right\rfloor \bmod p \right) \tag{3.8}$$

where p is an arbitrary prime number, and k is the index of the sequence element. For multidimensional sequences, it is sufficient to choose multiple distinct primes, and the result will be just a vector of the Halton sequences with different prime p. For example, Halton sequence $[2, 3]$ starts with $\left[\left[\frac{1}{2}, \frac{1}{3}\right], \left[\frac{1}{4}, \frac{2}{3}\right], \left[\frac{3}{4}, \frac{1}{9}\right]\right]$. Basically, it divides the $(0, 1)$ space to p equally sized intervals, outputting the border points. Then, each subspace is then subdivided, also outputting the border points. Breath-first politics is used for the traversal, so the distribution is more spread than in the case of the depth-first (recursion) approach.

The same sequence will be obtained if the k will be written in base p, inverted and written after the decimal point. For example of $p = 2$:

$$0.1_2 = \frac{1}{2}, 0.01_2 = \frac{1}{4}, 0.11_2 = \frac{3}{4}, 0.001_2 = \frac{1}{8}, 0.101_2, 0.011_2, 0.111_2... \quad (3.9)$$

The Halton sequence has the main advantage of fewer regularities, leading to ill-conditioned matrix avoidance and maintaining good interval coverage. The disadvantage is that the borders, in most cases, have to be added explicitly (0 and 1 are not included by default).

Iterative greedy search

In most cases, iterative approaches compete with direct methods in mathematics. It is also not an exception in the case of centre point search. The most straightforward approach is to search the single centre point, which decreases the difference between approximated function and approximant. Then, more functions will be searched in the space of the two function difference to lower the difference as much as possible. This procedure will be stopped if the overall error is sufficient or the number of centre points is exhausted.

The main advantage is that the approximant respects the original function features. Also, the shape parameter in each greedy step can be adjusted for a better fit. The disadvantage is its computational complexity. Each "guess" leads to solving a linear equation (which may become pretty large) and the sum of RBF. Some RBFs are expensive to evaluate on the computer.

Distribution with respect to original function

The justifiable approach is to choose the centre points, where the original function has some features. The minima and maxima are likely candidates because the RBFs have their extrema in the centre. Moreover, to cover more, the inflexion points can be included.

We have already dealt with this approach in [21], but in a slightly different scenario. As a result, the "sophisticated placement of radial basis functions significantly improves the quality of the RBF approximation." [21]. The main conclusion of the research was to avoid grid/equidistant centre point distribution as much as possible.

3.5.2 Polynomial extension

Let us imagine a non-zero constant function to approximate with the RBF approach. Without the polynomial extension, it would be difficult to approximate the function accurately. The centre points have to be everywhere to cover the whole function domain. That is why the polynomial is added to deal with constant-like and polynomial-like functions.

The extension includes the polynomial approximation into the RBF equation, expanding the matrix by the same number of rows and columns as the polynomial degree is used. It is done in the case of $2\frac{1}{2}D$ function like this:

$$\begin{bmatrix} \varphi_{11} & \dots & \varphi_{1N} & 1 & x_1 & y_1 \\ \vdots & \ddots & \vdots & 1 & \vdots & \vdots \\ \varphi_{N1} & \dots & \varphi_{NN} & 1 & x_N & y_N \\ 1 & 1 & 1 & 0 & 0 & 0 \\ x_1 & \dots & x_N & 0 & 0 & 0 \\ y_1 & \dots & y_N & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \\ a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.10}$$

where φ_{ij} is the value of the RBF according to the distance from the vertex i to vertex j, x_i and y_i are the coordinates of the centre points, a_0 is the constant coefficient, a_1, a_2 are linear coefficients of the linear expression $a_0 + a_1 x + a_2 y$. The f_i variables are function values of given vertices and λ_i is unknown RBF weight to solve.

This matrix can be further rewritten more compactly:

$$\begin{bmatrix} \mathbf{A} & \mathbf{P} \\ \mathbf{P^T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \lambda \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \tag{3.11}$$

where $\mathbf{A}$ is an RBF submatrix, $\mathbf{P}$ is a polynomial submatrix, $\mathbf{a}$ is a polynomial coefficient subvector, and $\mathbf{f}$ subvector contains the interpolated/approximated function values.

It is sufficient to fill the equation and get good results in practice. The problem, however, is more theoretical. The issue lies in the different units used in the matrix, φ_{ij} is some form of distance, which went through the RBF function (the units are somewhat unknown in the case of Gaussian RBF, it is something like "exponential metres" (e^m) if all constants are discarded. However, the x_i and y_i are in function domain units, usually metres (m). However, this mix of apples and oranges leads to theoretical problems, which are often not mentioned at all [54].

We have studied RBF to some depth [19–21, 76, 77, 84] and can see some potential of it for the approximation purposes. As far as I know, there was no usage of RBF approximation methods used in the muscle modelling field. My current contribution to applying RBF approach to muscle modelling problem is described in the article "Non-planar Surface Shape Reconstruction from a Point Cloud in the Context of Muscles Attachments Estimation" [46], where RBF is used to reconstruct a surface from the set of attachment points.

This chapter explored some of the approximation and interpolation methods, which may be used for data preprocessing but also as a part of some muscle modelling techniques. The first one is the finite element method (FEM), used mainly for modelling various physical phenomena, including muscle modelling. Even FEM uses an approximation method to work.

Chapter 4

Finite element method

The finite element method [5, 51] (FEM in short) is an option to perform physically accurate modelling. It has been successfully used to solve heat transfer problems [75], fluid dynamics [13] and more. Recently, George-Ghiocel et al. [31] used the stress and strain approximation approach in Storz coupling in fire hose coupling.

The mathematical formulation is described first to understand the basics, because partial differential equations (PDE) often define these problems.

4.1 Problem formulation in strong form

It is best to start with the most straightforward problem and work out the most general problem. The so-called "strong" form defines the problem strictly. The weak form will follow afterwards. The easiest problem to discuss is solving the Laplace equation.

4.1.1 Laplace equation

Laplace equation is a basic second-order partial differential equation mainly describing situations of equilibrium and also time-independent phenomena. The Laplace operator has already been described in section 5.5.2; however, it will be described in more detail here.

The Laplace operatorΔ is also used for this purpose, but there it is assumed it is zero everywhere. This assumption also means that there are no *s*ources or *s*inks, where the observed phenomenon could "enter" or "exit" the enclosed observed space. In other words, a closed system is assumed.

$$\nabla^2\psi = \nabla \cdot \nabla\psi = \Delta\psi = 0 \tag{4.1}$$

It is also true that the divergence of the gradient is another definition of the Laplace operator. The definition (4.1) states that there is no source

or sink anywhere in the field because the divergence operator is zero everywhere. If the inner gradient terms are considered, there are no *s*ources or *s*inks of gradients in the field, implying the function has to be linear. In other words, the curvature of the function has to be zero (in a single variable function case, the osculation circle has to have an infinite radius everywhere). This restriction seems to be fairly strict, so more complex equations follow.

4.1.2 Poisson equation

The Poisson equation is a Laplace equation, but with the non-zero right side of the equation, so it is defined:

$$\Delta\psi = f \tag{4.2}$$

The unknown function here is ψ, f function is known. When this equation is compared with the Laplace equation 4.1, it can be noticed that the f function describes the curvature of the ψ function. As an example, if f is a constant everywhere. The ψ function describes a circle or sphere, or hypersphere depending on the number of ψ parameters. The function f may be called a "source term" because the positive value forces the ψ function to be convex (in many circumstances, a "hole") as well as the negative function forces the ψ to be concave (basically said a "hill").

The equation 4.2 also describes the relationship between second-order derivative and a "constant" (meaning some independent term to ψ). Some relation, however, may be needed between second-order derivative, first-order derivative, function ψ itself (zero-order) and "constant" term.

4.1.3 Second-order partial differential equation

In general, a second-order partial differential equation is defined in two variables as follows:

$$A\psi_{xx} + 2B\psi_{xy} + C\psi_{yy} + D\psi_x + E\psi_y + F\psi + G = 0 \tag{4.3}$$

The subscript notation indicates partial derivative by subscripted variable. If $B = D = E = F = G = 0$ and $A = C = 1$, the Laplace equation is obtained. If $G \neq 0$ is allowed, then it becomes a Poisson equation. If unknown function ψ would have more variables, more terms will be involved (quadratic growth).

The mentioned problem is the most general and complicated case, so in this technical report, we will limit ourselves only to the following equation:

$$A\Delta\psi + B\nabla\psi + C\psi = Df \tag{4.4}$$

Second derivatives over multiple different variables are not allowed, and second and first-order derivative merging is not allowed, so each order is

multiplied with a single constant (A and B). The expression is also called *mixed partial derivative.*

The ratio between second-order and first-order terms are specified by Péclet number [89], which is defined related to 4.4 as:

$$Pe = \frac{B}{A} \Rightarrow \Delta\psi + Pe\nabla\psi + \frac{C}{A}\psi = \frac{D}{A}f \tag{4.5}$$

In other words, it is a rate between advection and diffusion.

4.2 Boundary condition

A certain number of boundary conditions must be known to solve the differential equation problem. The problem can be illustrated in a simple case. Suppose the equation $\psi' = 0$ on domain $\langle 0, 1\rangle$. The formulation tells the derivation is zero, so the single-parameter function is constant – the problem is to figure out the constant term. Because there would be an infinite number of solutions, a single boundary condition at a single x is required to reduce the degree of freedom. If $\psi(0.3) = 1.6$ is defined, for example, the function is defined as $\psi(x) = 1.6 \quad \forall x \in \langle 0, 1\rangle$.

Similarly, suppose the equation $\psi''(x) = 0$. The equation states that second derivation is zero; in other words, the function is linear. To define a linear function, two boundary conditions have to be known, and also one of these two options:

1. $x_1, x_2 \in D_\psi, \quad x_1 \neq x_2 : \psi(x_1) = a \wedge \psi(x_2) = b$
2. $x_1, x_2 \in D_\psi, \quad x_1 \neq x_2 : \psi'(x_1) = a \wedge \psi(x_2) = b$

The first option means that two values for two different points are known. Second the single function value and one first derivative value are known. It is not enough to know two first derivative values, as these two values have to be equal to the definition of the problem.

To simplify the explanation in the further text, let us call each boundary condition by its name. If the boundary condition restricts function value, it is called the Dirichlet (D) boundary condition. In the other case, if the first derivative is known, it is called Neumann (N) boundary condition. The naming is by their inventors – german mathematicians Johann Peter Gustav Lejeune Dirichlet (1805-1859) and Carl Gottfried Neumann (1832-1925).

So in the simple problem, either two Dirichlet or one Dirichlet and one Neumann boundary conditions must be defined. Reasons come from the line definition – determined by two points (D+D) or a point and a direction vector (D+N).

4.3 Weak formulation

Because solutions of PDEs can be quite complicated (e.g. (4.5) written in the strong form), the weak formulation takes its place. In weak formulation, each term by a test function v is multiplied and resulting product over the whole domain is integrated. So equation 4.5 will be rewritten in the weak form to:

$$\int_{\Omega} \Delta\psi v d\mathbf{x} + Pe \int_{\Omega} \nabla\psi v d\mathbf{x} + \frac{C}{A} \int_{\Omega} \psi v d\mathbf{x} = \frac{D}{A} \int_{\Omega} f v d\mathbf{x} \tag{4.6}$$

It is a tradeoff between the need for test function v and linearising the equation in some sense that one can solve approximately by linear approximation methods. The test function v describes the changes of the model which is allowed – for example, the movement of the effector in a deformation problem or heating node or group of nodes in a heat flow problem.

The main advantage of the weak form in the FEM is that the integral over a whole domain may be substituted with the sum of integrals over each element. The integrals through all elements are simpler to solve because they are just approximations (mainly linear functions) of the original problem. The single-dimensional problem will now be described.

4.4 One-dimensional problem

At first, the focus is on the elementary one-dimensional problem to better look at the problem itself. Assume a real problem modelled by equation 4.5 (4.6 in weak form). A real-life example that may be modelled this way is a heat transfer of fluid in the tube, where the fluid is dynamic. The tube is considered with an infinitely small radius or a laminar flow only, ensuring problem one-dimensionality. Because of that, the function ψ is a single variable function $\psi(x)$.

4.4.1 Discretization

As the finite element method comes in place, the considered space needs to be discretized into intervals small enough to reach the desired approximation error. There are many approaches to space discretization. Uniform sampling is very simple but may neglect important function features [21] (for example extrema points, inflex points, stationary points and more), regularity is also sometimes not satisfactory.

If the input data discretization is sparse or unsuitable for a given problem, the approximation and interpolation methods from chapter 3 will take place on the data as a preprocessing step. The interpolation is, however, also used as an integral part of the FEM. For further explanation purposes, the linear

interpolation method will be used[1]. However, let us point out that using a different estimation is no big issue. It is enough to extend the ideas described in further text.

In the following text, the discretized domain variables will be denoted as $x_1, x_2, \ldots x_n$, so one-based indexing of n domain nodes. Also, g_d notation is used for a discretized version of any function g.

Some basic functions have to be chosen to approximate function ψ, or in other words, a set of functions, which linear combination produces the discretized function ψ_d. There are many choices, e.g. *sin* and *cos* functions getting Fourier approximation (see section 3.4), radial basis functions (see section 3.5 or [21] [10]) family of functions or even more complex functions, but here the simple to explain (but also complex enough to work in practice) set of functions to explain are triangular functions.

4.4.2 Triangular basis

A triangular function is defined by three points a, b and c as follows:

$$\Lambda(x) = \begin{cases} \frac{a-x}{a-b}, & a < x \leq b \\ \frac{x-c}{b-c}, & b < x < c \\ 0, & otherwise \end{cases} \tag{4.7}$$

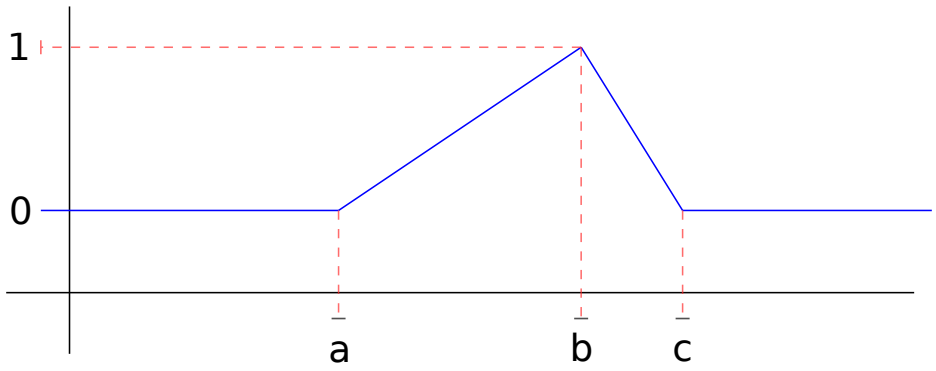

Figure 4.1: Triangular function $\Lambda(x)$ with parameters a, b and c.

and is also illustrated in Fig. 4.1. This function has compact support – the value is non-zero in a limited interval only. From the definition it can be seen that this function is piecewise linear (four linear segments) with $\Lambda(x < a) = \Lambda(x > c) = 0$ and $\Lambda(b) = 1$. The trick here is that a linear combination of multiple Λ functions (basis) with different a, b and c (this can be seen in Fig. 4.2) for each basis function can be done. Hence, obtaining a piecewise

[1]It reaches enough precision (better than the constant approximation) and is simple enough to explain to the reader.

linear function with more and more segments (depending on the number of basis functions) means approximating more complicated mathematical functions with lower approximation error. The result of the method is a set of weights of a linear combination of all functions (in this case, $\Lambda(x)$).

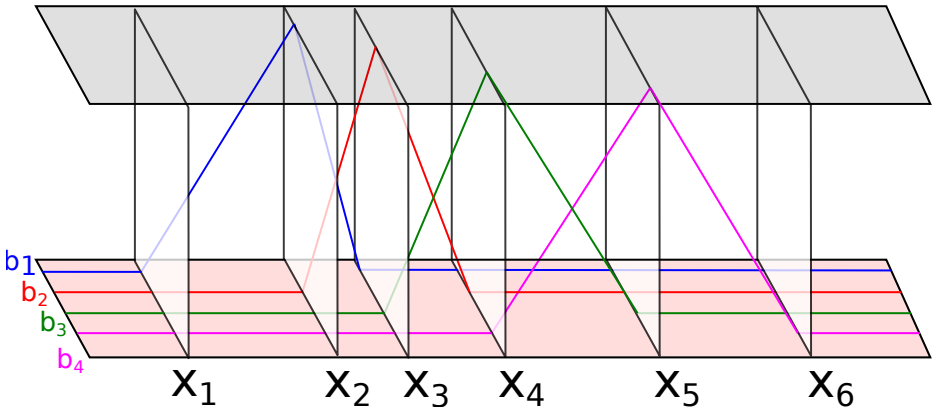

Figure 4.2: Triangular basis functions $b_1, \ldots, b_4$ on domain ψ_d with $n = 6$, values $\psi_d(x_1)$ and $\psi_d(x_6)$ are fixed by Dirichlet boundary conditions at zero. Idea adapted from [8].

4.5 Multidimensional triangular basis

As far as one-dimensional triangular basis were explored in section 4.4.2, the more complicated 2D triangular basis description will be described. To generalize the approach into higher dimensions, the transformation is similar to that from 1D to 2D.

4.5.1 2D triangular basis

Solving the problem (in any dimension) requires some kind of tesselation of the input space. The most common approach involving triangles is Delaunay triangulation; however, a general division into triangles (or simplices in higher dimensions) is also possible, as you can see in Fig. 4.3. The FEM also allows the usage of different shapes than simplexes; often, rectangles, cuboids etc., are used. These shapes require different elements (single functions). For the sake of simplicity, we stick in this report just with simplexes.

The elements will surely be more complex, as far as there are two dimensions there. By induction, the one-dimensional approach will be extended using the same idea to two dimensions. Each basis function b_i was piecewise linear, with value 1 at a specific point x_i and zero elsewhere on each sample. So the conditions for all basic functions are:

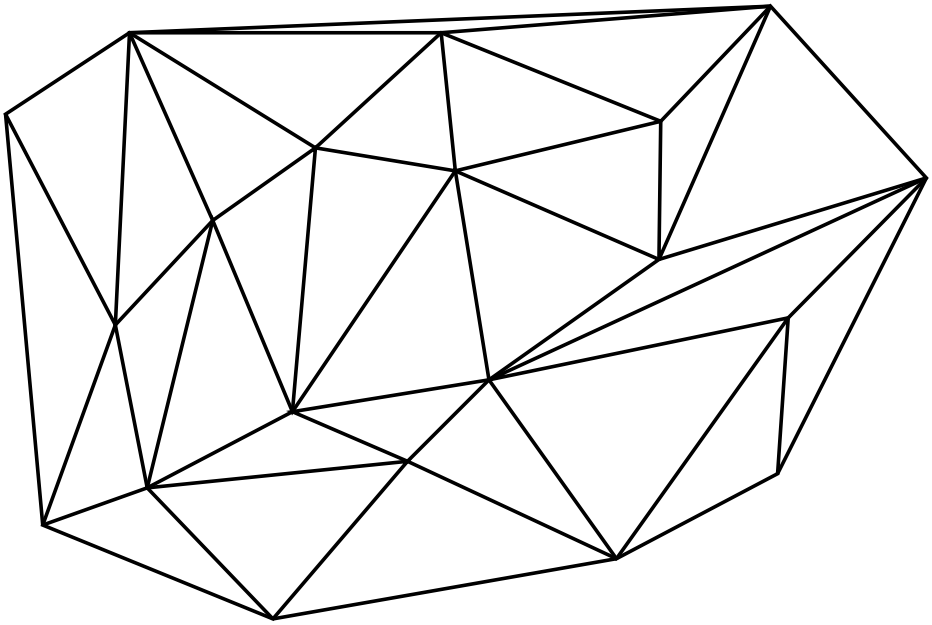

Figure 4.3: General 2D tesselation – triangulation.

$$\forall i, j \in \{1, \ldots, n\} \quad b_i(x_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{4.8}$$

In this sense, basis functions are defined in two dimensions with the same properties. To do so equation 4.8 is used and single scalar argument x_j is replaced by two parameters x_{jx}, x_{jy} (two cartesian coordinates x and y of the node vector $\mathbf{x}_j$). However, each b_i on a domain $\mathbb{R}^2$ has to be defined, not just for n discrete points and even with the piecewise linear property.

The trick to solving this problem is taking into account only adjacent triangles of the corresponding basis node and scaling the codomain of these triangles into $\langle 0, 1\rangle$ interval. Elsewhere the value will be zero. The function b_i will be obtained with compact support again.

The approach is illustrated in Figure 4.4. Value at node $[x_{j1}, x_{j2}]$ equals one (node surrounded by red color = 1). Then the value is linearly interpolated according to adjacent triangles (outlined by black colour) perpendicularly to opposite edges, where the value of the function b_j reaches zero (green colour). On the function domain outside the adjacent triangles, the function is also zero (green).

The last step is taking a linear combination of b_i to produce a two-dimensional interpolating triangular surface over the domain of ψ.

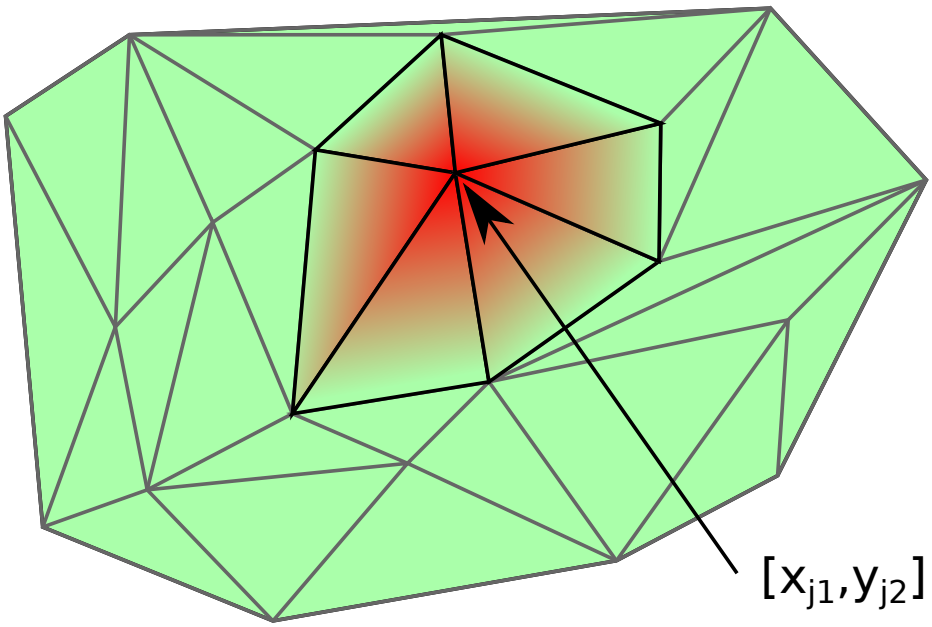

Figure 4.4: An illustration of single basis function b_j with compact support using heightmap (0–green, 1–red). The bold area has the value 0 on the outside vertices and one on the inside vertex.

4.6 Examples

Implementation in GNU Octave has been chosen for the finite element method mainly because of its rapid prototype development in combination with its GPL licence. At first, we will look at the one-dimensional problem because it is the most simple case to discuss.

4.6.1 One-dimensional problem

As a simple example, a heat distribution along a 1D approximation of a rod has been selected. There are two boundary conditions specified on both ends of the rod. The first end has a predefined temperature of 0 °C; the second end loses one degree Celsius every second (its derivative is -1). These conditions are mathematically described in Equ. (4.9). In the case of a single dimension, the unknown function ψ has only a single scalar parameter of x.

$$\psi(0) = 0 \qquad \psi'(1) = -1 \qquad \psi''(x) = 1 \tag{4.9}$$

The problem to solve is a second-order partial differential equation, so it is required to have two boundary conditions (D+D or D+N). The rod domain is $\langle 0, 1\rangle$, and the discretization is random using 100 samples (with explicitly defined 0 and 1 as a part of the sample set).

To check the result, the problem is also analytically solved. The two boundary conditions are known and also second derivative being one on the whole domain. It is also known that if the second derivative is constant, it produces a parabolic function in equation (4.10).

$$\psi(x) = ax^2 + bx + c \quad \Rightarrow \quad \psi'(x) = 2ax + b \quad \Rightarrow \quad \psi''(x) = 2a \tag{4.10}$$

From the formulation above, variable $a = \frac{1}{2}$, b can be computed from $-1 = 2ax + b = 2\frac{1}{2}1 + b \Rightarrow b = -2$ and c will be $0 = a*0+b*0+c \Rightarrow c = 0$. The parabola equation is hence $y = \psi(x) = \frac{1}{2}x^2 - 2x$ with roots 0 and 4 and center on $[2, -2]$. I can argue that part of the parabola in Fig. 4.5 may be a good approximation of a part of the parabola $y = \frac{1}{2}x^2 - 2x$ as far as there is no evidence against this hypothesis. The analytic function $y = \frac{1}{2}x^2 - 2x$ is also shown in Fig. 4.5 in red.

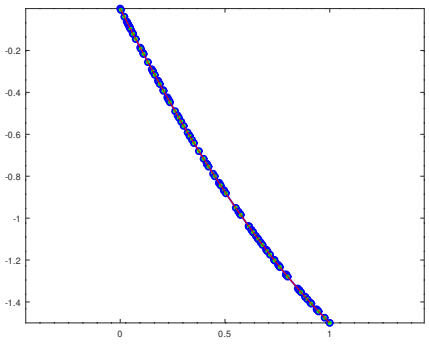

Figure 4.5: One-dimensional advection-diffusion problem with Dirichlet boundary condition $\psi(0) = 0$ and Neumann boundary condition $\psi'(1) = -1$. The blue curve is calculated using FEM, the analytical function in red.

Listings 4.1 shows the relevant Octave source code. The program executes with a single parameter, Péclet number Pe, but its default value is 1. At first, the count of samples on the domain and constants `dirichlet=1` and `neumann=2` are specified. The fourth line prepares the equation's right-hand side by heat sources applied to each element (linear segment). Splitting the domain to the defined number of nodes randomly with uniform distribution (to demonstrate the dependency of the solution on the discretization method) is the next step, but one can choose uniform sampling. After linear system allocation, each element is assembled into a global matrix `diff_op`, describing discretized first derivative operator. Each element is described via a small matrix containing derivatives of corresponding two non-zero basis functions at the element. The Pe value is also involved in the creation of each element.

When this is done, at the thirteenth line, boundary conditions are specified, Dirichlet $\psi(0) = 0$ (at node with index 1) and Neumann $\psi'(1) = -1$ (at

last node `count`). To set up these boundary conditions, iteration over all of them are performed, and it is also determined which condition is which type. In the case of the Neumann condition, the value from the right-hand side (this is fairly simple, that is why we can refer to the FEM-related literature about Neumann boundary condition as "natural") is subtracted.

On the other hand, much more for the Dirichlet condition (for every problem, we need at least one Dirichlet boundary condition, which is why some FEM-related literature called this type of condition "essential") has to be done. It is required to set the previously unknown value to the specified one (by setting the corresponding row and column to zero, leaving the value "one" on the diagonal. The other way around is just putting out the row and column from the equation) and correct the right-hand side of the equation (lines 19 and 22). Finally, the equation can be solved, and the result is plotted.

```
1  function fem1d(peclet=1)
2    count = 100;
3    [dirichlet,neumann] = (@()size([1,2]))();
4    sources = 1.*ones(1,count-1);
5    domain = [0 sort(rand(1,count-2)) 1];
6    right_side = (diff_op = zeros(count))(:,1);
7    for e=2:count
8      idel = peclet./(domain(e)-domain(e-1));
9      pos=[e-1,e];
10     diff_op(pos,pos)+=[-idel idel; idel -idel];
11     right_side(pos)+=sources(e-1)*.5.*[1;1]./idel;
12   endfor
13   conds = [dirichlet,1,0;neumann,count,-1];
14   for i=1:size(conds,1)
15     pos = conds(i,2);
16     if conds(i,1)==neumann
17       right_side(pos)-=conds(i,3);
18     elseif conds(i,1)==dirichlet
19       right_side-=conds(i,3)*diff_op(:,conds(i,2));
20       diff_op(:,pos)=diff_op(pos,:)=0;
21       diff_op(pos,pos)=1;
22       right_side(pos)=conds(i,3);
23     endif
24   endfor
25   sol=diff_op\right_side;
26   plot(domain,sol,'-ob','LineWidth',2,'MarkerFaceColor',
       'g','MarkerSize',7);
27   axis equal;
28 endfunction
```

Listing 4.1: One-dimensional problem solution

The result of the algorithm 4.1 (with $Pe = 1$ leaving default to make it less confusing) is shown in Fig. 4.5. Random sample distributions can be seen in the figure, fixed values at zero to zero and also first derivate on one set to -1, proving declared boundary conditions were satisfied.

So far, the results of FEM in one dimension have been shown, and results were also verified. It is time to test more complex problems involving more than one dimension.

4.6.2 Two-dimensional problem

A two-dimensional example may be a heat distribution problem on a 2D plate. Let us assume a plate where the borders are fixed on 0°C. The following example will show how it would end up when there is added heat to half of the plate. There are two examples, one on the square plane and one on the circular plane in Fig. 4.6 and Fig. 4.7, respectively. The square plane has domain $\langle 0, 1\rangle \times \langle 0, 1\rangle$ and consists of 400 uniformly (in 2D) sampled samples. The circular plane has a maximum radius of one, the distance from the centre is uniform from zero to one, and the angle is linear, with a golden ratio step (see equation (4.11)). For both the cases, the (artificial) problem is $26\psi'' + 2\psi + 0.05\psi = f$ (Péclet number $Pe = \frac{1}{13}$).

As the differential operator matrix is assembled for the one-dimensional problem, the problem is that it has to be done the same in two dimensions. And this is the purpose of the function `first_derivative` listed in 4.2. It takes two input parameters: `tri` matrix of $t \times 3$ elements (where t is the number of triangles) containing in each line three indices describing triangle structure; and also `points` matrix of $n \times 2$ elements, two coordinates in each line. The function produces a differential operator matrix `A`.

The code starts with preparing variables for the readability of the code. Variable `dim` will contain the dimension of the problem (the code is further also used in 3D), `n` is the dimension of the output square differential operator matrix `A`, allocated afterwards at line 4.

Then, while the iterations over all triangles are performed, the lengths of the triangle edges are calculated and then normalized. For each edge of the triangle, `dj` element is created and put into global matrix at positions `ds` and `dt`. The local matrix derives from the derivative of a single basis function.

```
function A = first_derivative(tri,points)
  dim = size(points,2);
  n = size(points,1)*dim;
  A = spalloc(n,n,dim*dim*3*size(tri,1));
  for i=1:size(tri,1)
    d=points(tri(i,:),:)-points(shift(tri(i,:),-1),:);
    d./=sqrt(sum(d.^2,2));
    for j=1:3
      dj = d(j,:)'*d(j,:);
      ds = (1:dim)+(tri(i,j)-1)*dim;
      dt = (1:dim)+(tri(i,mod(j,3)+1)-1)*dim;
      A(ds,dt)-=A(dt,ds)-(A(dt,ds)-=dj);
      A(ds,ds) += dj;
      A(dt,dt) += dj;
    endfor
  endfor
endfunction
```

Listing 4.2: Matrix with first derivative of basis functions

In two dimensions, there is a need for some point distribution there. The uniform distribution, Halton, random, and "spiral" distribution have been tested.

The listings 4.3 shows dedicated Octave code to a uniform distribution problem. The function accepts one single parameter (number of desired points, the requirement is that this is a square number) and outputs matrix `coords` of size $n \times 2$ containing x and y coordinates of each vector and also matrix `b` of size $2 \times n$ consists of boundary condition index and value.

```
function [coords,b] = uniform(n)
  sn = sqrt(n);
  [x,y] = meshgrid(linspace(0,1,sn));
  coords = [x(:) y(:)];
  b = [2:sn-1 1:sn:n-sn sn:sn:n-sn n-sn+1:n;...
         zeros(1,sn*4-4)];
endfunction
```

Listing 4.3: Uniform distribution with boundary conditions

To be able to solve the differential equation, one needs at least a single Dirichlet "essential" boundary condition. The code designated for it is at the listing 4.4. It accepts four parameters in total – `A` is differential operator matrix, `b` is right hand side vector of the equation, `dimension` is

self-explanatory (there it is 2, but it may be used for 3D as well) and `bnds` is a list of Dirichlet boundary conditions.

At first, the code iterates over all of the conditions. For each of them, it stores the value at the given node. Next, in the inner loop, the right-hand side and `A` matrix are modified by zeroing the corresponding row and column except the diagonal; there will be one value. Finally, both modified `A` and `b` are returned.

```
1  function [A,b] = dirichlet(A,b,dimension,bnds)
2    for i=1:size(bnds,2)
3      val = bnds(2,i);
4      for j=(dimension*bnds(1,i)-dimension+1):dimension*
          bnds(1,i)
5        b-=val*A(:,j);
6        A(:,j)=A(j,:)=0;
7        A(j,j)=1;
8        b(j)=val;
9      endfor
10   endfor
11 endfunction
```

Listing 4.4: Function which sets up Dirichlet boundary conditions

The code above can fix the particular node at a selected value. Domain nodes can also be discretized programmatically. The code to assemble the differential operator has already been shown, so it is time to put it all together. There is the main function which has a single input parameter in listings 4.5 – while non-zero, it loads the data from files. Otherwise, it generates them by uniform distribution already discussed and shown at listings 4.3. When the data are loaded or created, the differential operator is created, and the right-hand side of the equation is set to one for each node in which the y coordinate is above average. Else it is set to zero. The problem $26\psi'' + 2\psi + 0.05\psi = f$ (Péclet number $Pe = \frac{1}{13}$) is specified and Dirichlet conditions loaded/created are applied. In line 20, the linear system equation problem is solved, and the results are rearranged to fit correctly into visualization, which is done afterwards (depending on the data dimension).

```
function fem2d(load=1)
  if load~=0
    tri = csvread('tri.csv');
    points = csvread('points.csv');
    bnds = csvread('bounds.csv')';
  else
    bound = [-1 -1; 1 1];
    count = 20;
    sources = zeros(count.^2,1)+1;
    [points,bnds] = uniform(count.^2);
    points = points.*(bound(2,:)-bound(1,:))+bound(1,:);
    tri = delaunay(points);
  endif
  dim = size(points,2);
  A = first_derivative(tri,points);
  b = zeros(size(A,1),1);
  b+=repmat(points(:,2)'>mean(points,1)(2),dim,1)(:);
  A=(0.05*eye(size(A))+2*A+26*A*A);
  [A,b]=dirichlet(A,b,dim,bnds);
  x=sqrt(sum(reshape((A\b),size(points,2),size(points,1)
     )'.^2,2));
  if dim==3
    trisurf(tri,points(:,1),points(:,2),points(:,3),-x);
    axis equal off;
  elseif dim==2
    trisurf(tri,points(:,1),points(:,2),-x);
  endif
endfunction
```

Listing 4.5: Main function

Now, the results of the implemented FEM method in two dimensions follow. At first, the result of the unmodified code is in Fig. 4.6. The part where y is above average ($y > 0$, right part of the image) is deformed much more because the right-hand side was 1 there (it is like a heat distribution problem with some heat sources on the right side of the image). The Dirichlet boundary conditions are set from the code – zeroes on each point on the domain boundary.

The program can be modified to use some different distribution. In the Fig. 4.7, the points were generated in a "spiral" kind of shape, mathematically speaking like this:

$$\begin{aligned} \mathbf{x}_i &= \kappa\left(\frac{i}{n}, i\varphi\right) \\ i &\in \{1, \ldots, n\}, \quad \varphi = \frac{1+\sqrt{5}}{2} \\ \kappa(r, \theta) &= [r\cos\theta, r\sin\theta] \end{aligned} \tag{4.11}$$

where vector function κ converts polar coordinates to cartesian ones, and φ is a golden ratio, providing good distribution in the circular area. As it can be seen, the choice of distribution may be significant. In the circular example, the more dense centre shows some phenomena which are not visible otherwise.

4.6.3 Surface model problems

The simplest example on a triangular surface mesh is performing a heat distribution over it.

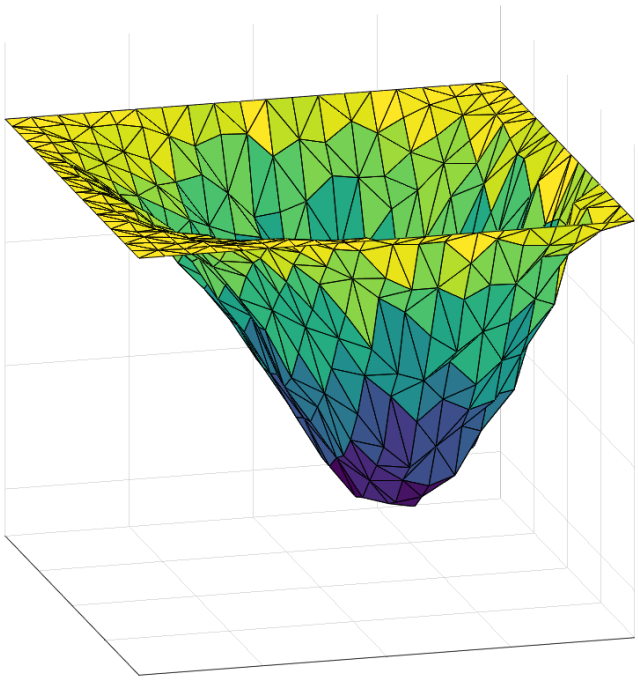

Figure 4.6: Tesselation of uniformly sampled 2D space with heat distribution.

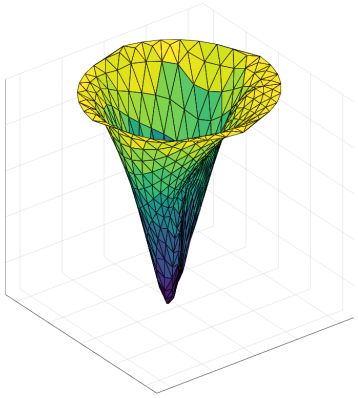

Figure 4.7: Tesselation of a spiral with heat distribution.

In this particular case, the data has been loaded instead of generated (using the parameter of the program 4.5 were non-zero) because the creation of a realistic surface model data automatically is a complicated task. For testing purposes, a model of man has been used, illustrated in Fig. 4.8. Head and body area ($y > \bar{y}$) have a right-hand side of the equation set to one. Also, the nose area is "anchored" by Dirichlet conditions on two nodes.

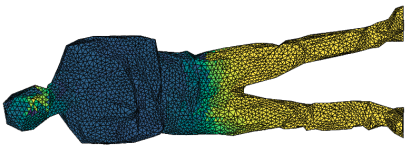

Figure 4.8: A heat distribution on a complex surface triangular mesh.

Application of the "essential" boundary conditions causes the values to transition around the nose and in the belt area, where the right-hand side of the equation drops from one to zero (from right to left). The detail of diffusion phenomena is shown in Fig. 4.9.

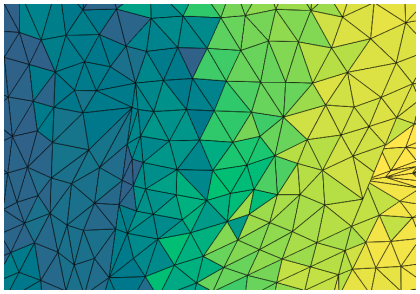

Figure 4.9: Diffusion phenomena in the belt area (detail of Fig. 4.8).

There is the opportunity to load a different model or even tweak the parameters of the partial differential equation in the code described above to approximate the solution of various problems.

Chapter 5

Existing methods

There are many methods of performing modelling of human movement and muscle modelling. These methods can be classified by the dimension they operate. The first methods are methods in one dimension, modelling muscle by a line, curve or set of lines/curves.

An exhaustive review of state-of-the-art methods has been recently done by Dereshgi et al. [23] and also previously by Lee et al. [52]. Although they seem to briefly summarise many currently existing methods, only a citation is needed. This paper covers fewer methods in more depth. Moreover, the statement by Dereshgi et al., "Although mechanical properties of muscles such as force, power, and work are well known", [23] shows a considerable overestimation of the current state-of-the-art, considering the exhausting list of recently published papers about the topic.

Description of the basic ideas of muscle modelling and some of the most appropriate methods follow.

The main distinction of muscle modelling is between forward and inverse kinematics approaches. The forward kinematics simulates the electrical excitation of the muscle cell, cell response, fibre response and muscle response in total. The latter simulates the muscle motion the other way around. We need to know the "result" (bone motions in general) to do inverse kinematics. Furthermore, the goal is to figure out how the muscle has to change to produce the motion. The ultimate task (which is not addressed here) is to figure out the electrical excitation of all muscle units; hence, figure out all of the internal muscle force. All of this allows performing direct kinematics simulation of the given muscle. However, this report aims mainly to describe inverse kinematics methods; hence, further details are omitted. More details can be found in Ezati et al. [26].

For this purpose of modelling, the muscle model is further approximated with a limited set of lines or curves (approximating the real muscle fibres), with the same direction as it is on real muscle (because the real muscle is anisotropic).

5.1 Hill-type model

The main idea behind the Hill-type models is to approximate the real muscle fibre by the triplet of parallel, serial and contractile elements. It is the most basic and probably the oldest mathematical model of the muscle presented by Hill [37] in 1938. He experimented on frog muscles and described the muscle contraction dynamic (supported by experiments) by the equation:

$$(F + a)\, v = b\,(F_0 - F) \tag{5.1}$$

where F denotes the current muscle force, F_0 is the maximum strength of the muscle, a is in the units of force and depends mainly on the maximum strength of the muscle (thus, indirectly on muscle size, mainly on the cross-sectional area), meaning that $\frac{a}{F_0}$ is a constant, even disregarding the temperature of the muscle. Variable b is in velocity units and is a constant (assuming constant temperature). The constant will increase in higher temperatures, allowing quicker muscle contraction. It is also proportional to the muscle length l (the measurement parallel to the muscle fibre from one attachment area to the other, $\frac{b}{l}$ is a (nearly) constant). Variable v denotes the velocity of the shortening.

The equation (5.1) may be written also as:

$$(F + a)\,(v + b) = const. \tag{5.2}$$

It means that the load F and velocity v are inversely proportional, so a more significant load will be moved slower.

The main problem here is that the a and b variables are hard to obtain for each muscle in the human body [37]. The next problem of this model is that muscle shape varies from person to person. Another problem here is (due to contraction) the cross-sectional area (the most significant area parallel to every fibre in the muscle) and length changes during the muscle activity; thus, a and b also.

According to Hill, "when a muscle shortens, in a tetanically maintained contraction, it liberates extra energy in two forms, (i) as "shortening heat", in amount proportional to the shortening, and (ii) as external mechanical work. The shortening heat is independent of the load, and therefore of the work done and the shortening speed." [37]. This idea was to model muscle fibres using three elementary units, sufficient energy, shortening heat and external mechanical work, arranged in a series-parallel connection. The model is shown in Figure 5.1.

The Hill model [37] has been improved significantly. The main addition was parallel element addition [91] in 1989. The improvements are called "Hill-type" models. The next addition is also a viscous damping element [42].

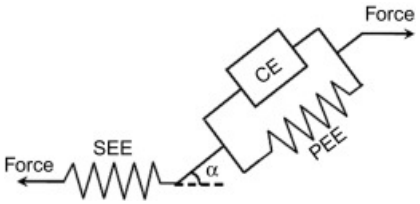

Figure 5.1: Hill-type model of a muscle fibre [2]. PEE = parallel element, SEE = serial element, CE = contractile element, α - pennate angle.

This Hill model has been proved to be insufficient to calculate internal muscle forces in some cases. According to Modenese et al., "from the mechanical point of view, this (Hill-type model, author's note) is a valid representation of a three-dimensional muscle only as long as the line segments pass through the centroids of the force distribution in the considered muscle sections" [61], which proved to not to be the case very often. Moreover, Martins et al. show that there are some "medical applications with muscle structures that are not reducible to 1D (like the muscles of the pelvic floor [...] and the diaphragm that separates thorax and abdomen)" [58]. Valente et al. [83] added that the muscle modelling using only a single line segment could produce errors of muscle force estimation up to 75% (while modelling *gluteus minimus* muscle). The solution is to model the muscle fibre using more complex curves.

There are still some applications of the pure Hill-type models [11]. However, the issue is to find the model variables a, b and pennate angle α. That is why methods do not model musculoskeletal model using Hill-type fibres nowadays, but instead incorporate the Hill-type fibre model idea into different approaches (PBD, FEM ...), trying to find the variable values or at least mimic the Hill-type model behaviour using fewer parameters.

5.2 Via-points

A via-points approach works on a predefined set of points. This set of points defines where the muscle fibre model should go through. When performing inverse kinematics movement, the goal is to reduce the number of intersections, mainly the muscle model with the closest adjacent bone model.

There are many options on how to define these points. The most common ones are points directly fixed to a bone, so whenever the bone moves, the point moves accordingly. The second option is that the point is present only if a condition is met (for example, if the joint flexion angle is more than x), so the natural shape of the fibre model is partially restored. The third option is a point which may move depending on some state (for example, depending on some angle, typically between two bones), following a predefined curve (see Fig. 5.2. These three variants are used in OpenSim modelling software).

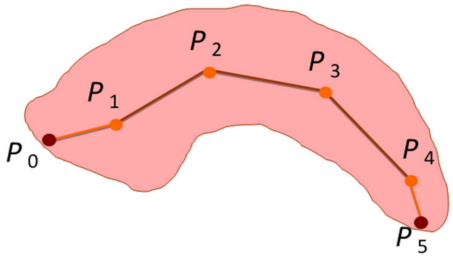

Figure 5.2: Via-points control curve inside a muscle [39]

There are, however, some catches. The first obvious one lies in the via-points definition. There has to be an approach to define these points because the user-defined points will be time-consuming for the physician and costly and subjective. The second main problem is that the fibre models may intersect other models or even self-intersect no matter what. Furthermore, the resulting muscle fibre model will not be smooth in some cases. Modenese stated that "straight-lines representation of muscles surrounding the hip joint was limiting the accuracy of hip contact force predictions" [61]. The better approach maybe not go through a set of points but "wrap around" a predefined set of geometric objects [30].

5.3 Wrapping obstacles

The wrapping obstacles approach has been developed to address some of the abovementioned problems. Some of the problems are still present:

- The geometric objects must be specified in the same fashion as the via-points.
- In the case of wrong curve selections, the intersection problem stays.

The improvement is in the smoothness of the curve. A curve which wraps around a sphere will be smooth even in the worst case, which cannot be said about the via-points approach. The wrapping around a cylinder is illustrated in Figure 5.3.

The main issue here is that infinitely many curves wrap around a volumetric object. From these curves, we can select the shortest one, the one with the least average curvature, maximum curvature etc. The selection depends mainly on the application (on the required precision, computing power etc.). The issue is even more present in some extreme bone arrangements.

The wrapping obstacles approach has been used (concerning muscle models) by, e.g. Lloyd et al. [56] and Kohout et al. [45]. Lloyd et al. wrap around an arbitrary geometrical model, subdividing the curve into segments divided

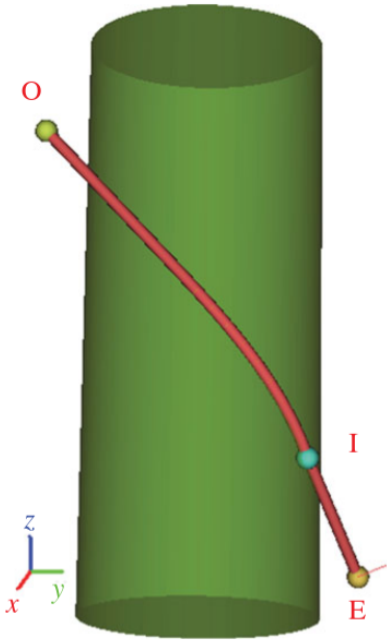

Figure 5.3: Wrapping obstacles approach. A line of action is wrapped around a single cylinder [45].

by knots with elastic forces to draw the knots together and keep them from penetrating obstacles. The latter (and significantly older) approach is limited to the sphere, single-cylinder and sphere-capped cylinder.

5.4 Finite element method

The finite element (FE) method has already been described in detail in chapter 4; however, its application for the muscle modelling problem has been omitted until now. For some reference, the most relevant approaches are described in chronological order.

Although multiple FE methods were published dealing with soft tissue modelling, the first paper on musculoskeletal modelling using the FE method is probably from Martins et al. [59] published in 1998. They incorporated the Hill-type model into their FE model (however, the same coauthor in 2006 showed that the Hill-type model might be insufficient in some cases [58]). They divided brachialis muscle into 4050 tetrahedra and assumed constant material constant (not considering muscle anisotropy). The external force has been applied just to an arbitrary part of the muscle ("right end").

Delp and Blemker [22] in 2005 used a template, which is then projected onto the target mesh. The projected template is then deformed using the finite element method. They consider the resolution of magnetic resonation

(MR) imaging of that time for template creation. To find out the boundary condition, they identified the tendon region from MR (the complete process is not described in detail, but I assume that some manual effort is necessary because of the phrasing "using [...] and knowledge of anatomy"). As a basis function for the FEM, they use Bernstein Basis Functions, which have been proven to approximate in the limit (in the sense of Weierstrass theorem) any continuous function defined on a finite interval.

Boubacker et al. [9] did a considerable amount of work to publish a survey on this topic and serves the purpose of a good problem introduction. However, it is worth noting that it was published in 2006. Therefore, some information there is obsolete.

Oberhofer et al. [66] in 2009 used cubic Hermite interpolation function in their FE approach to ensure C^1 smoothness of the model. Moreover, the boundary condition in attachment areas is used with the via-point approach (see section 5.2) to ensure no muscle penetrates any bone. These via-points are then integrated into the FE itself. The objective function consists of a distance between landmarks and targets and a Sobolev smoothing constraint.

Kaze et al. [24] in 2017 used FE to divide the model using tetrahedra. The boundary condition has been derived from muscles using the anatomic muscle attachment area. For the sake of simulating a tendon, a mass-spring-like system has been adopted (further details below in section 5.5.1). Their work's main focus was to estimate maximal strain.

Wei et al. [88] in 2019 also used FE to model a human hand. They used Nolan model [64] for hyperelastic soft tissue modelling (the main purpose was to model the human skin accurately). Their work mainly focused on analysing pressure generated by different hand grips.

Currently, there are many methods involving the pure finite element method. Fougeron et al. [28] currently use FE for above-knee socket load analysis. Using FE, Vila Pouca et al. [85] study muscle fatigue on a pelvic floor. Sun et al. [81] models spine movement also using the FE method.

Although the FE methods may produce great results, the main problem with FE methods is that they are difficult to set up because they require too many parameters [73]. Also, a disadvantage of the FE methods is that they are often hard to calculate. Fourgeron et al. [28] claim that their approach runs on 2 CPU machines for 40 minutes, which is far from real-time simulation. Due to this fact, the FE methods are introduced briefly, probably in less detail than they deserve. The main goal of this work to seek a faster method (real-time if possible) with comparable results.

There are multiple other methods currently available for muscle modelling (PBD, MSS), Moreover, some candidates will also be used for this purpose (ARAP). They promise comparable results using a smaller amount of computing power.

5.5 Other optimalization problems

The following methods work on the function minimisation principle. The optimisation is performed to find a geometric model (of muscle, muscle fibre) respecting some restrictions (e.g. volume preservation, shape preservation, fibre length etc.)

Some of the solutions go against each other. For example, if a muscle is stretched, the fibre length restriction wants to shorten the fibres back, but volume preservation wants to extend them to fill more space. Thus, these restrictions are often weighted according to modelling requirements. For all of the techniques, there are two common restrictions. The muscle is attached to a set of bones, so adjacent parts of the muscle must be fixed to the corresponding bone surface. Also, it is not appropriate for the muscle to be able to penetrate the bone. A collision detection and response (CD/R) form has to be implemented to avoid this issue. The following methods will describe what is optimised and how, but these two restrictions may be omitted because they are shared. Also, these restrictions are often threatened as strict (the second one may be a bit relaxed), meaning that there is no attachment displacement or collision should occur.

5.5.1 Mass-spring system

The mass-spring system (MSS) works as its name suggests, simulates complex motion by masses (point of masses) connected via virtual springs. The motion propagation is then done by transferring force through a series of springs. The spring is described using not such a complicated equation:

$$\mathbf{F}_{ij} = -k\mathbf{d}_{ij} \tag{5.3}$$

Where $\mathbf{F}_{ij}$ is the force produced by the spring between i-th and j-th particle, variable k denotes spring constant (force required to restore the spring to its original shape per unit of spring extension). The spring displacement is the variable $\mathbf{d}_{ij}$.

As far as the basic concept is not so complicated, there are many implementations of the approach. One of them is from Georgii and Westermann [32], who implemented this problem effectively on GPU.

There is a work from Aubel and Thalmann [3], using 1D mass-spring system model for each muscle fibre model independently, however, with some problems (mainly holes between the muscles). They also introduce angular springs, which hold the angle between two spring segments. Main problem with their work is their an almost lax attitute to collision detection: "Attempting to handle all muscle-muscle and musclebone collisions is unreasonable. In our framework, preventing muscle-bone collisions is easy and fast using repulsive force fields" [3]

The most considerable work is from Janak [40], the other work that combined mass-spring system and muscle modelling. He also considers the muscle fibre models as a whole and does not threaten them independently. Contrary to Aubel and Thalman, Janak's work works on muscle fibre models in continuous space. He decomposed the triangular mesh muscle model into a muscle fibre model, and these fibres were sampled uniformly to obtain nodes (masses) which he finally connected with edges (springs). His results show that his approach may work partially, but the problem is also his collision detection approach, where he approximated the surface mesh with spheres, meaning "the collision response is not perfectly precise and therefore the surfaces of the objects might partly intersect" [40].

The uniform sampling and connection between the mass points are illustrated on Figure 5.4. It should be noted that the author also considered randomisation and its consequences.

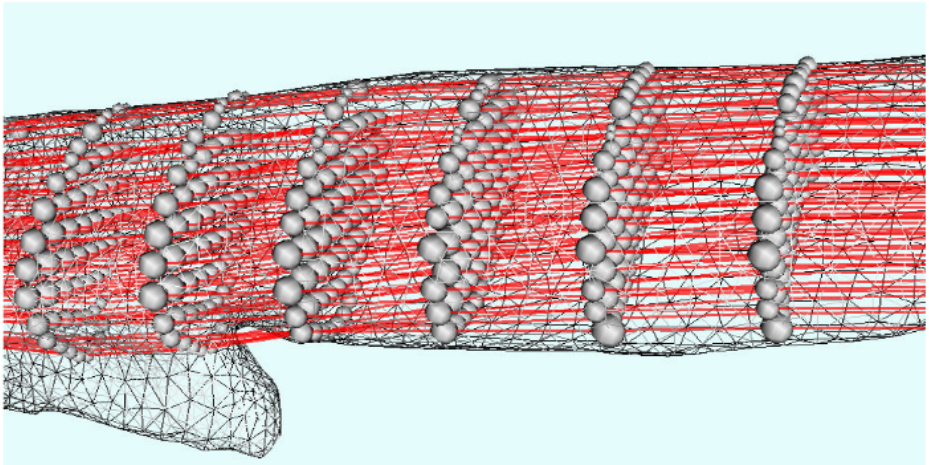

Figure 5.4: Mass-spring system, simulating a closer non-specified human muscle. The problem is with a part below, which is not approximated whatsoever. Image is taken from [41].

Optimization

If the springs are placed correctly, it is sufficient to optimise for spring length (respecting two common restrictions). In this case, we solve for:

$$m_i \mathbf{x}_i'' + c\mathbf{x}_i' + \sum_{\forall j \in \mathbf{N}_i} \mathbf{F}_{ij} = \mathbf{F}_i^e \tag{5.4}$$

where m_i is i-th particle mass, $\mathbf{x}_i$ is its position, c is a damping coefficient, $\mathbf{N}$ is a neighbourhood of i-th node, $\mathbf{F}_{ij}$ is a force of the spring, and $\mathbf{F}_i^e$ is an external force vector. The damping coefficient c can be calculated using the formula [41]:

$$c = 2\sqrt{2m\left(2 + \frac{4}{k_a}\right)} \tag{5.5}$$

where k_a is average stiffness. The exact solution can be obtained using integration over time. For computational purposes, it is sufficient to approximate the solution using finite differences, with a short time step dt. The result is as follows:

$$\mathbf{x}_i\left(t + dt\right) = \frac{\mathbf{F}_i^T\left(t\right)}{m_i} dt^2 + 2\mathbf{x}_i\left(t\right) - \mathbf{x}_i\left(t - dt\right) \tag{5.6}$$

$$\mathbf{F}_i^T\left(t\right) = \mathbf{F}_i^e - \sum_{\forall j \in \mathbf{N}_i} \mathbf{F}_{ij} - c\frac{\mathbf{x}_i\left(t\right) - \mathbf{x}_i\left(t - dt\right)}{dt} \tag{5.7}$$

This approach's main problem is its setup (variable k may vary between springs). This approach also does not preserve the original volume of the model, which is one of the desired outputs for muscle modelling. This could be probably solved using the approach described by Hong et al. [38], however, it would increase computational time dramatically.

5.5.2 ARAP

ARAP [79] stands for As-Rigid-As-Possible and is a method of finding minimal non-rigid transforms in the surface mesh. Its main contribution is that no internal structure is needed (for comparison, see e.g. [44], which is a very similar method to ARAP in some ways and includes volume constraints. However, it requires an internal structure "skeleton" to be defined). ARAP approach was used for medical purposes by Fasser et al. [27], where they used ARAP to morph the template of pelvic bone onto the subject-specific landmarks. However, they neglect some important features of the bone, for example, its volume, which the original ARAP cannot preserve. Wang et al. [87] also explored the ARAP approach. However, they discarded using the ARAP because "none of the methods worked well. [...] these methods produce non-smooth shapes with spikes (ARAP, BBW, FEM)" [87]. Moreover, their proposed approach also neglects the volume preservation requirement.

As-Rigid-As-Possible deformation should deform the model as little as possible. Any non-rigid transform is penalised via a cost function. The problem is mathematically formulated as finding the solution to a nonhomogenous system of linear equations. The matrix is a discrete Laplace operator of the mesh (after applying all boundary conditions, mainly fixed points), and the right-side vector contains second differences of each vertex concerning the close neighbourhood.

Laplace operator

The Laplace operator Δ is a second-order differential operator defined also as a divergence of the gradient. This leads to the assumption that the operand has to be a twice-differentiable function. The mathematical description is as follows (assuming x_i is a single cartesian coordinate of a vector produced by function f):

$$\Delta f = \nabla^2 f = \nabla \cdot \nabla f = \sum_{i=1}^{N} \frac{\partial^2 f}{\partial x_i^2} \tag{5.8}$$

The primary motivation behind Laplace operator usage is its significant local shape changes. If a Laplace operator is used on a triangular mesh, the definition in continuous space has to be modified to the discrete space. This step is done when optimal translation and rotation are computed (see the optimisation steps in 5.5.2 below). The discrete version of the Laplace operator over a manifold is also called the Laplace-Beltrami operator.

Volume preservation

The original ARAP approach is a fast and precise algorithm. However, it does not consider the preservation of the initial model volume. In the case of muscle modelling, this is crucial since muscle does not change its volume significantly while contracting.

To produce the correct volume, one can scale the whole model by scalar value so the volume will be restored. It can be done by factor[1]:

$$\sqrt[3]{\frac{V_0}{V}} \tag{5.9}$$

The simple idea is to multiply all coordinates by the initial and current volume ratio. Because there are three coordinates in 3D, it has to be cube rooted.

Although this is a simple and elegant solution, it also works in the case of concave shape, we cannot use it in the case where the muscle is attached to multiple bones in general at particular coordinates because these points are fixed and cannot be moved, so some volume will be missing/extra.

There is also a formula from Aubel & Thalman [4] which describes the scale factor as the square root of muscle elongation, but according to the authors, "We experimentally measured for various muscle shapes a maximal volume variation of 6% when the muscles shorten by 30%". The naive solution (used by the PBD algorithm, though) shows more accurate results [18] than Aubel's and Thalman's formula.

Seylan et al. [74] use ARAP for shape deformation with volume preservation; their approach for volume preservation was adding more edges, which

[1] This factor has no source because I derived it myself.

improves the result. However, according to them, the volume loss still occurs. After our experimenting in 2020 with the volume preservation, Dvorak et al. [25] used ARAP and volume tracking together for the time-varying surfaces, admittedly overperforming my preliminary experiments significantly (see section 6.2). Since their approach works well for the noiseless data (confirming our knowledge with problematic noise data) and its purpose is for general mesh, some ideas may be adopted from they work in our future work into muscle modelling techniques. The main issue there, however, is that they do not consider collisions between objects. There is also an issue where they require to deduce the internal structure of the surface mesh, so whenever they work with the internal structure, the algorithm's complexity rises from 2D to 3D, hence adding a degree of complexity and slowing down the modelling process. With that said, there are still some specific problems to solve (e.g. muscle anisotropy, collision detection and response, execution speed).

Optimization

In the ARAP approach, the optimisation is composed of two aspects, the optimal translation and rotation of a point in a given neighbourhood. The goal of ARAP is to minimise the energy in Equ. (5.10) [79]. A variable labelled with an apostrophe ($'$) belongs to the deformed mesh. Otherwise, it belongs to the original mesh.

$$E\left(S'\right) = \sum_{i=1}^{n} w_i \sum_{j \in \mathcal{N}(i)} E \left|\left|\left(\mathbf{p'}_i - \mathbf{p'}_j\right) - \mathbf{R}_i \left(\mathbf{p}_i - \mathbf{p}_j\right)\right|\right| \tag{5.10}$$

In the energy equation, E is the resulting energy, S is the triangular mesh on which the energy is calculated, and w_i is the weight of a connection between each vertex pair. The weights can calculate using the cotangent formula (described in Equ. 5.13), Tutte formula, Kirchhoff formula or anything else suitable for the given problem. Variable n describes the total number of vertices in the mesh, $\mathcal{N}\left(i\right)$ denotes the neighbourhood of the vertex i (all vertices which are connected via an edge). The position of the i-th vertex is labelled as $\mathbf{p}_i$, and the rigid rotation in the i-th cell denotes the variable $\mathbf{R}_i$.

The optimal translation and rotation of each vertex in the surface mesh in ARAP can also be described as follows. Let us start with optimal translation:

- Find centroid $\overline{\mathbf{c}}$ of original vertices
- Find centroid $\overline{\mathbf{c}}'$ of transformed vertices
- Optimal translation = $\overline{\mathbf{c}}' - \overline{\mathbf{c}}$

The centroid is calculated as a weighted average coordinate of neighbouring vertices (all vertices connected by an edge) using the weights in Equ. (5.13). The optimal rotation can be calculated as follows:

- Shift origins to centroids: $\mathbf{o'}_i = \mathbf{p'}_i - \bar{\mathbf{c}}'$ and $\mathbf{o}_i = \mathbf{p}_i - \bar{\mathbf{c}}$
- Stack $\mathbf{o}_i$ into matrix $\mathbf{P}$ $(3 \times N)$ and $\mathbf{o'}_i$ into matrix $\mathbf{P'}$ $(3 \times N)$
- Compute covariance matrix $\mathbf{C} = \mathbf{P}\mathbf{P'}^T$ (3×3)
- Compute Singular Value Decomposition (SVD) $\mathbf{C} = \mathbf{P}\mathbf{P'}^{\mathbf{T}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}$
- Optimal rotation $\mathbf{R}_i = (\mathbf{U}\mathbf{V})^T$

It has been shown [79] that it is sufficient to alternate between these two transformations until the error is reasonably small.

Linear equation system

When the local translation vectors and rotation matrices have been found. While optimal local rotations are independent of each other (a rotation in each cell can be calculated independently using SVD described above), the optimal translations are not independent (if a vertex moves, it affects all adjacent cells). Sokrine [79] derived that using the Laplace-Beltrami operator on the already deformed positions $\mathbf{p}$ (using local rotation $\mathbf{R}_i$) solves the problem of minimising the energy (see [79] for further details). The system will be further noted as $\mathbf{L}\mathbf{p'} = \mathbf{p}$ and each part of it will be further described.

An example of the linear equation system matrix $\mathbf{L}$ construction is in Fig. 5.5. The weight of each edge is there just "1" in the example for the sake of simplicity. The negative vertex degree is stored on each position on the main diagonal (sum of all weights in general). For the other cases, if there is an edge between the given pair of vertices, there is a value of 1 (edge weight in general), zero otherwise. Given definition assures that sum in each row and column stays zero. The nonzero elements become more complicated if different weights are introduced (e.g. cotangent weight in 5.13).

$$\mathbf{L} = \begin{pmatrix} -2 & 1 & 1 & 0 & 0 \\ 1 & -3 & 0 & 1 & 1 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -2 & 1 \\ 0 & 1 & 0 & 1 & -2 \end{pmatrix}$$

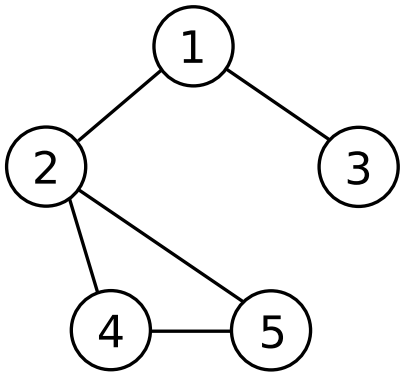

Figure 5.5: An example of ARAP linear equation system. The Laplace-Bertrami operator $\mathbf{L}$ on the right is deduced from the mesh connectivity on the left.

The right-hand side of the equation $\mathbf{b}$ is a bit more complicated to express than the matrix $\mathbf{L}$. Each element of the vector is equal to:

$$\mathbf{b}_i = \sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{2} \left(\mathbf{R}_i + \mathbf{R}_j\right) \left(\mathbf{p}_i - \mathbf{p}_j\right) \tag{5.11}$$

If some vertices are fixed, the number of dependent variables is reduced, thus the matrix becomes rectangular. To solve the system anyway, the rows of the matrix with the same indices as the fixed points can be simply ignored (removed), or the Ordinary Least Squares approach can be used to solve the recangular matrix system:

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} \\ \mathbf{A}^T\mathbf{A}\mathbf{x} &= \mathbf{A}^T\mathbf{b} \\ \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{A}\mathbf{x} &= \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} \\ \mathbf{x} &= \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} \end{aligned} \tag{5.12}$$

Edge weights

The weight of an edge may differ by the computing power requirements, accuracy requirements or usage in general. The simplest case is $w_{ij} = 1$ (called Kirchhoff), so all edges are considered the same. This approach works well for equilateral polygons, and its computing power requirements are negligible. The resulting matrix (see Fig. 5.5), however, produces a diagonally dominant matrix, which is often hard to solve computationally (because of number precision issues).

In order to solve the issue, each row can be divided by the value on the diagonal. This normalisation step would produce values -1 on the diagonal. The normalisation step may be the better approach for some numerical methods; also, it is better for compression purposes (diagonal does not need to be stored anymore). This approach is called Tutte.

These two approaches belong to a combinatorial Laplacian group, which does not consider anything else but mesh connectivity. There is also a cotangent formula, which considers the mesh's geometry. The weight is calculated as an average of the contangent of the angles opposite the given edge, ensuring that longer edges (with wider angles) would get lower weight than the shorter ones. Mathematically speaking, it is described as:

$$w_{ij} = \frac{1}{2}\left(\cot \alpha_{ij} + \cot \beta_{ij}\right) \tag{5.13}$$

where angles α_{ij} and β_{ij} are angles opposite to the edge between vertices i and j.

This approach has been successfully tested, and there was also an idea to extend the approach with volume preservation constraints. These results can be seen in section 6.2.

5.5.3 PBD

Position-based dynamics [62] is a fast approach used mainly in the animation industry to model elastic object (and cloth) deformations. Nowadays, it is making its way into physical simulations as well. The original algorithm does not consider the possibility of object anisotropy as far as the algorithm has been developed for general objects. The method accepts a manifold surface mesh and produces its deformed variant as input.

The PBD also exists in xPBD form (extended). It incorporates the concept of elastic potential energy and eliminates the need to know the time step and iteration count. For further detail, the reader may be interested in Macklin et al. [57].

Romeo et al. [73] is the first main contribution, which uses the PBD algorithm for muscle modelling problems. They mention the problem of using FEM or FVM: "Unfortunately, FEM and FVM, while providing excellent results, are known for not complying with the requirements [...]" which are "fast convergence of simulation, easy to set up, intuitive controls and artistic control" [73]. Their basic idea is to build an internal structure above the surface mesh to respect the anisotropy of the muscle (the internal structure respects the general direction of the muscle fibres). With an intelligent edge-creation process, they can create a volumetric model better suitable for the PBD[2] algorithm.

Angles et al. [1] developed a PBD-based approach for muscle modelling in 2019. Their approach virtually decomposes the muscle into "rods" (which may approximate the muscle fibres). These rods are allowed to change their diameter wherever to preserve their volume. Their main contribution is the capability to provide real-time simulation, which Romeo's approach is incapable of because "its $\approx$ 40s/frame of processing time makes it unsuitable to interactive applications" [1].

My contribution to using PBD for muscle modelling started with my master's thesis [17], in which the basic PBD approach was developed, supplemented by the anisotropy respect (finished the same year as Romeo's article [73], working concurrently on the same). After my master's thesis, the article "Fast and Realistic Approach to Virtual Muscle Deformation" [18] followed, where the approach has been extensively tested and integrated into an existing framework. The article "Muscle Deformation using Position Based Dynamics" [49] then further tests the approach and compares the results to an existing FEM approach. The main advantage of our proposed

[2]They use the XPBD - eXtended version of PBD, which respects the concept of elastic potential energy.

approach is that no interior is needed. Even the anisotropy is calculated on the surface of the mesh only, using muscle fibres on the mesh surface, defining the fibre direction. During my PhD study, also the implementation has been integrated into OpenSim, a well-known and widely used platform for modelling various physical phenomena. Currently, a publication abstract in collaboration with Havlicek is submitted, where we extend his ideas from his bachelor's thesis [35], improving collision detection and response approaches.

The output of the algorithm is a deformed surface triangular mesh. This serves the visualisation purposes well, but to calculate, e.g. force of the muscle, the muscle has to be transformed into a set of fibres. This process is called muscle decomposition and its details can be found in Kohout and Kukacka [48] or Kohout and Cholt [47] papers.

The pseudocode of the algorithm can be seen in Listings 1. The x_i variable is the position of each vertex, v_i is its velocity, Δt is the discretisation step (the smaller, the better if the accuracy is considered). Variable w_i is the inverse of the weight of each vertex, and p_i is a "working" position of each vertex (to ensure that the whole mesh's position will change coherently). The most complicated is the C_i variable, which describes a constraint, will be described further in the following text.

Mathematical background

The basic mathematic idea behind the PBD algorithm is the iterative minimisation of cost function C. Definitions of each C will be described further in the text. The cost function is a n-variable scalar function. The goal is to minimise the function value. Minimisation is done by gradient descent. A gradient of C is therefore required to express. If we neglect varying mass from the equation (we can do it as far as the muscles are assumed homogenous), the change of point position $\Delta\mathbf{p}_i$ is defined by equation 5.14.

$$\Delta\mathbf{p}_i = -\frac{\nabla_{p_i} C\left(\mathbf{p}_1, \ldots, \mathbf{p}_n\right)}{\sum_j \left|\nabla_{p_j} C\left(\mathbf{p}_1, \ldots, \mathbf{p}_n\right)\right|^2} \cdot C\left(\mathbf{p}_1, \ldots, \mathbf{p}_n\right) \tag{5.14}$$

Given equation can be understood as simply moving the point in the the opposite direction of the gradient (numerator) with the magnitude of the cost function value (expression after fraction). It is required to normalise the fraction by the sum of gradients to obtain directions somewhere between $\langle 0; 1\rangle$ as the sum of every normalized vector magnitudes should equal one.

The simple sum of $\Delta\mathbf{p}_i$ through all of the cost functions will become a weighted sum if we take into account point mass. The weights will be the inversion of mass. The movement will be opposite to the gradient direction (thus the minus sign) – gradient descent is performed.

Because the problem consists of multiple cost functions, the problem becomes a nonlinear equation system problem. This system will be solved by

Algorithm 1 PBD algorithm [17].

for all vertices i **do**
    initialize $\mathbf{x}_i = \mathbf{x}_i^0$, $\mathbf{v}_i = \mathbf{v}_i^0$, $w_i = \frac{1}{m_i}$.
end for
loop
    for all vertices i **do**
        $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$
    end for
    dampVelocities($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
    for all vertices i **do**
        $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
    end for
    for all vertices **do**
        generateCollisionConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
    end for
    loop solverIterations **times**
        projectConstraints($C_1, \ldots, C_{M+M_{coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N$)
    end loop
    for all vertices i **do**
        $\mathbf{v}_i \leftarrow \frac{\mathbf{p}_i - \mathbf{x}_i}{\Delta t}$
        $\mathbf{x}_i \leftarrow \mathbf{p}_i$
    end for
    velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
end loop

the Gauss-Seidel method, ignoring the nonlinearity of the equations. Each equation will be solved independently, propagating results between equations. It ensures massive speedup and, unfortunately, as shown afterwards, some acceptable errors.

Point distance cost function

The essential is to maintain initial distances between each point in the mesh.

Maintenance is done in a very similar fashion as it is in *mass-spring* [41] system. Cost is zero if and only if the distance between pair of points is the same as at the beginning of the simulation. If it is longer or shorter, the cost rises.

Equation 5.15 describes the distance constraint [62]. Function C is the scalar cost function of two variables (points), and its value describes the deviation from the initial distance. Vector variables $\mathbf{p}_1$ and $\mathbf{p}_2$ are temporary point positions (see Listings 1), value d is the original distance between given pair of points.

$$C\left(\mathbf{p}_1, \mathbf{p}_2\right) = \left|\mathbf{p}_1 - \mathbf{p}_2\right|_2 - d \tag{5.15}$$

In order to allow further calculation, it is necessary to determine the gradient of the cost function C with respect to $\mathbf{p}_1$ and also $\mathbf{p}_2$

Gradient of two vector difference norm First, the gradient of $L2$ norm from equation 5.15 has to be determined. This is done in 5.16 for the gradient respecting point $\mathbf{p}_1$, for $\mathbf{p}_2$ the derivation will be very similar.

$$\begin{aligned}
&\nabla_{\mathbf{p_1}} \left|\mathbf{p}_1 - \mathbf{p}_2\right|_2 = \nabla_{\mathbf{p_1}} \sqrt{\left(p_{1x} - p_{2x}\right)^2 + \left(p_{1y} - p_{2y}\right)^2 + \left(p_{1z} - p_{2z}\right)^2} \\
&= \frac{1}{2\left|\mathbf{p}_1 - \mathbf{p}_2\right|_2} \nabla_{\mathbf{p_1}} \left(\left(p_{1x} - p_{2x}\right)^2 + \left(p_{1y} - p_{2y}\right)^2 + \left(p_{1z} - p_{2z}\right)^2\right) \\
&= \frac{\left[\frac{\partial(p_{1x}-p_{2x})^2}{\partial p_{1x}} \quad \frac{\partial(p_{1y}-p_{2y})^2}{\partial p_{1y}} \quad \frac{\partial(p_{1z}-p_{2z})^2}{\partial p_{1z}}\right]}{2\left|\mathbf{p}_1 - \mathbf{p}_2\right|_2} \\
&= \frac{2\left[\frac{\partial(p_{1x}-p_{2x})}{\partial p_{1x}} \quad \frac{\partial(p_{1y}-p_{2y})}{\partial p_{1y}} \quad \frac{\partial(p_{1z}-p_{2z})}{\partial p_{1z}}\right]}{2\left|\mathbf{p}_1 - \mathbf{p}_2\right|_2} = \frac{\mathbf{p}_1 - \mathbf{p}_2}{\left|\mathbf{p}_1 - \mathbf{p}_2\right|_2}
\end{aligned} \tag{5.16}$$

Resulting gradient of the cost function C is (using previous derivation 5.16) shown in 5.17.

$$\begin{aligned}
\nabla_{\mathbf{p}_1} C\left(\mathbf{p}_1, \mathbf{p}_2\right) &= \nabla_{\mathbf{p}_1}\left(\left|\mathbf{p}_1 - \mathbf{p}_2\right|_2 - d\right) = \frac{\mathbf{p}_1 - \mathbf{p}_2}{\left|\mathbf{p}_1 - \mathbf{p}_2\right|_2} = \mathbf{u} \\
\nabla_{\mathbf{p}_2} C\left(\mathbf{p}_1, \mathbf{p}_2\right) &= \nabla_{\mathbf{p}_2}\left(\left|\mathbf{p}_1 - \mathbf{p}_2\right|_2 - d\right) = \frac{\mathbf{p}_2 - \mathbf{p}_1}{\left|\mathbf{p}_1 - \mathbf{p}_2\right|_2} = -\mathbf{u}
\end{aligned} \tag{5.17}$$

These cost function derivations are just direction vectors of the edges formed by the involved points. By common sense, it has to be this way to change the distance between points. It is logical to move along the directional vector. The result is further illustrated on 5.6.

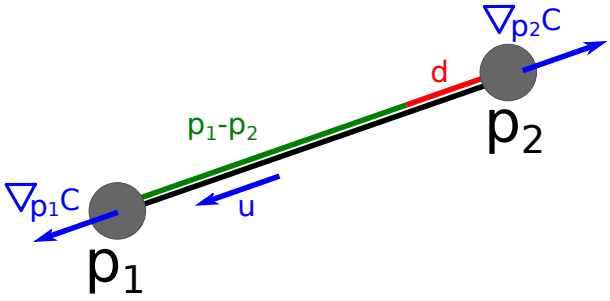

Figure 5.6: Cost of the distance change. The gradient is highlighted in blue.

Volume preservation

It is also a required feature of the muscle to preserve its initial volume. In the case of the musculoskeletal system and muscles especially, this requirement is reasonable because the muscles do not change their volume due to constant density.

At first, we must figure out how to calculate the volume of triangular, closed and manifold mesh. It may seem like a complex problem, as mesh may have different shapes (even more complex for higher genera). Luckily, the problem is elegant and straightforward to solve. At first, an arbitrary point in the space has to be selected. For the sake of simplicity, often $(0, 0, 0)$ is selected. Sometimes, the centre of gravity is selected instead, which may be numerically more stable. Then, for each triangle, we calculate the tetrahedra volume involving the triangle connected to our arbitrary point. The tetrahedra volume is just a sixth of the volume of the parallelogram formed by the three vectors originated in the arbitrary centre point and ending in each vertex of the triangle, as it is written in 5.18 with an arbitrary point at the origin of the coordinate system. Sometimes the volume may end up negative, which is also desired phenomenon because it allows subtracting concave parts or parts forming a higher genus of the mesh.

$$V_{tetra} = \frac{1}{6}\begin{vmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{vmatrix} = \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) \tag{5.18}$$

Vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are edges of the tetrahedra sharing a common vertex. In this case, the origin is used, simplifying the calculation so we can use the vertexes $\mathbf{p}$ of the triangle mesh instead.

An example is visualized on image 5.7. Cross product of vectors $\mathbf{a}$ and $\mathbf{b}$ gives vector $\mathbf{d}$ (in pink) has length the same as the area of the yellow region. Then, the dot product of this vector with $\mathbf{c}$ vector, we obtain the volume of the whole parallelogram. This procedure is also called *triple product*. A sixth of the volume of the obtained parallelogram is the desired tetrahedra.

We can now also determine the cost function of the volume change. The equation is shown on 5.19.

$$C\left(\mathbf{p}_1, \ldots, \mathbf{p}_n\right) = \sum_{i=1}^{m} \left(\mathbf{p}_{t_1^i} \cdot \left(\mathbf{p}_{t_2^i} \times \mathbf{p}_{t_3^i}\right)\right) - kV_0 \tag{5.19}$$

In the equation, m is the total triangle count, $\mathbf{p}_{t_1^i}$ denotes the first vertex of i-th triangle in the mesh. Variable V_0 represents the initial volume of the mesh (so the cost function value will be zero if it ever becomes the same). An optional parameter k allows us to change the volume over time, and we leave it for now at a value of 1.

Derivative of the cost function can be done either from 5.19 or from determinant sum in 5.18. At first, choose a vertex of the mesh to determine

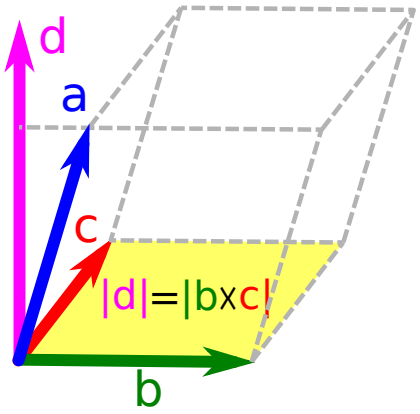

Figure 5.7: Volume of the tetrahedra formed by vectors **a**, **b** and **c**.

the derivative. A single triangle was selected for simplicity, where the derivate will be derived. The resulting total gradient is obtained using the sum rule of derivatives and is shown on 5.20.

$$\nabla_{\mathbf{p}_i} C(\mathbf{p}_1, \ldots, \mathbf{p}_n) = \begin{vmatrix} p_{ix} & p_{iy} & p_{iz} \\ p_{jx} & p_{jy} & p_{jz} \\ p_{kx} & p_{ky} & p_{kz} \end{vmatrix} + \cdots = \mathbf{p}_j \times \mathbf{p}_k + \ldots \tag{5.20}$$

The equation implies equality between the derivative in a vertex triangle and the cross product of the other two points. If we put all triangles together, the derivative of a vertex equals the cross product of each pair of vertices sharing the triangle with the selected vertex. Mathematically speaking, like in 5.21.

$$\nabla_{\mathbf{p}_i} C(\mathbf{p}_1, \ldots, \mathbf{p}_n) = \sum_{h=1}^{t} \mathbf{p}_j \times \mathbf{p}_k; \quad i \neq j \neq k \tag{5.21}$$

Variable t denotes a total number of triangles that share the vertex with index i, vertices with indexes i, j and k lie on the same triangle. The control variable h we do not need in this particular case.

In his work, Janak [41] did not consider volume preservation, which may enhance his results further. However, math between volume preservation in the mass-spring system is highly complitaced [38].

Shape preservation

The last restriction is to preserve the initial shape of the muscle. One of the possibilities is to preserve the dihedral angle between each pair of adjacent triangles as much as possible.

It is also necessary to describe how the dihedral angle between two adjacent triangles is calculated. For vertices are given: $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{p}_4$, Vertices

with indices one and two are shared, whereas index three belongs to the first triangle only. Moreover, index four belongs to the second one. Using the cross product, we obtain triangle normals like in equation 5.22.

$$\begin{aligned} \mathbf{u} &= (\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1) \\ \mathbf{n} &= \frac{\mathbf{u}}{|\mathbf{u}|_2} \end{aligned} \tag{5.22}$$

Here the $\mathbf{u}$ vector is declared to simplify the following derivation. The vector has the same direction as the normal but may not have the length of one in general. Vertices $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ belongs to first triangle, in which the normal is calculated. The second triangle will be calculated similarly.

Thus, the angle between two adjacent triangles can be determined. At first, both normals ($\mathbf{n}_1$ and $\mathbf{n}_2$) are required. Then, the angle between the two normals is obtained from the definition of the dot product. Definition and derivation are shown on equation 5.23.

$$\begin{aligned} \cos\varphi &= \frac{\mathbf{n}_1 \cdot \mathbf{n}_2}{|\mathbf{n}_1|_2 \, |\mathbf{n}_2|_2} = \mathbf{n}_1 \cdot \mathbf{n}_2 \\ \varphi &= \arccos(\mathbf{n}_1 \cdot \mathbf{n}_2) \end{aligned} \tag{5.23}$$

The next step is a cost function definition. The same approach will be applied with volume and distance cost functions, subtracting the initial and current angle values. It is defined in 5.24.

$$\begin{aligned} C(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) &= \varphi - \varphi_0 \\ &= \arccos(\mathbf{n}_1 \cdot \mathbf{n}_2) - \varphi_0 \\ &= \arccos\left(\frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)}{|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)|_2} \cdot \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 - \mathbf{p}_1)}{|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 - \mathbf{p}_1)|_2} \right) - \varphi_0 \end{aligned} \tag{5.24}$$

In the equation, φ_0 denotes inital angle (calcualted according to equation 5.23).

The derivative is a bit more complicated in this case. At first, we use the property that the cost function value is translation-invariant, so the pair of the triangles can be moved so one of the vertices will lie on the coordinate system origin.

If we translate each point with the vector $-\mathbf{p}_1$, the first vertex would lie on the origin, and the calculation of the normal vector of both triangles simplifies enormously. By translating the whole system, the dihedral angle will not change.

It is also written in equation 5.25, vectors with apostrophes are the ones translated by $-\mathbf{p}_1$.

$$\begin{aligned}\mathbf{n}_1 &= \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)}{|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)|_2} = \frac{\mathbf{p'}_2 \times \mathbf{p'}_3}{|\mathbf{p'}_2 \times \mathbf{p'}_3|_2} \Leftrightarrow \mathbf{p'}_1 = 0 \\ \mathbf{n}_2 &= \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 - \mathbf{p}_1)}{|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 - \mathbf{p}_1)|_2} = \frac{\mathbf{p'}_2 \times \mathbf{p'}_4}{|\mathbf{p'}_2 \times \mathbf{p'}_4|_2} \Leftrightarrow \mathbf{p'}_1 = 0\end{aligned} \tag{5.25}$$

Finally, with the knowledge of the derivative of the function arccos (defined as $\frac{d}{dx} \arccos x = -\frac{1}{\sqrt{1-x^2}}$), the gradients for each cost functions are obtainable (via equation 5.26). Because we choose $\mathbf{p}_1$ to be zero, it will be defined differently than the second shared vertex $\mathbf{p}_1$. The $\mathbf{p}_3$ and $\mathbf{p}_4$ will be very similar due to symmetry. The gradient does not depend on the absolute location of the object, so we do not need to do any backwards translation.

$$\begin{aligned} d =& \mathbf{n}_1 \cdot \mathbf{n}_2 \\ \bigtriangledown_{\mathbf{p'}_4} C =& -\frac{1}{\sqrt{1-d^2}} \left(\bigtriangledown_{\mathbf{p'}_4} (\mathbf{n}_2) \cdot \mathbf{n}_1 \right) \\ \bigtriangledown_{\mathbf{p'}_3} C =& -\frac{1}{\sqrt{1-d^2}} \left(\bigtriangledown_{\mathbf{p'}_3} (\mathbf{n}_1) \cdot \mathbf{n}_2 \right) \\ \bigtriangledown_{\mathbf{p'}_2} C =& -\frac{1}{\sqrt{1-d^2}} \left(\bigtriangledown_{\mathbf{p'}_2} (\mathbf{n}_1) \cdot \mathbf{n}_2 + \bigtriangledown_{\mathbf{p'}_2} (\mathbf{n}_2) \cdot \mathbf{n}_1 \right) \\ \bigtriangledown_{\mathbf{p'}_1} C =& -\sum_{i=2}^{4} \bigtriangledown_{\mathbf{p'}_i} C \end{aligned} \tag{5.26}$$

The basic idea behind determining the normal gradients of the triangles is decomposing the normals into cross products, which can also be written in matrix calculus. Each element of the matrix can then be solved independently. This is further explained on $\mathbf{n}_2$ normal at vertex $\mathbf{p'}_4$. The next step is to determine the partial derivative of the normalised cross product.

Normalized cross-product partial derivative Let me first substitute the cross product with vector $\mathbf{r}$ (as shown in 5.27) to simplify the further derivation process.

$$\mathbf{r} = \mathbf{p'}_2 \times \mathbf{p'}_4 \tag{5.27}$$

The next step is to calculate just the partial derivative of the cross-product (without normalisation, see 5.28). The result is a matrix $\mathbf{A}$, which, after multiplying with a vector, produces the cross product of given vector and $\mathbf{p'}_2$ vertex.

$$\begin{aligned}\bigtriangledown_{\mathbf{p}_4}\mathbf{r} &= \begin{bmatrix} \frac{\partial p'_{2y}p'_{4z}-p'_{4y}p'_{2z}}{\partial p'_{4x}} & \frac{\partial p'_{2y}p'_{4z}-p'_{4y}p'_{2z}}{\partial p'_{4y}} & \frac{\partial p'_{2y}p'_{4z}-p'_{4y}p'_{2z}}{\partial p'_{4z}} \\ \frac{\partial p'_{4x}p'_{2z}-p'_{2x}p'_{4z}}{\partial p'_{4x}} & \frac{\partial p'_{4x}p'_{2z}-p'_{2x}p'_{4z}}{\partial p'_{4y}} & \frac{\partial p'_{4x}p'_{2z}-p'_{2x}p'_{4z}}{\partial p'_{4z}} \\ \frac{\partial p'_{2x}p'_{4y}-p'_{4x}p'_{2y}}{\partial p'_{4x}} & \frac{\partial p'_{2x}p'_{4y}-p'_{4x}p'_{2y}}{\partial p'_{4y}} & \frac{\partial p'_{2x}p'_{4y}-p'_{4x}p'_{2y}}{\partial p'_{4z}} \end{bmatrix} \\ &= \begin{bmatrix} 0 & -p'_{2z} & p'_{2y} \\ p'_{2z} & 0 & -p'_{2x} \\ -p'_{2y} & p'_{2x} & 0 \end{bmatrix} = \mathbf{A} \end{aligned} \tag{5.28}$$

Next, the norm is taken into account, as is shown on 5.29, result is the cross product again, but now with normal vector $\mathbf{n}_2$.

$$\begin{aligned}\bigtriangledown_{\mathbf{p'}_4} |\mathbf{r}|_2 &= \bigtriangledown_{\mathbf{p'}_4} \sqrt{r_x^2 + r_y^2 + r_z^2} = \frac{1}{2\,|\mathbf{r}|_2} \bigtriangledown_{\mathbf{p'}_4} \left(r_x^2 + r_y^2 + r_z^2\right) \\ &= \frac{1}{2\,|\mathbf{r}|_2} \begin{bmatrix} \frac{\partial r_y^2+r_z^2}{\partial p'_{4x}} & \frac{\partial r_x^2+r_z^2}{\partial p'_{4y}} & \frac{\partial r_x^2+r_y^2}{\partial p'_{4z}} \end{bmatrix} \\ &= \frac{1}{|\mathbf{r}|_2} \begin{bmatrix} r_y p'_{2z} - r_z p'_{2y} & -r_x p'_{2z} + r_z p'_{2x} & r_x p'_{2y} - r_y p'_{2x} \end{bmatrix} \\ &= \frac{\mathbf{r} \times \mathbf{p'}_2}{|\mathbf{r}|_2} = \frac{|\mathbf{r}|_2\, \mathbf{n}_2 \times \mathbf{p'}_2}{|\mathbf{r}|_2} = \mathbf{n}_2 \times \mathbf{p'}_2 = \mathbf{b} \end{aligned} \tag{5.29}$$

Last but not least it is necessary to join both partial derivative together (from 5.28 and 5.29). We apply the derivative division rule and get the result in 5.30.

$$\begin{aligned}\bigtriangledown_{\mathbf{p'}_4} \frac{\mathbf{r}}{|\mathbf{r}|_2} &= \frac{\mathbf{A}\,|\mathbf{r}|_2 - \mathbf{r}\cdot\mathbf{b}}{|\mathbf{r}|_2^2} = \frac{1}{|\mathbf{r}|_2} (\mathbf{A} - \mathbf{n}_2 \cdot \mathbf{b}) \\ &= \frac{1}{|\mathbf{p'}_2 \times \mathbf{p'}_4|_2} \left(\begin{bmatrix} 0 & -p'_{2z} & p'_{2y} \\ p'_{2z} & 0 & -p'_{2x} \\ -p'_{2y} & p'_{2x} & 0 \end{bmatrix} - \mathbf{n}_2 \cdot (\mathbf{n}_2 \times \mathbf{p'}_2) \right) \end{aligned} \tag{5.30}$$

Similarly, the other partial derivative in 5.26 would be obtained, it will differ only in indices and signs. The result of the partial derivative is put into equation 5.26 determining the final partial derivative of the cost function. Following the same path as other restrictions, the movement will be according to 5.14 because we already know everything required to do so.

The whole idea is also illustrated on picture 5.8. For the sake of simplicity, only two triangles are shown. Normal vectors of both triangles are drawn in green ($\mathbf{n}_1$ and $\mathbf{n}_2$). The gradient in vertices $\mathbf{p}_3$ and $\mathbf{p}_4$ is shown for both vertices by two blue vectors forming a surface (in grey), where these vertices may move. Movement in the other direction would not change the angle

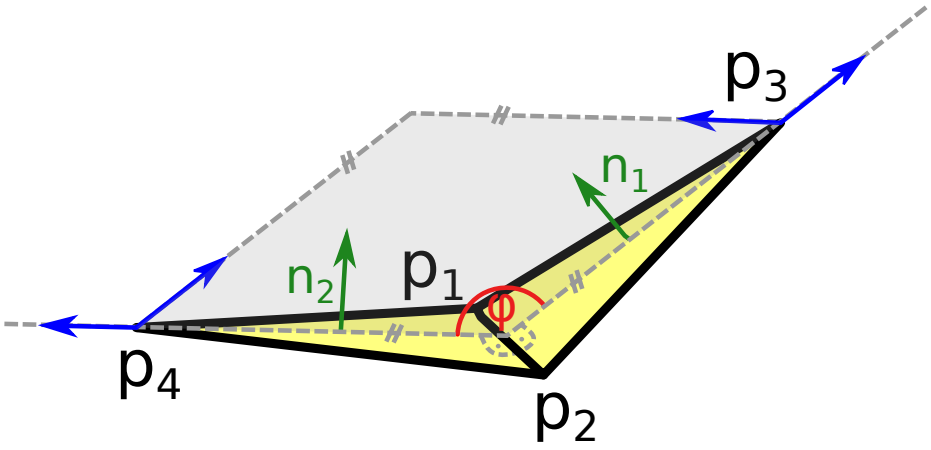

Figure 5.8: Angle φ preservation between two triangles (dihedral angle).

between the triangle pair. The middle point gradients were not illustrated due to complex $\mathbf{p}_1$ and $\mathbf{p}_2$ gradients.

In this chapter, seven considerable approaches to muscle modelling have been described: Hill-type model, Via-points, Wrapping obstacles, Finite element method, Mass-spring system, ARAP and PBD. The following table 5.1 provides a summary of these methods, with (partially subjective) speed and precision comparison.

Algorithm	Muscle model form	Speed	Precision	Reference
Hill-type	Single line	+ + +	− − −	[37] [91] [11]
Via-points	Polyline	++	−−	[39] [30]
Wrap. obst.	Complex curve	+	−	[45] [56]
FEM	Volumetric model	− − −	+ + +	[59] [9] [22]
MSS	Volumetric model	++	−	[32] [3] [40]
ARAP	Surface model	++	?	[79] [4] [25]
PBD	Surface model[3]	++	?	[62] [73] [49]

Table 5.1: Comparison of the muscle modelling approaches.

[3]With the exception in the paper from Romeo et al. [73], where they introduced volumetric data to improve their results.

Chapter 6

Experiments and Results

At this point, the theoretical aspects of muscle modelling have been exhausted, so the further direction of this text goes towards the practical experiments.

6.1 PBD results

The real data and an artificial dataset (described below) have been used to test the proposed approach described in section 5.5.3 and also in our papers [17, 18, 49]. The results of the experiments are described in detail below. In each iteration, a bone (or its artificial replacement) moves in a given direction. Also, in that particular iteration, there are some sub iterations (the user gives the number), where the bones do not move, but it is used for muscle movement stabilisation (reducing the Δt but also increasing the computational time). Unless stated otherwise, three sub-iterations have occurred in the following tests and each iteration step.

6.1.1 Collision detection and response

In order to simplify the problem, the decision to use a simple voxelisation approach has been made. The space has been subdivided into smaller cubes (the number of cubes depends on the user specification and model size), and to each box, information has been assigned whether it "contains" a part of the bone. The collision of muscle with bone has been approximated by detecting a triangle (from a muscle mesh) collision with the small box.

This simple idea, however, leads to some problems. Luckily, some ideas are already to entirely improve or even fix the problem, using more complex collision detection algorithms. Havlicek [35] changed the collision detection to DiscreGrid (using a signed distance field) and FLC (using a binary search tree), beating the voxelisation approach in terms of accuracy. However, there is still some work to do because even those methods do not work correctly in

the case of some extreme conditions, mainly if the movement is rapid (see[1] section 11.4, especially Fig. 11.13 in [35]).

The collision response is a complicated task as well. Assume that two bones move towards each other and narrowly miss each other (like shear blades). If a muscle is attached to both of the bones and appears to be in between the bones, there is no such room to go. This problem seems artificial but happens fairly often on a smaller scale, near joints especially, where two bones move close. My former solution [17] was to assume, in this particular case, only one of the bones and move a muscle in the direction opposite of it, but it shortly proved to be insufficient. Havlicek [35] targets this problem especially, and he proposed a better approach of considering all adjacent bones and moving opposite to the sum of all collision vectors. Even his approach, however, even his approach does not fix the problem entirely, as said before.

6.1.2 Muscle decomposition

The approaches described so far work primarily with the surface model, however, the last step of the pipeline in section 1.2 has to be performed for others to determine the muscle's internal forces (and other physical properties). It can be done using, for example, Kohout and Kukacka, CHMD, or VIPER muscle decomposition methods, which were proposed in [48], [47] and [1], respectively. In the following subsections, these methods are briefly introduced.

[1]The original paper is in the Czech language.

Kohout and Kukacka decomposition

The inputs of the Kohout & Kukacka decomposition method [48] are:

- triangular (and manifold) surface model of the muscle to decompose,
- fibre template, giving information about internal fibre arrangement,
- attachment areas to adjacent bones (origin and insertion), defined by a set of points lying on the adjacent bone surface models

Decomposition is then done as follows. Attachment areas are projected from the bone surface onto the muscle surface to define the muscle parts that are subsequently removed. Isocontours are computed using a piecewise linear scalar field on the modified muscle model. The scalar field has its maximum on the insertion boundary vertices, whereas its minimum is on the origin boundary vertices. Users can specify how many isocontours are generated in this process.

Similarly to [22], muscle fibre architecture (template) is represented by a unit 3D space with an arbitrary (user-defined) number of fibres inside the space. The fibres are represented analytically by Bezier spline curves. One of the predefined templates is selected according to the muscle being modelled (depends on if it has parallel or pennate fibres, optionally on a pennate angle, etc.), and it is mapped one-to-one on isocontours calculated in the previous step, forming the fibres going through the muscle model. Finally, noise is eliminated using quadratic smoothing to make the result more realistic and visually plausible.

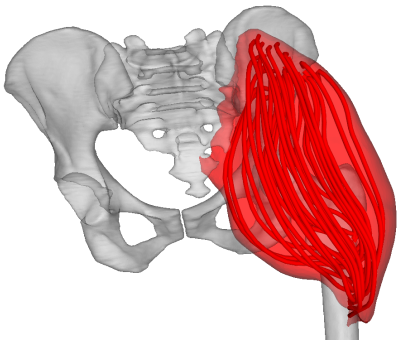

Figure 6.1: Gluteus maximus decomposed to individual fibres by Kohout & Kukacka's [48] algorithm.

CHMD decomposition

A technique by Kohout & Cholt [47] performs a centripetal Catmull-Rom interpolation of the input fibres lying on the surface model of the muscle or nearby to get the fibres inside the muscle. Their approach can even work with multiple-headed muscles, distributing the fibres automatically among the heads.

Compared to Kohout & Kukacka's proposed method, this method needs the specification of fibres on the surface, which typically requires some manual effort because these are not obtainable for the patient automatically. For the data to be valuable in practice, it must be adopted from a cadaveric dataset. On the other hand, it can work with multiple-headed muscles, whereas Kohout & Kukacka's approach can not. To avoid this problem, the authors used split multi-headed muscles into multiple biheaded, which partially works, but introduced a problem of multiple "submuscle" intersections and overlaps.

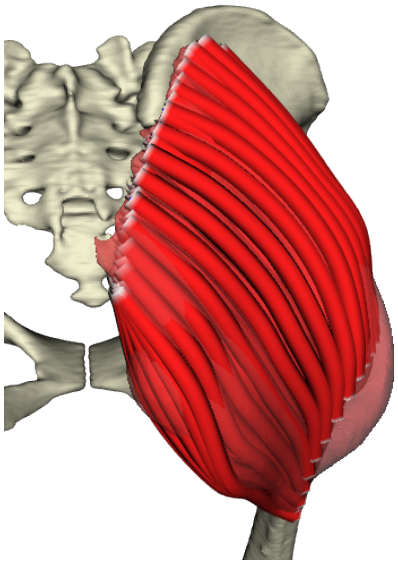

Figure 6.2: Gluteus maximus decomposed to fibres by Kohouts's & Cholt's [47] algorithm.

VIPER decomposition

Angles et al. [1] proposed in 2019 a complete muscle modelling approach, including their muscle decomposition technique. They decompose the muscle into a predefined number of "rods", using similar techniques as Kohout and Kukacka's approach above uses; however, they require 3D volumetric model instead of a 2D surface model as an input model. They also require the attachment area definition, which they specify manually on the muscle.

6.1.3 Artificial data

To test collision detection and behaviour in extreme cases, we prepared an artificial dataset where a box of 5292 triangles on its surface is squished between two plates (12 triangles each). The initial setup is visualized in Fig. 6.3. The goal of the test was to deform the muscle abnormally and test whether it regained its initial shape or not (in other words, if the internal forces will be capable of overcoming the external forces eventually).

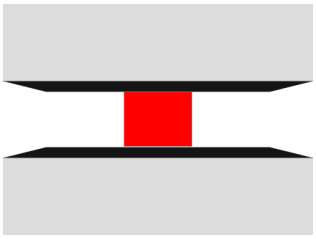

Figure 6.3: Input artificial data.

At first, in one hundred iterations, the top plate moves to decrease the space between both places. The distance between plates in 100^{th} iteration is 10% of the distance in the first iteration. Inverse movement is then performed between 100^{th} and 200^{th} iteration, returning the plates to their initial position. Additional one hundred iterations are used for box stabilisation.

As shown in Fig. 6.4, the box is squished quite a lot. Even though it returns to its original shape in 300^{th} iteration (only with slight rotation caused by asymmetrical triangulation). Even in 200^{th} iteration, the results are acceptable, except for one corner of the box.

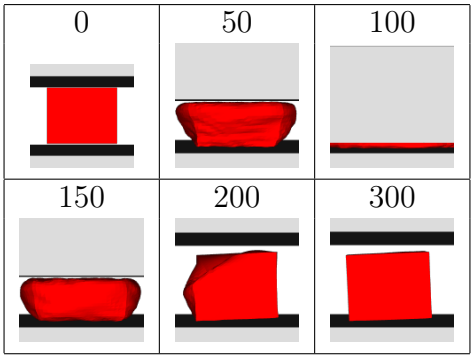

Figure 6.4: Results in different simulation frames (artificial data). The two black areas are just shadows from the grey plates, and no collision occurs.

The artificial muscle has been forced to squish, so the volume changed drastically during the simulation. The result is plotted in Fig. 6.5, the minimal volume was lower than 30% of the original volume of the artificial muscle.

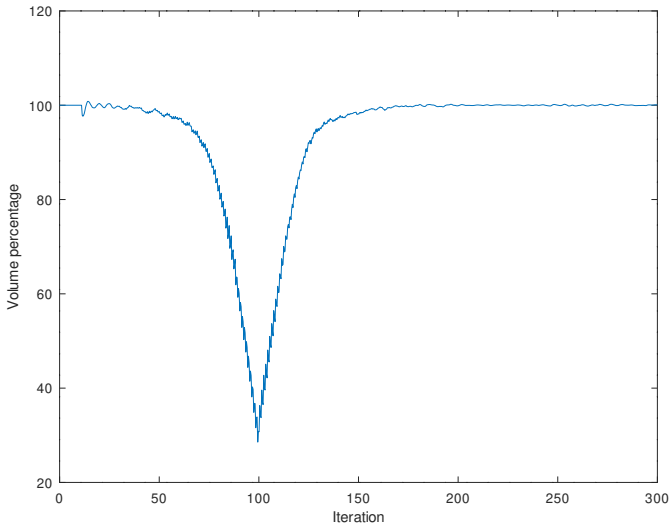

Figure 6.5: Volume change over 300 PBD iterations. The muscle was squished to less than 30% of its original volume during simulation.

6.1.4 Iliacus

We also tested the iliacus muscle, which connects the femur and pelvic bones from the front side. The bones and the muscle are visualised in Fig. 6.6. The muscle and bones are closed triangular meshes with 13858 triangles for muscle and 254442 (42456 for femur) triangles for all the bones.

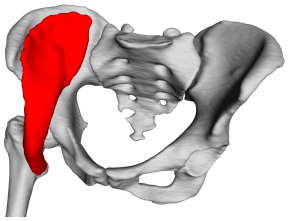

Figure 6.6: Input iliacus muscle and adjacent bones.

A similar simulation scenario like the artificial data case has been applied to the iliacus dataset. The first hundred iterations are used for hip flexion, in which the femur bone rotates one radian during this movement. The second hundred iteration is allocated for the inverse movement. The last hundred (200^{th}-300^{th}) iterations are there to stabilise the muscle movement.

The posterior part of the iliacus muscle is pushed into the joint during the flexion, as can be seen in Fig. 6.7. The explanation for this behaviour is as follows. The main problem is that vertices near the joint have been incorrectly fixed to the femur bone (see bullet 1 in the section 2.2). This problem leads to a situation when (due to flexion) the near part of the

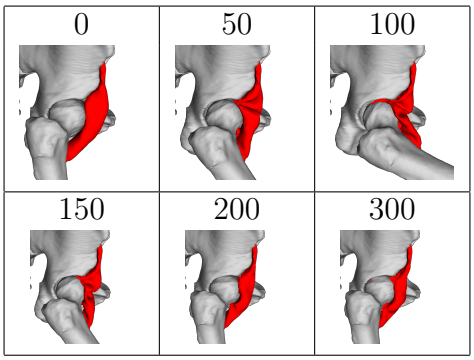

Figure 6.7: Results in different simulation frames (Iliacus muscle data).

muscle is pulled into the joint. Also, this part of the mesh is unrealistically arched towards the joint and, therefore, distance and local shape constraints tend to move the points of this part closer to the joint. As the femur and pelvic bones do not touch, there is a narrow space into which this part of the mesh can squeeze, making it difficult to get out. Even though the result is not visually plausible, the quantitative tests (described later) show that all critical factors of the muscle are preserved as much as possible.

6.1.5 Gluteus maximus

The Gluteus maximus muscle is attached to the same bones as the iliacus but from the other side. The triangular mesh consists of 19752 triangles. Fig. 6.8 shows how the model looks like.

Figure 6.8: Input gluteus maximus muscle and adjacent bones.

The muscle undergoes the same movement scenario as the iliacus mentioned above. Visualization in six important iterations is shown in Fig. 6.9. As we can see, the result in iteration 300 is nearly the same as in the first iteration (the original muscle pose).

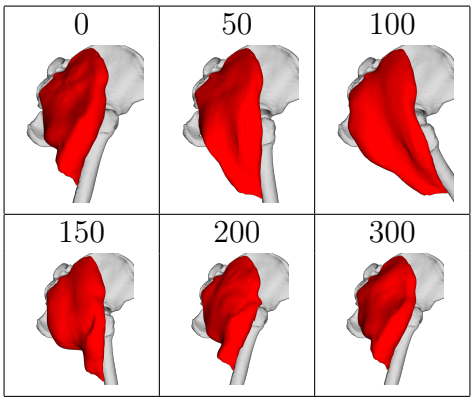

Figure 6.9: Results in different simulation frames (gluteus maximus muscle data).

6.1.6 Other muscles

The gluteus medius and adductor brevis muscles within this testing procedure have been tested. Both muscles deform quite realistically, as can be seen in Fig. 6.10 and Fig. 6.11, where the situation in the maximal hip flexion (same scenario as in the gluteus maximus and iliacus case) is shown.

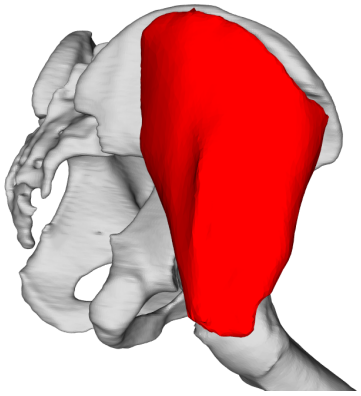

Figure 6.10: Gluteus medius in maximum flexion position.

6.1.7 Quantitative tests

To make some exact results, we use quantitative tests. These tell us how well constraints are satisfied during simulation.

The volume preservation constraint is tested by determining the ratio between original and actual volumes. Fig. 6.12 for artificial data, Fig. 6.13 and Fig. 6.14 for real data respectively, show us the volume preservation results.

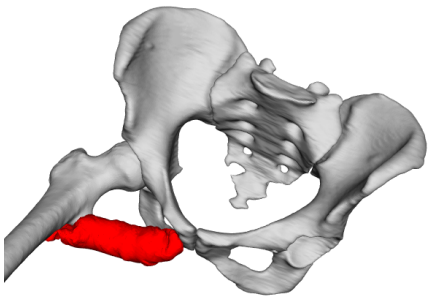

Figure 6.11: Adductor brevis in maximum flexion position.

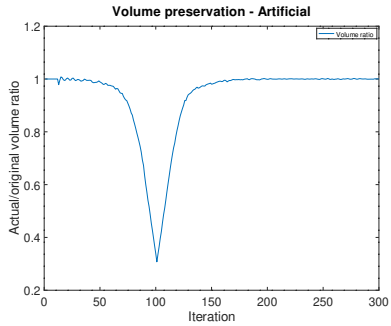

Figure 6.12: Volume preservation of artificial data.

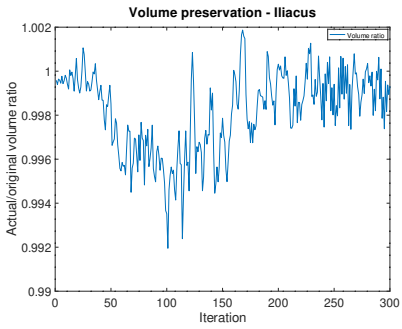

Figure 6.13: Volume preservation of iliacus muscle data.

As we can see from the results, the volume is well preserved in both real data (the error is less than 1% in both cases). A nice curve describes squishing (reducing) box volume during the simulation and restoring artificial data.

The next measurable property is average edge extension. At first, we cannot say much about artificial data (squished box) from the plot in Fig. 6.15. As for the iliacus muscle, some edges remain more prolonged than expected (see Fig. 6.16) because they are stuck in the hip joint. In the case of the gluteus maximus dataset (Fig. 6.17), the first 100 iterations show edge extension during hip flexion (it is correct behaviour because muscle extends

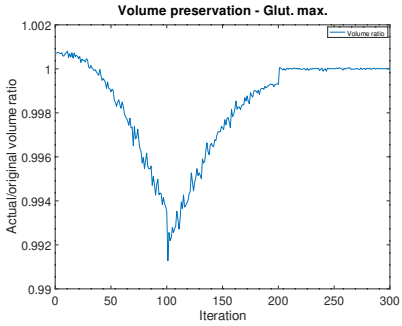

Figure 6.14: Volume preservation of gluteus maximus muscle data.

in this phase), and the second hundred iterations return the average length extension to almost 1 (i.e., the muscle returns to its original pose correctly). We can also see that the last 100 iterations are not crucial in this scenario.

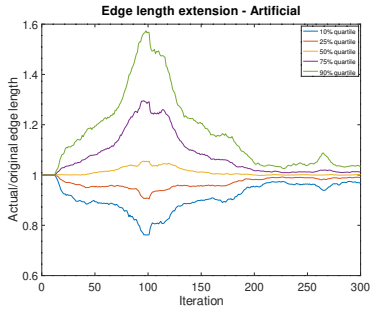

Figure 6.15: Average edge extension of artificial data.

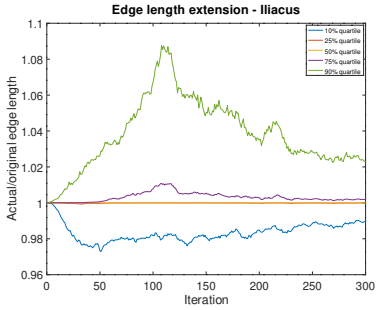

Figure 6.16: Average edge extension of iliacus muscle data.

We also tested how well the dihedral angles were preserved during the simulation. In this paper, the dihedral angle is the angle between two adjacent triangles in the muscle triangle mesh. According to plots in Fig. 6.18, Fig. 6.19 and Fig. 6.20, we can conclude that there are some pairs of triangles which do not preserve their original angle to the *acceptable*[2] extends, but most of them do.

[2]Depends on multiple factors, but the absolute distance deviation below 0.5cm is still

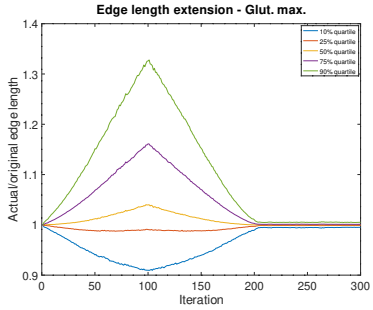

Figure 6.17: Average edge extension of gluteus maximus muscle data.

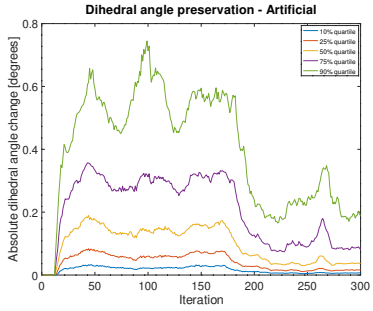

Figure 6.18: Average absolute dihedral angle change of artificial data.

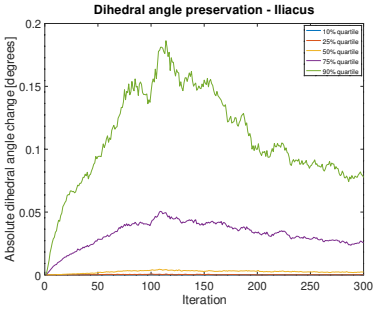

Figure 6.19: Average absolute dihedral angle change of iliacus muscle data.

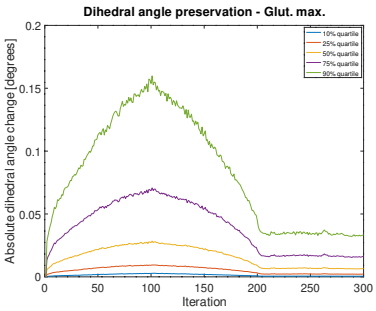

Figure 6.20: Average absolute dihedral angle change of gluteus maximus muscle data.

inside a measurable error interval in the worst case, luckily, the higher error is often acceptable.

6.1.8 Fibre length

Last but not least, the lengths of fibres were analysed. In the iliacus muscle case, as we can see in Fig. 6.21, many length curves exhibit two big bumps shortly after 100^{th} iteration. A part of the muscle stuck in the hip joint (as discussed previously) causes the problem. Nevertheless, when the bones return to their initial rest-pose (i.e., after 200 iterations), most fibres restore their original lengths quite well. Gluteus maximus muscle behaves as expected – see Fig. 6.22. During the flexion, all lengths increase; during the extension, they decrease.

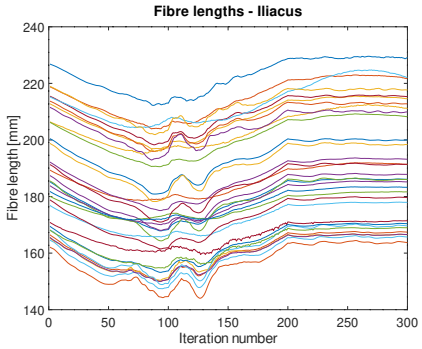

Figure 6.21: Total length of each fibre during simulation in iliacus muscle.

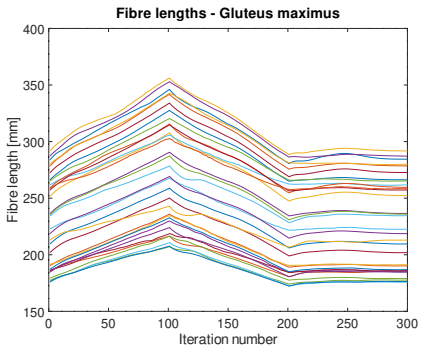

Figure 6.22: Total length of each fibre during simulation in gluteus maximus muscle.

6.1.9 Speed

The proposed method was designed to be mainly fast and precise. It was implemented in C++ using the VTK toolkit. Its current version is publicly available at `https://github.com/cervenkam/muscle-deformation-PBD`.

All testing scenarios above have been measured how fast each one was. FPS (Frames Per Second) is used as a speed metric in this case.

All tests were performed on Intel® Core™ i7-4930K 3.40GHz CPU, Radeon HD 8740 GPU and WDC WD40EURX-64WRWY0 4TB HDD. Results are listed on Tab.6.1.

Deforming object	Triangle count	FPS
Gluteus maximus	19752	33.85
Abductor brevis	17124	35.89
Iliacus	13858	47.21
Gluteus medius	10622	57.12
Artificial box	5292	153.61

Table 6.1: FPS of each simulation

The results show that FPS strictly depends on triangle count (Spearman's $\rho = -1$). The more triangles are used, the slower the method is.

Even though the program is primarily unoptimised and runs sequentially at the moment, the FPS is sufficient for considered purposes in general. For more details about the approach, see our paper [18].

6.2 Preliminary ARAP Results

In the experiments with ARAP, volume preservation has been added. There are results illustrated on the figures 6.23-6.26. Original models are shown in yellow. User deformation is then in grey. Two variants of linear equation assembly and two rotation tests were performed in the testing phase. The simple "interleaving" approach to preserve muscle volume was introduced in each test. In each step, the shape has been fixed by the original ARAP approach, and then the volume has been fixed using the formula (5.9).

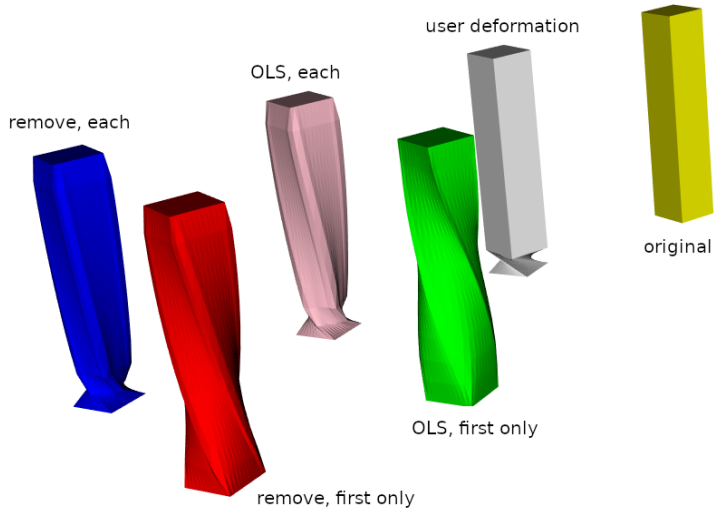

Figure 6.23: Results of four ARAP approaches with volume preservation, rotation of a box face around the Y axis.

The input model is a triangular mesh of a block. Its base has been discretised uniformly using a 20x20 grid, and 12 samples have been uniformly sampled at the height of the block. Finally, the sampled block has been triangulated.

The first option to assemble the linear system is to remove the rows and columns of the fixed points to get a regular square matrix (referred to as "remove"). The other option is to solve the rectangular system using ordinary least squares ("OLS"). A variant of the rotation approach is to rigidly rotate the object in each iteration ("each"). The other variant is to rotate it only once ("first only"), generating different results (probably acceptable for different applications).

OLS produces empirically better results. However, it has a higher computational cost in general. The reason behind the computational cost is the OLS, where (typically) sparse matrix has to be multiplied with its transposition (producing a more dense matrix) followed by matrix decomposition.

The methods produces satisfying result in the case of the artificial data (6.23-6.25), however, real data of the gluteus maximus muscle are problem-

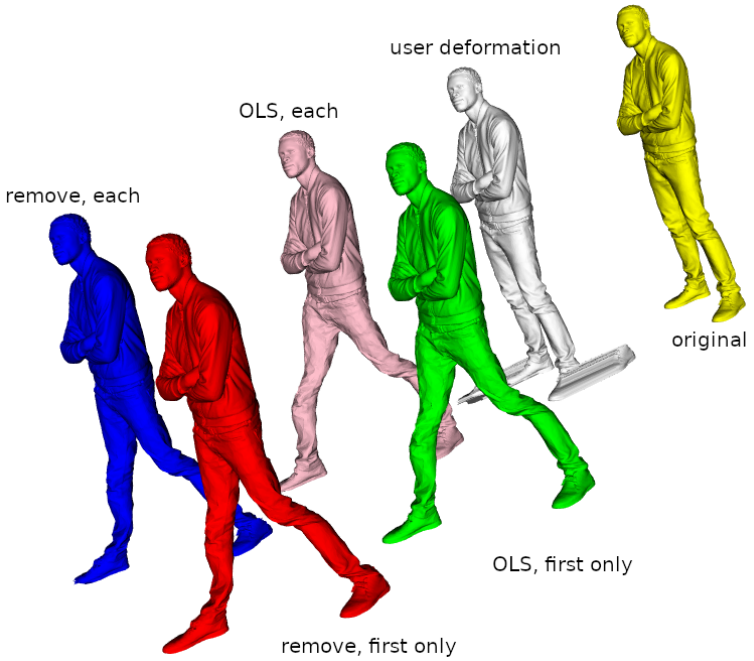

Figure 6.24: Results of four ARAP approaches with volume preservation, translation of men's legs.

atic (see 6.26), because of the rougher surface and noisiness of the original (real) data.

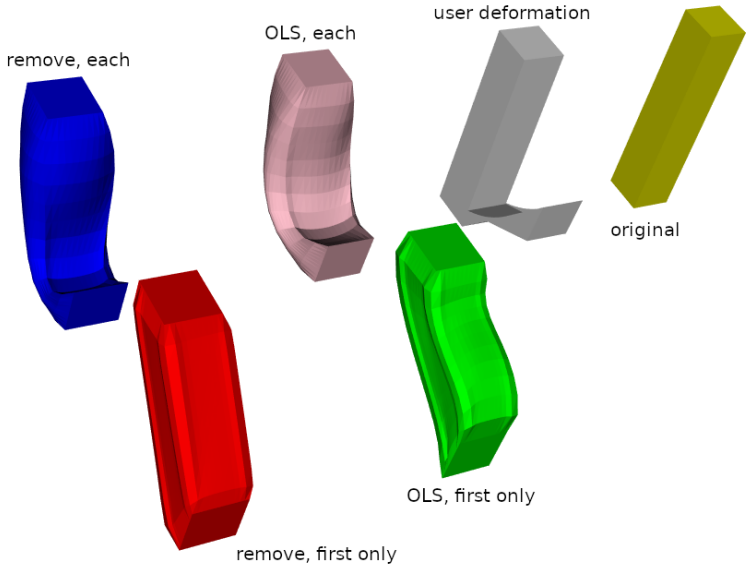

Figure 6.25: Results of four ARAP approaches with volume preservation, rotation of a box face around the X axis.

The ARAP method is capable of smooth deformation in general. How-

ever, it does not work as it is in the case of muscle data with the volume preservation requirement forced by switching between shape and volume constraints. The possibility is to combine PBD and ARAP methods and (in an ideal case) get the best of both worlds. A promising way is to use ARAP as an initial guess for the PBD, drastically reducing convergence time.

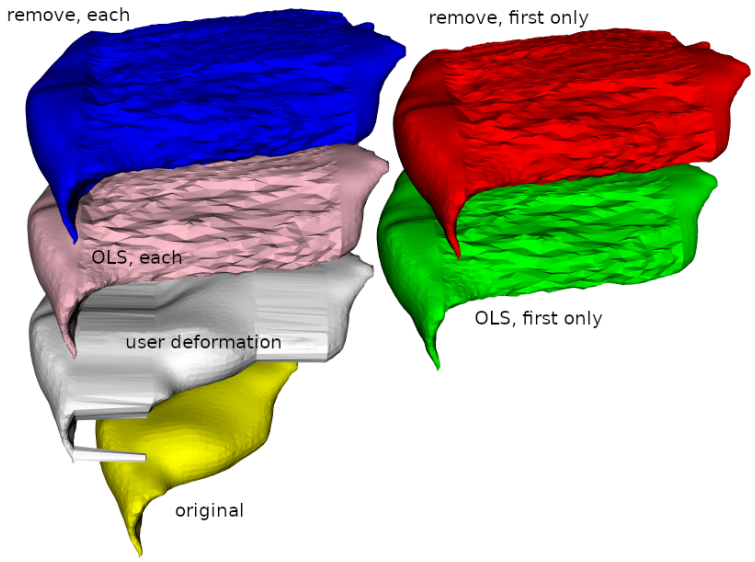

Figure 6.26: Results of four ARAP approaches with volume preservation, stretching gluteus maximus muscle.

We have successfully provided the muscle with a smooth surface only, but a simple "interleaving" approach test to include the restriction of the same volume does not work, so this is a candidate for future work, probably using some ideas from Dvorak's et al. work [25]. My initial idea was to alternate between all three aspects (local translation, rotation and volume preservation), but this simple idea failed because the volume and shape restriction goes against each other. So the final result either preserves volume but does not respect local shape features, or vice versa. The more complex approach, either involving internal structure [25] or involving a cost function and optimisation approach, is required; however, this is an idea for future work.

Experiments show that all of the mentioned methods have their advantages and drawbacks. Using ARAP for muscle modelling may be suitable, but there is a problem with putting together pure ARAP with volume preservation. Using RBF for approximation and interpolation purposes also seems promising. The challenge is finding suitable centre points and shape parameters to get the best (or at least plausible) results. PBD algorithm generates excellent visual results, for the most part, but with some problems in the joint area.

Chapter 7

Conclusion & Future Work

Approaches to muscle modelling are still evolving and cannot be expected to stop one day. All of the described approaches to muscle modelling provide good outcomes; however, each has some drawbacks. Some are inaccurate (Hill-type model, Via-points), some are accurate but difficult to set up or simply too slow to be useful (Finite element methods). Other methods are "compromise solutions" in terms of accuracy and computational complexity (MSS, PBD, ARAP). As you can probably imagine, there are still many open problems to solve.

7.1 The ambitious goal

The more ambitious goal for future work is to develop a static model of the musculoskeletal system using a representation other than a triangular surface mesh. Returning to the pipeline in section 1.2, this would mean changing step 4. (b). The model could be then used for a new, faster muscle modelling approach. The main idea is to use the RBF approximation method to satisfy these three points:

1. Smoothen the muscle and bone models to be more realistic, and overcoming the current problems with the PBD implementation (particularly the problem with rough surfaces near joints).

2. Reduce the number of parameters defining the model, thus reducing the computational complexity. The current number of parameters depends on number of verticies and connectivity, which should be further reduced.

3. Create the most realistic (in the terms of e.g. muscle forces) and visually plausible simulation possible.

The PBD focuses primarily on the first point but fails in the other two. The third point is met very well by finite element methods. However, they

fail fatally on the second point. Via-points and wrapping obstacles focus on satisfying the second but seldom satisfying the third point.

The second and third points are slightly contradictory. In general, more parameters are needed to obtain a more realistic output. The main idea for future work is to reduce the number of parameters using a Gaussian RBF function, which is used in many realistic scenarios. The Gaussian plate would approximate realistic musculoskeletal models using fewer parameters (few centre points and shape parameters) than the FEM grid, thus achieving similar results with fewer parameters. The implicit RBF was specifically chosen for muscle approximation to allow deformation of a portion of the muscle with only one manipulation of the centre point (movement or shape parameter change). The RBF method has already been successfully used in our research [46] as an attempt to recover the surface of the muscle attachment area, so a natural extension to recover the entire muscle surface is right on the table.

The main inspiration for the static model creation can be found in Carr et al. [14], where the authors create an implicit RBF surface from a point cloud. They are capable of reducing the number of parameters approximatelly ten times opposite to the triangular mesh parameters, maintaining the high accuracy of the resulting model, moreover, generating a smoother model (which is desired in many scenarios, muscle modelling is one of them).

The first example of such a model can be seen in Fig. 7.1, where two RBF functions are near each other and together create a single implicit surface.

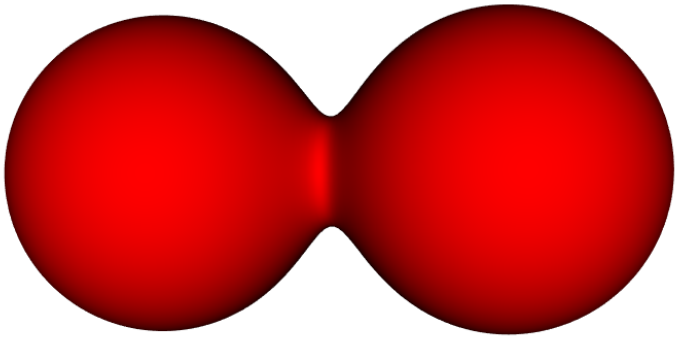

Figure 7.1: Two Gaussian RBF functions "joined" together, forming a single implicit surface.

Such a model would open a new possibility to develop a new approach to deformation of this model, which would theoretically allow smooth and rapid deformation of the muscle. The future deformation approach would ideally require to handle a collision detection and response, volume preservation and also take into account muscle anisotropy.

The ultimate challenge is changing implicit RBF parameters while maintaining the volume of the internal part bounded by the implicit surface.

It can be done by changing the shape parameters of a subset of elementary RBF to match the original volume of the muscle. Another challenge is the ability to avoid obstacles (bone models). A new issue arises: the bone may theoretically force the muscle to "split" itself (this is not an issue from the mathematical point of view because implicit RBF can describe not just one but multiple surfaces at once), which is not possible in a real scenario. The anisotropy is also a challenge because the anisotropy direction has to be taken into account during the deformation, and the direction may also change due to muscle movement. There are many problems which were not, to our best knowledge, dealt with yet.

7.2 ARAP & PBD

The second work for the future to consider is the combination of ARAP [79] and PBD [62] approaches to getting "the best of both worlds". The shape preservation of the ARAP approach is fast and accurate, but the volume preservation constraint is not solvable by simply introducing a new constraint into the system since the interleaving approach does not work there. For this reason, the idea of a PBD solver in the ARAP approach may be the key to solving the problem. Dvorak et al. [25] show some ideas. Unluckily, their approach is far from using it directly for muscle modelling problems. Our goal is to avoid the introduction of internal structure computation entirely to reach lower computational complexity, meaning that their approach has to be altered drastically.

The first option is to start with PBD and replace the shape preservation constraint with the shape preservation constraint from the ARAP approach. This would require a mathematical reformulation of the shape constraint and finding a gradient expression for the ARAP shape preservation constraint. The other option is to start with ARAP and replace the interleaving approach with gradient descent from PBD. Then, a volume constraint can be added. Either way, both approaches should end up with the same result.

Appendix A

Publications

- Cervenka, M., Kohout, J.:
  Fast and Realistic Approach to Virtual Muscle Deformation,
  in Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 5: HEALTHINF, ISBN 978-989-758-398-8, pages 217-227. (2020)
  UT WoS: 000571479400020, EID: 2-s2.0-85083710925, OBD: 43929104
  https://doi.org/10.5220/0009129302170227

- Kohout, J., Cervenka, M.:
  Non-planar Surface Shape Reconstruction from a Point Cloud in the Context of Muscles Attachments Estimation,
  in Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, ISBN 978-989-758-555-5, pages 236-243. (2022)
  UT WoS: ×, EID: ×, OBD: ×
  https://doi.org/10.5220/0010869600003124

- Kohout, J., Cervenka, M.:
  Muscle Deformation Using Position Based Dynamics,
  in: Ye X. et al. (eds) Biomedical Engineering Systems and Technologies. BIOSTEC 2020. Communications in Computer and Information Science
  vol 1400. Springer, Cham. (2021)
  UT WoS: ×, EID: 2-s2.0-85107281398, OBD: 43932927
  https://doi.org/10.1007/978-3-030-72379-8_24

- Cervenka, M., Skala, V.:
  Behavioral Study of Various Radial Basis Functions for Approximation and Interpolation Purposes,
  IEEE 18th World Symposium on Applied Machine Intelligence and Informatics, SAMI 2020,
  pp.135-140, ISBN 978-1-7281-314, Slovakia, (2020) (Scopus)

UT WoS: 000589772600026, EID: 2-s2.0-85087093548, OBD: 43929006
https://doi.org/10.1109/SAMI48414.2020.9108712

- Cervenka, M., Skala, V.:
Conditionality Analysis of the Radial Basis Function Matrix,
ICCSA 2020 proceedings, part II, LNCS, pp. 30-43,
Springer, (2020)
UT WoS: ×, EID: 2-s2.0-85093112881, OBD: 43932697
https://doi.org/10.1007/978-3-030-58802-1_3

- Cervenka, M., Smolik, M., Skala, V.:
A New Strategy for Scattered Data Approximation Using Radial Basis Functions Representing Points of Inflection,
Computational Science and Its Application, ICSSA 2019 proceedings, Part I, LNCS 11619, pp.322-226, ISSN 0302-9743, ISBN 978-3-030-24288-6, Springer, (2019)
UT WoS: 000661318700024, EID: 2-s2.0-85069157052, OBD: 43926678
https://doi.org/10.1007/978-3-030-24289-3_24

- Skala, V., Cervenka, M.:
Novel RBF Approximation Method Based on Geometrical Properties for Signal Processing with a New RBF Function: Experimental Comparison,
Informatics 2019, IEEE proceedings,
pp.357-362, ISBN 978-1-7281-3178-8, Poprad, Slovakia, (2019)
UT WoS: 000610452900074, EID: 2-s2.0-85087090327, OBD: 43929007
https://doi.org/10.1109/Informatics47936.2019.9119276

- Vasta, J., Skala, V., Smolik, M., Cervenka, M.:
Modified Radial Basis Functions Approximation Respecting Data Local Features,
Informatics 2019, IEEE proceedings,
pp.445-449, ISBN 978-1-7281-3178-8, Poprad, Slovakia, (2019)
UT WoS: 000610452900015, EID: 2-s2.0-8508762067, OBD: 43928987
https://doi.org/10.1109/Informatics47936.2019.9119330

- Skala, V., Karim, S., Cervenka, M.:
Finding Points of Importance for Radial Basis Function Approximation of Large Scattered Data,
Computational Science - ICCS 2020,
Part VI, LNCS 12142, pp. 239-250, Springer, (2020)
UT WoS: ×, EID: 2-s2.0-85087274721, OBD: 43932925
https://doi.org/10.1007/978-3-030-50433-5_19

Appendix B

Other activities

- Lecturer, Introduction to Computer Graphics (KIV/UPG):
  - Summer semester of 2018/2019, one lecture per week
  - Summer semester of 2019/2020, four lectures per week
  - Summer semester of 2020/2021, five lectures per week
  - Summer semester of 2021/2022, four lectures per week
- Lecturer, Programming Techniques (KIV/PT):
  - Winter semester of 2020/2021, two lectures per week
  - Winter semester of 2021/2022, two lectures per week
- Lecturer, Programming Techniques in English (KIV/PT-E)
  - Winter semester of 2021/2022, one lecture per week
- Internship, LMU Munchen, from 7$^{\text{th}}$ of June 2022 till 1$^{\text{st}}$ of July 2022
- Student member of the education council (ROV in Czech)

Bibliography

[1] Angles, B., Rebain, D., Macklin, M., Wyvill, B., Barthe, L., Lewis, J., Von Der Pahlen, J., Izadi, S., Valentin, J., Bouaziz, S., Tagliasacchi, A.: Viper: Volume invariant position-based elastic rods. Proc. ACM Comput. Graph. Interact. Tech. **2**(2) (jul 2019). https://doi.org/10.1145/3340260, https://doi.org/10.1145/3340260

[2] Arslan, Y.Z., Karabulut, D., Ortes, F., Popovic, M.: Exoskeletons, Exomusculatures, Exosuits: Dynamic Modeling and Simulation, pp. 305–331 (04 2019). https://doi.org/10.1016/B978-0-12-812939-5.00011-2

[3] Aubel, A., Thalmann, D.: Interactive modeling of the human musculature. pp. 167 – 255 (02 2001). https://doi.org/10.1109/CA.2001.982390

[4] Aubel, A., Thalmann, D.: Efficient muscle shape deformation (05 2002). https://doi.org/10.1007/978-0-306-47002-8_12

[5] Babuska, I., Melenk, J.: The partition of unity finite element method. International Journal for Numerical Methods in Engineering **40**, 38 (06 1995). https://doi.org/10.1002/nme.1459

[6] Bagchi, S., Mitra, S.: The nonuniform discrete fourier transform and its applications in filter design. i. 1-d. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing **43**(6), 422–433 (1996). https://doi.org/10.1109/82.502315

[7] Barzan, M., Carty, C., Maine, S., Brito da Luz, S., Lloyd, D., Modenese, L.: Subject-specific knee kinematics during walking in children and adolescents with recurrent patellar dislocation (10 2017)

[8] Blaheta, R.: Matematické modelování a metoda konečných prvků (2012), in czech language only

[9] Boubaker, B., Pato, M., Pires, E.: A finite element model of skeletal muscle. Virtual and Physical Prototyping **1**, 159–170 (09 2006). https://doi.org/10.1080/17452750601040626

[10] Buhmann, M.: Radial basis functions: Theory and implementations. Radial Basis Functions **12** (07 2003). https://doi.org/10.1017/CBO9780511543241

[11] Burzyński, S., Sabik, A., Witkowski, W., Łuczkiewicz, P.: Influence of the femoral offset on the muscles passive resistance in total hip arthroplasty. PLOS ONE **16**(5), 1–12 (05 2021). https://doi.org/10.1371/journal.pone.0250397

[12] Carbone, V., Fluit, R., Pellikaan, P., van der Krogt, M., Janssen, D., Damsgaard, M., Vigneron, L., Feilkas, T., Koopman, H., Verdonschot, N.: Tlem 2.0–a comprehensive musculoskeletal geometry dataset for subject-specific modeling of lower extremity. Journal of biomechanics **48** (01 2015). https://doi.org/10.1016/j.jbiomech.2014.12.034

[13] Carnevale, L., Anjos, G., Mangiavacchi, N.: Stream function-vorticity formulation applied in the conjugated heat problem using the fem with unstructured mesh (11 2018). https://doi.org/10.26678/ABCM.ENCIT2018.CIT18-0173

[14] Carr, J., Beatson, R., Cherrie, J., Mitchell, T., Fright, W., Mccallum, B., Evans, T.: Reconstruction and representation of 3d objects with radial basis functions. ACM SIGGRAPH (09 2001). https://doi.org/10.1145/383259.383266

[15] Catmull, E., Rom, R.: A class of local interpolating splines. Computer Aided Geometric Design - CAGD **74** (12 1974). https://doi.org/10.1016/B978-0-12-079050-0.50020-5

[16] Cervenka, M.: Non-rigid registration of superficial muscle-tendon fascicles. Bachelor's Thesis (2017), supervisor: Kohout, J.

[17] Cervenka, M.: Muscle Fibres Deformation using Particle System. Master's thesis (2019), supervisor: Kohout, J.

[18] Cervenka., M., Kohout., J.: Fast and realistic approach to virtual muscle deformation. In: Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies - HEALTHINF,. pp. 217–227. INSTICC, SciTePress (2020). https://doi.org/10.5220/0009129302170227

[19] Cervenka, M., Skala, V.: Behavioral study of various radial basis functions for approximation and interpolation purposes. In: 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI). pp. 135–140 (2020). https://doi.org/10.1109/SAMI48414.2020.9108712

[20] Cervenka, M., Skala, V.: Conditionality analysis of the radial basis function matrix (10 2020). https://doi.org/10.1007/978-3-030-58802-1_3

[21] Cervenka, M., Smolik, M., Skala, V.: A New Strategy for Scattered Data Approximation Using Radial Basis Functions Respecting Points of Inflection, pp. 322–336 (06 2019). https://doi.org/10.1007/978-3-030-24289-3_24

[22] Delp, S., Blemker, S.: Three-dimensional representation of complex muscle architectures and geometries. Annals of biomedical engineering **33**, 661–73 (06 2005). https://doi.org/10.1007/s10439-005-1433-7

[23] Dereshgi, H., Serbest, K., Sahin, S., Balik, B.: Skeletal muscle mechanics from hill-based muscle model to computer applications: State of the art review **2**, 27–39 (06 2021)

[24] Diffo Kaze, A., Maas, S., Arnoux, P.J., Wolf, C., Pape, D.: A finite element model of the lower limb during stance phase of gait cycle including the muscle forces. BioMedical Engineering OnLine **16**, 138 (12 2017). https://doi.org/10.1186/s12938-017-0428-6

[25] Dvořák, J., Káčereková, Z., Vaněček, P., Hruda, L., Váša, L.: As-rigid-as-possible volume tracking for time-varying surfaces. Computers & Graphics **102**, 329–338 (2022). https://doi.org/https://doi.org/10.1016/j.cag.2021.10.015, https://www.sciencedirect.com/science/article/pii/S0097849321002284

[26] Ezati, M., Ghannadi, B., McPhee, J.: A review of simulation methods for human movement dynamics with emphasis on gait. Multibody System Dynamics **47** (11 2019). https://doi.org/10.1007/s11044-019-09685-1

[27] Fasser, M.., Jokeit, M., Kalthoff, M., Gomez Romero, D.A., Trache, T., Snedeker, J.G., Farshad, M., Widmer, J.: Subject-specific alignment and mass distribution in musculoskeletal models of the lumbar spine. Frontiers in Bioengineering and Biotechnology **9** (2021), www.scopus.com, cited By :2

[28] Fougeron, N., Rohan, P.Y., Rose, J.L., Bonnet, X., Pillet, H.: Finite element analysis of the stump-ischial containment socket interaction: a technical note. Medical Engineering & Physics **105**, 103829 (06 2022). https://doi.org/10.1016/j.medengphy.2022.103829

[29] Fukuda, N., Otake, Y., Takao, M., Yokota, F., Ogawa, T., Uemura, K., Nakaya, R., Tamura, K., Grupp, R., Farvardin, A., Sugano, N.,

Sato, Y.: Estimation of attachment regions of hip muscles in ct image using muscle attachment probabilistic atlas constructed from measurements in eight cadavers. International Journal of Computer Assisted Radiology and Surgery **12** (02 2017). https://doi.org/10.1007/s11548-016-1519-8

[30] Garner, B., Pandy, M.: Estimation of musculotendon properties in the human upper limb. Annals of biomedical engineering **31**, 207–20 (03 2003). https://doi.org/10.1114/1.1540105

[31] George-Ghiocel, O., Băbuț, C., Ungureanu, N., Deleanu, L.: Fem analysis of storz coupling **6**, 249–258 (09 2021)

[32] Georgii, J., Westermann, R.: Mass-spring systems on the gpu. Simulation Modelling Practice and Theory **13**, 693–702 (11 2005). https://doi.org/10.1016/j.simpat.2005.08.004

[33] Halton, J.: Algorithm 247: Radical-inverse quasi-random point sequence. Commun. ACM **7**, 701–702 (12 1964). https://doi.org/10.1145/355588.365104

[34] Hardy, R.: Multiquadric equations of topography and other irregular surfaces. Journal of Geophysical Research **76**, 1905–1915 (03 1971). https://doi.org/10.1029/JB076i008p01905

[35] Havlíček, O.: Fast collision detection in the context of muscle deformation by a position based dynamics method. Bachelor's Thesis (2021), supervisor: Kohout, J.

[36] Haykin, S.: Neural Networks: A Comprehensive Foundation (2nd Edition) Neural Networks: A Comprehensive Foundation (01 1998)

[37] Hill, A.: The heat of shortening and the dynamic constants of muscle. Proc. R. Soc. Lond. B **126**, 612–745 (01 1938)

[38] Hong, M., Jung, S., Choi, M.H., Welch, S.W.: Fast volume preservation for a mass-spring system. IEEE Computer Graphics and Applications **26**(5), 83–91 (2006). https://doi.org/10.1109/MCG.2006.104

[39] Hájková, J., Kohout, J.: Human body model movement support: Automatic muscle control curves computation. pp. 196–211 (05 2014). https://doi.org/10.1007/978-3-319-07148-0_18

[40] Janák, T.: Fast soft-body models for musculoskeletal modelling. Tech. rep., University of West Bohemia, Faculty of Applied Sciences (2012)

[41] Janák., T., Kohout., J.: Deformable muscle models for motion simulation. In: Proceedings of the 9th International Conference on Computer Graphics Theory and Applications - GRAPP, (VISIGRAPP 2014). pp. 301–311. INSTICC, SciTePress (2014). https://doi.org/10.5220/0004678903010311

[42] Jin, Z., Li, J., Chen, Z.: Computational Modelling of Biomechanics and Biotribology in the Musculoskeletal System (10 2020)

[43] Kaptein, B., van der Helm, F.: Estimating muscle attachment contours by transforming geometrical bone models. Journal of Biomechanics **37**(3), 263–273 (2004). https://doi.org/https://doi.org/10.1016/j.jbiomech.2003.08.005, https://www.sciencedirect.com/science/article/pii/S0021929003003257

[44] Kellnhofer, P., Kohout, J.: Time-convenient deformation of musculoskeletal system (09 2012)

[45] Kohout, J., Clapworthy, G.J., Zhao, Y., Tao, Y., Gonzalez-Garcia, G., Dong, F., Wei, H., Kohoutová, E.: Patient-specific fibre-based models of muscle wrapping. Interface Focus **3**(2), 20120062 (2013). https://doi.org/10.1098/rsfs.2012.0062, https://royalsocietypublishing.org/doi/abs/10.1098/rsfs.2012.0062

[46] Kohout, J., Cervenka, M.: Non-planar surface shape reconstruction from a point cloud in the context of muscles attachments estimation. pp. 236–243 (01 2022). https://doi.org/10.5220/0010869600003124

[47] Kohout, J., Cholt, D.: Automatic reconstruction of the muscle architecture from the superficial layer fibres data. Computer Methods and Programs in Biomedicine **150** (08 2017). https://doi.org/10.1016/j.cmpb.2017.08.002

[48] Kohout, J., Kukačka, M.: Real-time modelling of fibrous muscle. Computer Graphics Forum **33** (05 2014). https://doi.org/10.1111/cgf.12354

[49] Kohout, J., Červenka, M.: Muscle deformation using position based dynamics. In: Ye, X., Soares, F., De Maria, E., Gómez Vilda, P., Cabitza, F., Fred, A., Gamboa, H. (eds.) Biomedical Engineering Systems and Technologies. pp. 486–509. Springer International Publishing, Cham (2021)

[50] Kutilek, P., Viteckova, S., Svoboda, Z., Smrcka, P.: The use of artificial neural networks to predict the muscle behavior. Central European Journal of Engineering **3** (09 2013). https://doi.org/10.2478/s13531-012-0067-4

[51] Larson, M., Bengzon, F.: The Finite Element Method: Theory, Implementation, and Applications, vol. 10 (01 2013). https://doi.org/10.1007/978-3-642-33287-6

[52] Lee, D., Glueck, M., Khan, A., Fiume, E., Jackson, K.: Modeling and simulation of skeletal muscle for computer graphics: A survey. Foundations and Trends® in Computer Graphics and Vision **7**, 229 (01 2012). https://doi.org/10.1561/0600000036

[53] Lee, D., Li, Z., Sohail, Q.Z., Jackson, K., Fiume, E., Agur, A.: A three-dimensional approach to pennation angle estimation for human skeletal muscle. Computer methods in biomechanics and biomedical engineering **18**, 1–11 (05 2014). https://doi.org/10.1080/10255842.2014.917294

[54] Lewis, J., Pighin, F., Anjyo, K.: Scattered data interpolation for computer graphics. ACM SIGGRAPH 2014 Courses, SIGGRAPH 2014 (01 2010). https://doi.org/10.1145/1900520.1900522

[55] Li, H., Sumner, R., Pauly, M.: Global correspondence optimization for non-rigid registration of depth scans. Computer Graphics Forum **27** (07 2008). https://doi.org/10.1111/j.1467-8659.2008.01282.x

[56] Lloyd, J.E., Roewer-Després, F., Stavness, I.: Muscle path wrapping on arbitrary surfaces. IEEE Transactions on Biomedical Engineering **68**(2), 628–638 (2021). https://doi.org/10.1109/TBME.2020.3009922

[57] Macklin, M., Müller, M., Chentanez, N.: Xpbd: Position-based simulation of compliant constrained dynamics (10 2016). https://doi.org/10.1145/2994258.2994272

[58] Martins, J.A.C., Pato, M.P.M., Pires, E.B.: A finite element model of skeletal muscles. Virtual and Physical Prototyping **1**(3), 159–170 (2006). https://doi.org/10.1080/17452750601040626, https://doi.org/10.1080/17452750601040626

[59] Martins, J., Pires, E., Salvado, R., Dinis, P.: A numerical model of passive and active behavior of skeletal muscles. Computer Methods in Applied Mechanics and Engineering **151**(3), 419–433 (1998). https://doi.org/https://doi.org/10.1016/S0045-7825(97)00162-X, https://www.sciencedirect.com/science/article/pii/S004578259700162X, containing papers presented at the Symposium on Advances in Computational Mechanics

[60] of Medicine, N.L.: Visible human project, https://www.nlm.nih.gov/research/visible/visible_human.html

[61] Modenese, L., Kohout, J.: Automated generation of three-dimensional complex muscle geometries for use in personalised musculoskeletal models. Annals of Biomedical Engineering **48** (03 2020). https://doi.org/10.1007/s10439-020-02490-4

[62] Müller, M., Heidelberger, B., Hennix, M., Ratcliff, J.: Position based dynamics. Journal of Visual Communication and Image Representation **18**(2), 109–118 (2007). https://doi.org/https://doi.org/10.1016/j.jvcir.2007.01.005, https://www.sciencedirect.com/science/article/pii/S1047320307000065

[63] Ni, R., Meyer, C., Blemker, S., Hart, J., Feng, X.: Automatic segmentation of all lower limb muscles from high-resolution magnetic resonance imaging using a cascaded three-dimensional deep convolutional neural network. Journal of Medical Imaging **6**, 1 (12 2019). https://doi.org/10.1117/1.JMI.6.4.044009

[64] Nolan, D., Gower, A., Destrade, M., Ogden, R., McGarry, P.: A robust anisotropic hyperelastic formulation for the modelling of soft tissue (09 2020)

[65] Oatis, C.A.: Biomechanics of skeletal muscle (2017)

[66] Oberhofer, K., Mithraratne, K., Stott, N., Anderson, I.: Anatomically-based musculoskeletal modeling: Prediction and validation of muscle deformation during walking. The Visual Computer **25**, 843–851 (09 2009). https://doi.org/10.1007/s00371-009-0314-8

[67] Otake, Y., Yokota, F., Fukuda, N., Takao, M., Takagi, S., Yamamura, N., O'Donnell, L.J., Westin, C.F., Sugano, N., Sato, Y.: Patient-specific skeletal muscle fiber modeling from structure tensor field of clinical ct images. In: Descoteaux, M., Maier-Hein, L., Franz, A., Jannin, P., Collins, D.L., Duchesne, S. (eds.) Medical Image Computing and Computer Assisted Intervention – MICCAI 2017. pp. 656–663. Springer International Publishing, Cham (2017)

[68] Oudeman, J., Nederveen, A., Strijkers, G., Maas, M., Luijten, P., Froeling, M.: Techniques and applications of skeletal muscle diffusion tensor imaging: A review. Journal of Magnetic Resonance Imaging **43**, n/a–n/a (07 2015). https://doi.org/10.1002/jmri.25016

[69] Pan, R., Skala, V.: Continuous global optimization in surface reconstruction from an oriented point cloud. Computer-Aided Design **43**, 896–901 (08 2011). https://doi.org/10.1016/j.cad.2011.03.005

[70] Park, J.S., Chung, M., Hwang, S., Lee, Y., Har, D.H., Park, H.: Visible korean human: Improved serially sectioned images of the entire body.

IEEE transactions on medical imaging **24**, 352–60 (04 2005). https://doi.org/10.1109/TMI.2004.842454

[71] Pellikaan, P., van der Krogt, M., Carbone, V., Fluit, R., Vigneron, L., Van Deun, J., Verdonschot, N., Koopman, H.: Evaluation of a morphing based method to estimate muscle attachment sites of the lower extremity. Journal of Biomechanics **47** (01 2013). https://doi.org/10.1016/j.jbiomech.2013.12.010

[72] Ranzenberger LR, S.T.: Diffusion tensor imaging (2021), https://www.ncbi.nlm.nih.gov/books/NBK537361/

[73] Romeo, M., Monteagudo, C., Sánchez-Quirós, D.: Muscle Simulation with Extended Position Based Dynamics. In: García-Fernández, I., Ureña, C. (eds.) Spanish Computer Graphics Conference (CEIG). The Eurographics Association (2018). https://doi.org/10.2312/ceig.20181146

[74] Çağlar Seylan, Sahillioğlu, Y.: 3d shape deformation using stick figures. Computer-Aided Design **151**, 103352 (2022). https://doi.org/https://doi.org/10.1016/j.cad.2022.103352, https://www.sciencedirect.com/science/article/pii/S0010448522001075

[75] Shi, X., Lu, L.Z., Wang, H.: New superconvergence estimates of fem for time-dependent joule heating problem. Computers & Mathematics with Applications **111**, 91–97 (04 2022). https://doi.org/10.1016/j.camwa.2022.02.011

[76] Skala, V., Abdul Karim, A.P.T.D.S.A., Cervenka, M.: Finding Points of Importance for Radial Basis Function Approximation of Large Scattered Data, pp. 239–250 (06 2020). https://doi.org/10.1007/978-3-030-50433-5_19

[77] Skala, V., Cervenka, M.: Novel rbf approximation method based on geometrical properties for signal processing with a new rbf function: Experimental comparison (06 2019). https://doi.org/10.1109/Informatics47936.2019.9119276

[78] Skala, V., Kansa, E.: Why is the least square error method dangerous? Computación y Sistemas **25** (02 2021). https://doi.org/10.13053/cys-25-1-3473

[79] Sorkine, O., Alexa, M.: As-Rigid-As-Possible Surface Modeling. In: Belyaev, A., Garland, M. (eds.) Geometry Processing. The Eurographics Association (2007). https://doi.org/10.2312/SGP/SGP07/109-116

[80] Spitzer, V., Ackerman, M., Scherzinger, A., Whitlock, D.: The visible human male: A technical report. Journal of the American Medical Informatics Association:JAMIA **3**, 118–30 (03 1996). https://doi.org/10.1136/jamia.1996.96236280

[81] Sun, X., Wang, H., Wang, W., Li, N., Hamalainen, T., Ristaniemi, T., Liu, C.: A statistical model of spine shape and material for population-oriented biomechanical simulation. IEEE Access **PP**, 1–1 (11 2021). https://doi.org/10.1109/ACCESS.2021.3129097

[82] Uhlir, K., Skala, V.: Reconstruction of damaged images using radial basis functions. 13th European Signal Processing Conference, EUSIPCO 2005 (01 2005)

[83] Valente, G., Martelli, S., Taddei, F., Farinella, G., Viceconti, M.: Muscle discretization affects the loading transferred to bones in lower-limb musculoskeletal models. Proceedings of the Institution of Mechanical Engineers. Part H, Journal of engineering in medicine **226**, 161–9 (02 2012). https://doi.org/10.1177/0954411911425863

[84] Vasta, J., Skala, V., Smolik, M., Cervenka, M.: Modified radial basis functions approximation respecting data local features (06 2020). https://doi.org/10.1109/Informatics47936.2019.9119330

[85] Vila Pouca, M., Areias, P., Göktepe, S., Ashton-Miller, J., Natal Jorge, R., Parente, M.: Modeling permanent deformation during low-cycle fatigue: Application to the pelvic floor muscles during labor. Journal of the Mechanics and Physics of Solids **164**, 104908 (04 2022). https://doi.org/10.1016/j.jmps.2022.104908

[86] Wade, S., Strader, C., Fitzpatrick, L., Anthony, M., O'Malley, C.: Estimating prevalence of osteoporosis: Examples from industrialized countries. Archives of osteoporosis **9**, 182 (12 2014). https://doi.org/10.1007/s11657-014-0182-3

[87] Wang, B., Matcuk, G., Barbič, J.: Modeling of personalized anatomy using plastic strains. ACM Trans. Graph. **40**(2) (jun 2021). https://doi.org/10.1145/3443703, https://doi.org/10.1145/3443703

[88] Wei, Y., Zou, Z., Wei, G., Ren, L., Qian, Z.: Subject-specific finite element modelling of the human hand complex: Muscle-driven simulations and experimental validation. Annals of Biomedical Engineering **48** (12 2019). https://doi.org/10.1007/s10439-019-02439-2

[89] Winter, V.J.: Numerical methods for scalar convection-dominated problems (2014)

[90] Yuksel, C., Schaefer, S., Keyser, J.: On the parameterization of catmull-rom curves. In: 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling. pp. 47–53. ACM, New York, NY, USA (2009). https://doi.org/10.1145/1629255.1629262, http://doi.acm.org/10.1145/1629255.1629262

[91] Zajac, F.: Muscle and tendon: Properties, models, scaling, and application to biomechanics and motor control. Critical reviews in biomedical engineering **17**, 359–411 (02 1989)

[92] Zhang, G., Wang, C., Liu, Q., Wei, J., Luo, C., Duan, L., Long, J., Zhang, X., Wang, G.: Development of skeletal muscle model for bridge-style movement rehabilitation. Journal of Physics: Conference Series **2026**, 012061 (09 2021). https://doi.org/10.1088/1742-6596/2026/1/012061

[93] Zhang, S.X., Heng, P.A., Liu, Z.J., Tan, L.W., Qiu, M.G., Li, Q.Y., Liao, R.X., Li, K., Cui, G.Y., Guo, Y.L., Yang, X.P., Liu, G.J., Shan, J.L., Liu, J.J., Zhang, W.G., Chen, X.H., Chen, J.H., Wang, J., Chen, W., Xie, Y.M.: Creation of the chinese visible human data set. Anatomical record. Part B, New anatomist **275**, 190–5 (11 2003). https://doi.org/10.1002/ar.b.10035

[94] Zhao, Y., Clapworthy, G., Kohout, J., Dong, F., Tao, Y., Wei, S., Mcfarlane, N.: Laplacian musculoskeletal deformation for patient-specific simulation and visualisation. pp. 505–510 (07 2013). https://doi.org/10.1109/IV.2013.67