Автономная некоммерческая
образовательная организация высшего образования
«НАУЧНО-ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «СИРИУС»

Научный центр информационных технологий и искусственного интеллекта

направление «Математическое моделирование в биомедицине и геофизике»

К ЗАЩИТЕ ДОПУСТИТЬ
Руководитель направления
«Математическое моделирование в биомедицине и геофизике»,
к.т.н., д.э.н.
_____ М.В. Ширяев
«____» _____ 2025 г.

«**БИОМЕХАНИЧЕСКАЯ МОДЕЛЬ КИСТИ РУКИ ЧЕЛОВЕКА**»

Магистерская диссертация
по направлению подготовки 01.04.02 «Прикладная математика и информатика»
направленность (профиль) «Математическое моделирование процессов и материалов»

Студент гр. М01ММ-23

_____ Г.К. Савон
«____» _____ 2025 г.

Согласовано
Консультант
заместитель директора по науке
ИВМ РАН имени Г. И. Марчука,
д.ф.-м.н., чл.-корр. РАН
_____Ю.В. Василевский
«____» _____ 2025 г.

Руководитель
доцент направления
«Математическое моделирование в биомедицине и геофизике» к.ф.-м.н.
_____ В.Ю. Саламатова
«____» _____ 2025 г.

Федеральная территория «Сириус» - 2025

Автономная некоммерческая
образовательная организация высшего образования
«НАУЧНО-ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «СИРИУС»

Научный центр информационных технологий и искусственного интеллекта

направление «Математическое моделирование в биомедицине и геофизике»

УТВЕРДИТЬ
Руководитель направления
«Математическое моделирование в биомедицине и геофизике»,
к.т.н., д.э.н.
_____ М.В. Ширяев
«____» _____ 2025 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ
на выполнение магистерской диссертации
по направлению подготовки 01.04.02 Прикладная математика и информатика
направленность (профиль) «Математическое моделирование процессов и материалов»

Савон Галина Константиновна

1. Тема: «Биомеханическая модель кисти руки человека».

2. Цель: разработка и апробация подхода к интеграции экспериментальных данных (углов суставов и электромиографических сигналов) в биомеханическую модель кисти человека с использованием OpenSim для последующего анализа мышечной активности и потенциального применения в управлении протезами и ортезами.

3. Задачи:

- ознакомление с анатомическими особенностями мышц и суставов кисти и предплечья, регистрации мышечной активности;
- проведение обзора существующих моделей кисти для использования в OpenSim, выбор наиболее подходящей;
- разработка программы для обработки и сегментации экспериментальных данных;

- выполнение калибровки углов и интеграция их в выбранную OpenSim модель;
- визуализация жестов;
- проведение расчетов мышечной активности при помощи встроенных методов в OpenSim. Сравнение результатов с экспериментальными ЭМГ-данными.

4. Рабочий график (план) выполнения магистерской диссертации:

№	Перечень заданий	Сроки выполнения
1.	Анализ основ движения кисти. Обзор моделей и методов регистрации ЭМГ	18.02.2025 – 15.03.2025
2.	Подбор и адаптация модели кисти в OpenSim. Реализация кода предобработки из базы данных NinaPro	16.03.2025 – 12.04.2025
3.	Интеграция угловых данных в модель, визуализация движений. Расчет удлинения мышц	13.04.2025 – 01.05.2025
4.	Расчет мышечной активности методом оптимизации. Сравнение с исходными ЭМГ-данными	01.05.2025 – 30.05.2025

Дата выдачи: «18» февраля 2025 г.

Руководитель:

_____ В.Ю. Саламатова

Задание принял к исполнению:

Студент группы М01ММ-23

_____ Г.К. Савон

«18» февраля 2025 г.

Реферат

Выпускная квалификационная работа, 81 страница, 33 рисунка, 2 таблицы, 43 источника, 7 приложений.

БИОМЕХАНИКА, КИСТЬ, ЭЛЕКТРОМИОГРАФИЯ, МЫШЕЧНО-СУХОЖИЛЬНАЯ МОДЕЛЬ, NINAPRO, CYBERGLOVE, OPENSIM, ОБРАТНАЯ СИМУЛЯЦИЯ, STATIC OPTIMIZATION

Объектом исследования является нейромышечная активность мышц предплечья при выполнении движений пальцами кисти, предметом – возможность её оценки на основе экспериментальных данных при помощи биомеханического моделирования.

Целью работы являлась разработка и апробация подхода к восстановлению мышечной активности с применением модели кисти в OpenSim, основанной на кинематических данных и пЭМГ-сигналах из базы NinaPro DB1.

В качестве исходных данных используются углы сгибания и разгибания суставов пальцев, снятые с перчатки CyberGlove II. Данные были интегрированы в модель посредством Python API. Для восстановления мышечной активности использовался метод Static Optimization. Получившиеся активации сравнивались с исходными пЭМГ-данными.

Результаты показали, что при текущей конфигурации модели основную нагрузку брали на себя координатные актуаторы, что снижало уровень активации мышц-агонистов. Вместе с этим растяжения у мышц антагонистов были физиологичными. Это указывает на необходимость донастройки модели для улучшения качества обратной симуляции.

После частичной корректировки модели удалось получить физиологичные активации. Метод показал потенциал для достоверной реконструкции мышечной активности. Подход реализуется в открытом ПО, применим в реабилитации, протезировании и может служить анатомически обоснованной альтернативой существующим методам.

The Abstract

Graduation qualification work, 81 pages, 33 figures, 2 tables, 43 sources, 7 applications.

BIOMECHANICS, HAND, ELECTROMYOGRAPHY, MUSCLE-TENDON MODEL, NINAPRO, CYBERGLOVE, OPENSIM, INVERSE SIMULATION, STATIC OPTIMIZATION

The object of the study is the neuromuscular activity of forearm muscles during finger movements, and the subject is the possibility of its assessment based on experimental data using biomechanical modeling.
The aim of the work was to develop and test an approach for reconstructing muscle activation using a hand model in OpenSim, based on kinematic data and surface EMG signals from the NinaPro DB1 database.

The input data included flexion and extension angles of finger joints recorded with the CyberGlove II. These data were integrated into the model via the Python API. To estimate muscle activation, the Static Optimization method was used. The resulting activations were compared with the original surface EMG data.

The results showed that, in the current model configuration, most of the load was carried by coordinate actuators, reducing activation of agonist muscles. At the same time, the stretching of antagonist muscles appeared physiologically accurate. This indicates the need for further model tuning to improve the quality of inverse simulation.

After partial model adjustment, physiologically plausible activations were obtained, especially in antagonist muscles due to passive stretching. The method demonstrated potential for reliable reconstruction of muscle activity. The approach is implemented using open-source software, applicable in rehabilitation and prosthetics, and can serve as an anatomically grounded alternative to existing methods.

Сокращения, обозначения, термины и определения

В настоящей работе применяют следующие термины с соответствующими определениями:

- биомеханика – наука, изучающая биологические системы (в том числе человека), механику их движений на основе законов механики;
- ампутант – человек, частично или полностью утративший конечность в результате травмы или хирургической операции;
- агонист – мышца, которая своим сокращением вызывает движение биологической системы;
- антогонист – мышца, противодействующая движению, которое вызвано мышцей-агонистом;
- эндомизий – тонкий слой соединительной ткани, окружающий каждое отдельное мышечное волокно;
- перимизий – соединительнотканная оболочка, объединяющая несколько мышечных волокон в пучки (двигательные единицы);
- эпимизий – внешняя соединительнотканная оболочка всей скелетной мышцы;
- актино-миозиновый мостик – временное соединение между молекулами актина и миозина, возникающее в процессе мышечного сокращения и обеспечивающее генерацию силы;
- миофибрилла – структурный элекмент мышечного волокна, обеспечивающий активное сокращение мышц;
- титин – гигантский белок, связывающий Z-диск и миозиновую нить в саркомере;
- мотонейрон – двигательный нейрон, передающий нервный импульс от центральной нервной системы к мышечным волокнам и вызывающий их сокращение;
- сухожилие – волокнистая соединительная ткань, соединяющая мышцу с костью и передающая механическую силу, создаваемую мышцей;

- мышечно-сухожильный комплекс (мышечно-сухожильный элемент) – функциональное объединение мышечного волокна и сухожилия, обеспечивающее передачу силы от мышцы к кости и участвующее в управлении движением;
- пястно-фаланговый сустав (метакарпофаланговый сустав, MCP) – сустав между пястной костью и проксимальной фалангой пальца;
- межфаланговый сустав – сустав между фалангами одного пальца;
- проксимальный сустав – сустав пальца, расположенный ближе к туловищу или основанию конечности;
- дистальный сустав – сустав пальца, расположенный дальше от туловища, ближе к кончику пальца;
- неинвазивный метод – это способ получения информации о состоянии организма без проникновения внутрь тела и без нарушения целостности кожных покровов;
- парсинг – процесс автоматического анализа и структурирования данных из исходного формата;

В настоящей работе применяют следующие сокращения и обозначения:

- БД – база данных
- ЭМГ – электромиография
- пЭМГ – поверхностная электромиография
- ВП пЭМГ – высокоплотная электромиография
- ПО – программное обеспечение
- ДЕ – двигательная единица
- SO – Static Optimization
- CMC – Computed Muscle Control

Оглавление

Введение

Биомеханика – это наука, изучающая движение биологических объектов, в первую очередь человека. В отношении человека биомеханика объединяет в себе механику, физиологию, анатомию и решает задачи количественной оценки движений, расчета прикладываемых усилий и нагрузки на ткани и суставы в нормальном состоянии, а также при наличии патологий [1-2]. Исследования биомеханики широко применяются в травматологии, спортивной медицине, ортопедии, нейрореабилитации и протезировании [3-4,6].

В исследованиях биомеханики человека используются экспериментальные методы сбора данных, описывающих движение, такие как:

- поверхностная и игольчатая электромиография – для регистрации электрических сигналов от мышц [5,9];
- оптические системы захвата движений – для записи положения маркеров на теле [1];
- инерциальные измерительные устройства (IMU) – для фиксации ускорений и угловых скоростей отдельных сегментов тела [1];
- тензоплатформы и силовые датчики – для измерения силы взаимодействия тела с опорой [1].

Данные, полученные при помощи этих средств, помогают строить и калибровать биомеханические модели, в которых численно воспроизводятся реальные движения человека. Это позволяет не только визуализировать движения, но и оценивать нагрузку на суставы и поведение мышечного каркаса. Примеры исследований:

1. Измерение реактивных сил при ходьбе в реальных условиях [9]

а) **Цель исследования:** Разработка системы для автоматической генерации лицевых и шейных мышц через видел или изображения с передачей выражений на биомеханическую модель лица.

б) **Методы:** Предсказание активации мышц на основе FACS Action Units

2. Оценка кинематики нижних конечностей с помощью EKF и малого количества IMU [10]

а) **Цель исследования:** Разработка метода для оценки кинематики таза, бедра, голени и стопы, используя 2-3 IMU, и достичь точности, сопоставимой с OMC, с применением Extended Kalman Filter.

б) **Методы:** IMU: на тазу и ногах. Применение EKF с ограничениями.

3. Подходит ли OpenSim для анализа жевательной системы? [11]

а) **Цель исследования:** анализ движения нижней челюсти, оценка усилий жевательных мышц. Выделение преимуществ и недостатков ПО.

б) **Методы:** Использование модели нижней челюсти в OpenSim. Траектория движения, полученная с помошью трехмерного видеоанализа. Расчет мышечной активности методом StaticOptimization. Сравнительный анализ моделирования с данными, полученными по ЭМГ измерениям.

На сегодняшний день наблюдается рост количества исследований в этой области, что обусловлено развитием высокоточных технологий и их доступностью, сформированными обширными медицинскими базами данных, а также возможностью применения методов машинного обучения для анализа больших массивов биомеханических параметров. Благодаря алгоритмам машинного обучения возможно индивидуализировать модели и получать паттерны активности для разработки систем управления протезами и экзоскелетами. Примеры исследований:

1. Нейромышечный контроль лицево-шейно-головного биомеханического комплекса с переносом мимики с видео [12]

а) **Цель исследования:** Разработка системы для автоматической генерации лицевых и шейных мышц через видео или изображения с передачей выражений на биомеханическую модель лица.

б) **Методы:** Предсказание активации мышц на основе FACS Action Units. Активируемая мышцами биомеханическая модель.

2. Универсальная модель на основе transfer learning для непрерывной оценки углов в суставах пальцев у нового пользователя [13]

а) **Цель исследования:** разработка единой модели для всех субъектов (с возможностью адаптации к новым субъектам)

б) **Методы:** обучение на данных 35 испытуемых из БД NinaPro. Архитектура LSTA-Conv + стратегия Subjects Adversarial Regularzation (SAR)

Одной из наиболее сложных и актуальных задач биомеханики движения является моделирование кисти руки человека. Кисть обладает тонкой моторикой и применяется для большинства повседневных задач. Повреждение и частичная или полная деактивация этой части тела значительно снижает качество жизни. Соответственно разработка точной модели кисти необходима и для анализа подобных нарушений и для реабилитации (при таковой возможности) или протезирования. Однако сложность моделирования заключается в том, что движение кисти является мелкой моторикой, она управляется множеством мелких мышц и описание положений суставов во времени включает в себя больше, чем два десятка угловых параметров. Более того, даже набор мышц предплечья у здоровых людей частично отличается от субъекта к субъекту [7]. Эти биологические особенности кисти делает данную задачу сложной с точки зрения определения подхода и анализа экспериментальных данных, но все еще очень значимой для человечества.

Основной метод для моделирования движений кисти использует пЭМГ (поверхностной электромиографии) с мышц предплечья [5]. Подавляющая часть исследований подразумевает под собой классификацию «жестов» (определенного набора движений, наиболее используемых в повседневной жизни) по ЭМГ сигналу. Примеры исследований:

1. Классификация движений человеческой руки с использованием поверхностной ЭМГ для миоэлектрическоого управления [14].

а) **Цель исследования:** разработка и тестированиие метода классификаций движений кисти на основе сигналов пЭМГ для реализации в системе управления протезами и человеко-машинными интерфейсами.

б) **Методы:** данные пЭМГ во время выполнения движений руки у здоровых людей и у ампутантов (частично из NinaPro). Предобработка сигналов: фильтрация, скользящее окно. Извлечение признаков мышечных синергий через NMF (неотрицательное матричное факторизование). Сравнение четырех классификаторов машинного обучения.

2. Распознавание жестов на основе поверхностной электромиографии с применением многовидового глубокого обучения [15].

а) **Цель исследования:** повышение точности распознавания жестов на основе пЭМГ с использованием малоканальных записей, применяя подход многовидового глубокого обучения. Объединение преимуществ классических признаков пЭМГ и возможностей сверточных нейронных сетей для повышения точности и устойчивости систем управления жестами.

б) **Методы:** извлечение данных из 11 БД (в том числе NinaPro и BioPatRec), преобразование признаков в изображения сигналов, multi-view CNN (многовидовая сверточная нейронная сеть) с параллельными потоками.

3. Одновременное управление положениями суставов кисти человека и силой захвата на основе ВП пЭМГ и глубокого обучения [16].

а) **Цель исследования:** объединение управления положениями суставов кисти и силой захвата с помощью HD-ЭМГ (матрица из электродов) и нейронной сети.

б) **Методы:** Система ВП пЭМГ с высокой плотностью (массив 8x8), регистрирующая ЭМГ-сигнал с предплечья. Полносвязная DNN, обученная на совокупности данных (углы и зила захвата) при выполнении двух- и трехпальцевых захватов.

Но также был найдено исследование, в котром предсказывалась непрерывная кинематика, а не классификация: Непрерывная оценка кинематики пальцев на основе sEMG с использованием крупномасштабной временной сверточной сети [17].

Цель исследования: реализация непрерывной оценки 10 суставных углов пальцев кисти по 12-канальным пЭМГ-сигналам.

Методы: обучение и тестирование на NinaPro DB (несколько движений). LS-TCN для извлечения временных признаков.

В этой дипломной работе было решено сформулировать задачу иначе и исследовать возможность получения изначальных активация мышц при помощи Inverse dynamics и их сопоставления со значениями ЭМГ из экспериментальных данных. При проведении литературного обзора была обнаружена всего одна работа, в которой пытались бы сделать непрерывную оценку движения по ЭМГ (описание этой работы приведено выше). Во всех остальных работах они использовались исключительно для классификации данных. Такое исследование может значительно расширить возможности протезирования и улучшить качество жизни у людей с патологиями и травмами кисти.

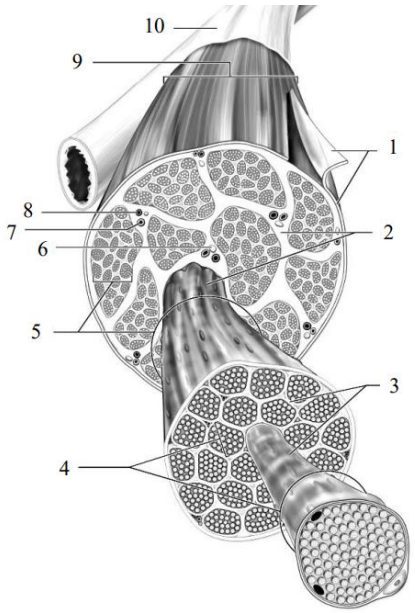
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В этой части подробнее опишем предмет исследования: анатомию сгибателей кисти, а также метод снятия данных (ВП пЭМГ).

Несмотря на то, что в исследовании мы не строим новую модель, а используем готовую биомеханическую модель мышц кисти и предплечья, важно понимать ключевые особенности строения и функционирования мышц. В дальнейшем это поможет соотнести объекты модели с реальными мышцами и правильно интерпретировать отличие результатов от экспериментальных данных.

Скелетная мышца состоит из мышечных волокон, объединенных в пучки (они же «двигательные единицы») и окруженных соединительными оболочками (рисунок 1.1):

- эндомизием: оболочка отдельных волокон
- перимизием: оболочка каждой двигательной единицы
- эпимизием: оболочка всей мышцы [18] [19].



1 – эпимизий; 2 – перемизий; 3 – эндомизий; 4 – мышечные волокна; 5 – пучки мышиных волокон; 6 – нерв; 7 – вена; 8 – артерия; 9 – скелетная мышца; 10 – глубокая фасция

Рисунок 1.1 – Строение скелетной мышцы [21]

Именно эти оболочки обеспечивают целостность ткани мышц и участвуют в передачи усилия.

Внутри мышечного волокна расположены миофибриллы, разделенные на саркомеры – основные сократительные единицы. Их сокращение происходит за счет образования актино-миозиновых мостиков между нитями белков (актинового и миозинового), благодаря чему реализуется механизм скольжения [18]. При растяжении мышцы появляется пассивное сопротивление, стремящееся вернуть мышцу в ее нормальное состояние. Это обеспечивает белок титин, соединяющий Z-диск саркомера с миозиновой нитью [20].

Как уже было сказано, каждая мышца управляется через двигательные единицы (один мотонейрон) и мышечные волокна, которые он активирует. Все волокна, входящие в одну двигательную единицу, активируются одновременно по прохождении через них нервного импульса [18]. Мелкие двигательные единицы обеспечивают тонкую и точную моторику (такие, например, в мышцах кисти и лица) (рисунок 1.2). Такие мышцы в биомеханических моделях иногда представляются в виде нескольких сегментов для того, чтобы учесть обособленное сокращение их двигательных единиц.

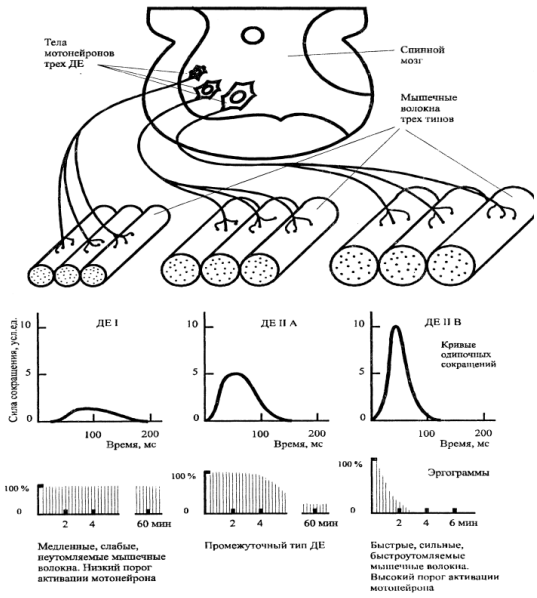


Рисунок 1.2 – Двигательные единицы и их типы [22]

Кроме того, задача декомпозиции пЭМГ-сигнала на отдельные двигательные единицы сейчас активно изучается. Такие исследования помогают обособить активации отдельных мышц и отдельных двигательных единиц [20]. При дальнейшей работе по данному исследованию стоит учитывать наработки по данной теме для более точного результата.

Движение кисти и пальцев задается, в первую очередь мышцами предплечья. Основные волокна этих мышц располагаются на предплечье, а к кисти тянутся сухожилиями (рисунок 1.3).

Рисунок 1.3 – Строение длинной ладонной мышцы [23]

Мышцы предплечья делятся на сгибатели и разгибатели. Причем активность разгибателей наблюдается также и при сгибании пальцев, так как появляется пассивная сила мышцы (за счет удлинения волокон относительно нормального положения), которая стремится вернуть их в нейтральное состояние [24]. И наоборот – активность сгибателей присутствует при

разгибании. У всех мышц в нашем теле есть соответствующие «противоположные мышцы», называющиеся антагонистами.

Некоторые из мышц предплечья участвуют в движении всей кисти, некоторые лишь в движении отдельных пальцев. Расположение этих мышц важно учитывать, как при построении модели, так и при снятии экспериментальных данных для правильного расположения датчиков (рисунок 1.4).

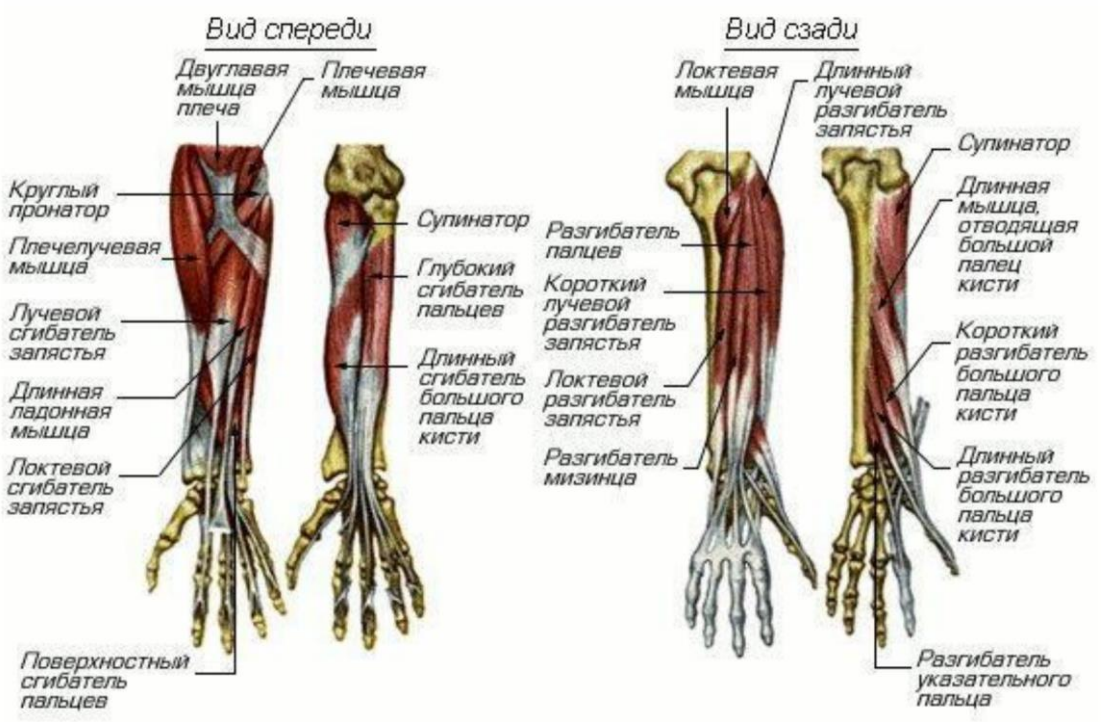


Рисунок 1.4 – Сгибатели и разгибатели предплечья

Метод снятия данных – поверхностная электромиография [25]. Этот метод позволяет неинвазивно регистрировать суммарную биоэлектрическую активность мышц, причем как в состоянии покоя, так и во время выполнения движения. Для снятия сигнала используются накожные электроды, устанавливающихся строго над брюшком мышцы, в области максимальной электрической активности (рисунок 1.5).

Рисунок 1.5 – Структурные элементы мышцы

Для этого предварительно по протоколу определяется положение брюшка мышцы при её напряжении, затем очищается кожа и фиксируется датчик (рисунок 1.6). Получаемые таким образом сигналы являются информацией о степени активации конкретных мышц.

Рисунок 1.6 – Пример крепления датчиков пЭМГ-сигнала

2 СТРУКТУРА ИССЛЕДОВАТЕЛЬСКОГО ПРОГРАММНОГО КОМПЛЕКСА

Во время анализа литературы и существующих подходов к моделированию движений кисти на основе анализа пЭМГ-данных с мышц предплечья были выделены два наиболее распространенных направления:

1. **Классификация жестов** по исходному сигналу пЭМГ с 8 или 12 каналов. Такие работы строят модели, определяющие категорию движения (например, сжатие кулака, отведение пальцев и т. д.). Однако не восстанавливают биомеханику движения.

2. **Непрерывное моделирование угловых координат суставов** по пЭМГ. Таких исследований гораздо меньше, и они ограничены 10 суставами.

Оба типа исследований основаны на использовании моделей машинного обучения. При этом построение моделей машинного обучения осуществляется без учета анатомической и биомеханической структуры кисти и рассматривают ЭМГ-активность как абстрактный входной сигнал, не связанный напрямую с конкретным расположением мышц, их взаимодействия и точек крепления.

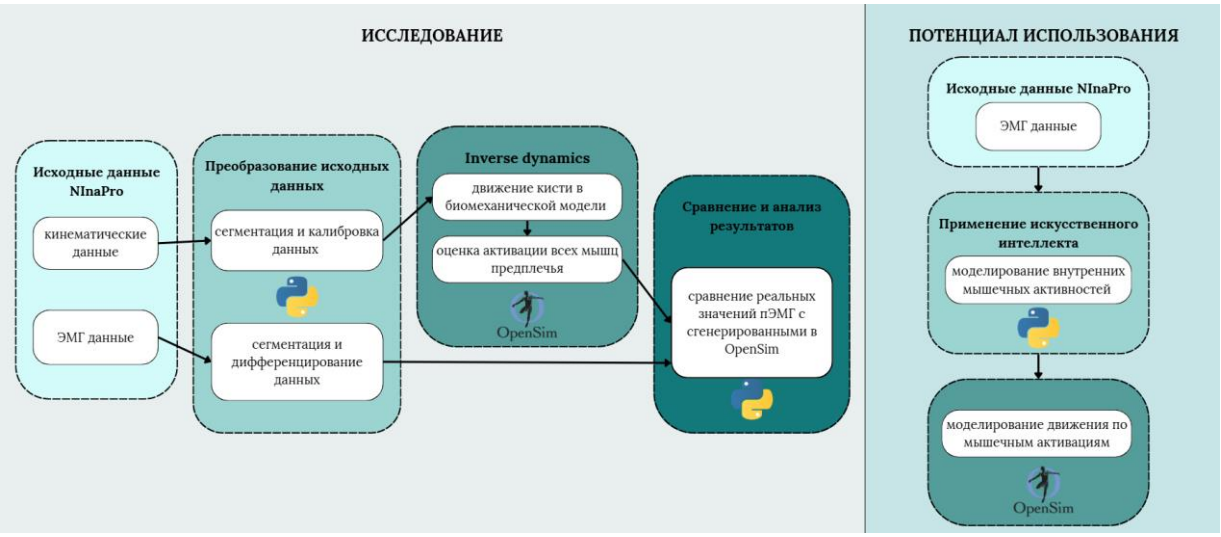


Рисунок 2.1 – Описание общей структуры программного комплекса

В настоящей работе предложен иной подход, отраженный на схеме (рисунок 2.1), сочетающий биомеханическое моделирование в среде OpenSim и использование данных из открытой базы NinaPro DB1:

1. **Исходные данные** – углы сгибания суставов пальцев, полученные с сенсорной перчатки CyberGlove II. Получены из базы данных NinaPro и преобразованы в формат, совместимый с OpenSim.

2. Угловые данные использованы для **воссоздания движения кисти** в биомеханической модели руки. Сохранен в файл движения (.mot)

3. Из сформированного файла движения при помощи встроенного инструмента OpenSim – **Static Optimization** получили оценки **активации всех мышц** предплечья.

4. Эти **активации сравнивались с реальными пЭМГ-данными**, также полученными из базы данных NinaPro для 8 каналов.

Таким образом, данный подход, в отличие от классических, позволяет моделировать не категорию жеста, и даже не суставные углы напрямую, а внутреннее мышечное состояние.

Отметим, что структура исследования позволяет в дальнейшем использовать рассчитанные активации всех мышц как целевые входные данные для обучения модели машинного обучения. Вход – реальные ЭМГ-активации с 8 датчиков. Выход – активации всех мышц модели, которые могут быть поданы обратно в OpenSim для генерации движения.

3 ОБРАБОТКА ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

3.1 Описание базы экспериментальных данных

Основным источником данных для исследования стала база данных NinaPro DB1. Она широко используется для анализа движений верхних конечностей и управления протезами для классификации «жестов», как было написано ранее. [33-34]

Пример файла с данными NinaPro представлены на графике (рисунок 3.1):

- на верхнем графике: данные, снятые с 8ми датчиков с электродами. На протяжении записей всех жестов и всех повторений;
- на нижнем: параллельно с данными пЭМГ в файл записаны номер жеста (обозначено синим цветом) из группы жестов, номер повторения конкретного жеста (оранжевым цветом);
- также параллельно с этими записями записаны 22 значения углов суставов. Подробнее будет описано ниже.

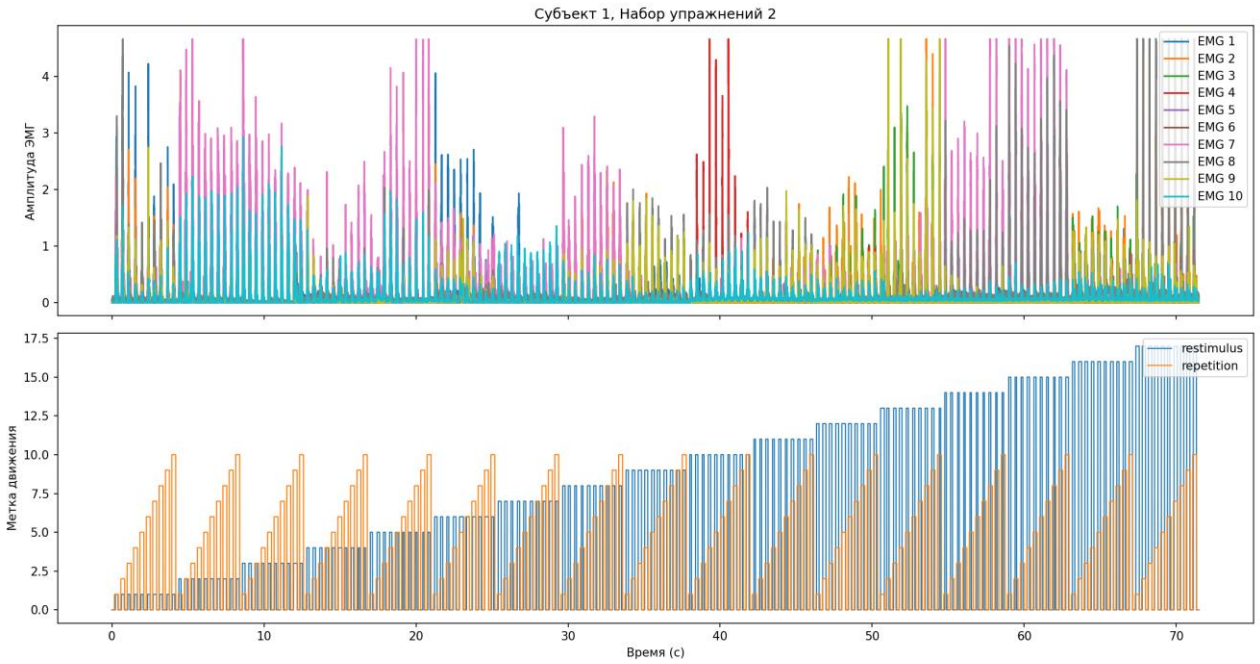


Рисунок 3.1 – пЭМГ-сигналы всей группы жестов (до сегментации)

NinaPro DB1 была выбрана в качестве источника данных по следующим причинам:

- включает в себя данные по многим испытуемым с указанием их характеристик (возраст, пол, вес);
- имеет одинаковый формат данных (в том числе для других групп испытуемых, например, для ампутантов). Что может облегчить дальнейшие исследования с применением написанного в этой работе программного кода;
- полный набор жестов для каждого испытуемого (например, в БД с ампутантими неполный набор данных, из-за невозможности записать полностью из-за возникавших болевых ощущений субъектов);
- помимо записей ЭМГ сигнала также имеет запись с перчатки CyberGlove. Что дает нам более подробные данные и позволяет решать не только задачу классификации жестов;
- При снятии данных этой БД все сбоев датчиков не наблюдалось (как например в DB2. В ней на всех субъектах данные с одного из датчиков перчатки выходит за область допустимых значений (рисунок 3.2));

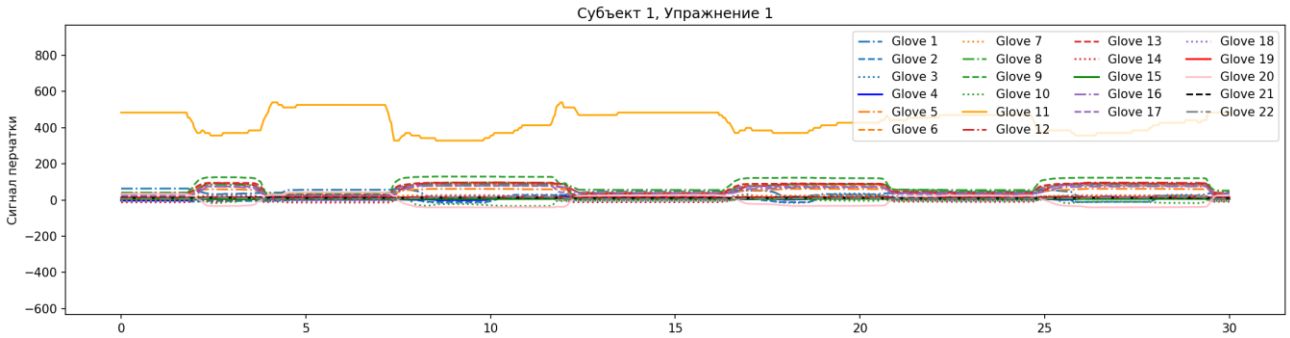


Рисунок 3.2 – Пример значений датчиков CyberGlove II с дефектным датчиком

База включает в себя 52 «жеста», которые разделены на 3 группы (рисунок 3.3):

- 12 жестов для движения отдельных пальцев (сгибание и разгибание, отведение, приведение большого пальца)

- 17 жестов, в которых изменяется положение 2 и более пальцев, либо всей кисти
- 23 функциональных жестов, имитирующих повседневную деятельность (взятие ножа, стакана, бутылки или ручки)

Эти группы жестов были собраны по функциональному принципу, основанному на ADL (Activities of Daily Living - рутинной активности человека), классификации типов движений кисти и запястья, а также на реабилитационных и робототехнических таксономиях.

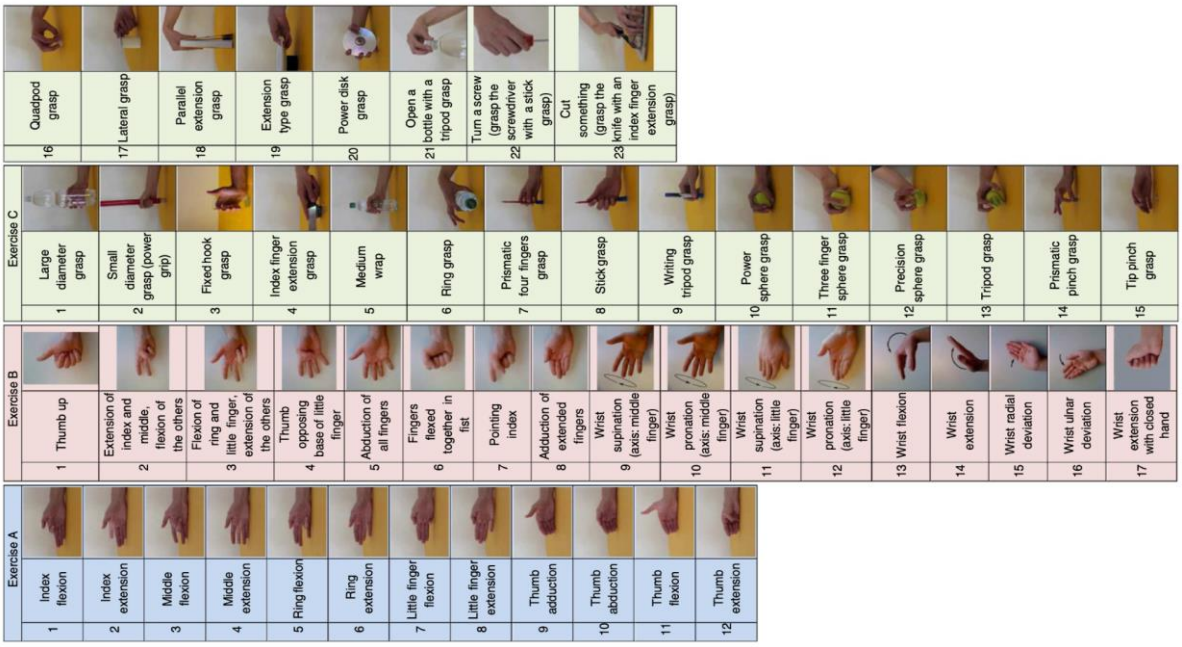


Рисунок 3.3 – Примеры жестов из базы данных NinaPro [34]

У всех испытуемых был один и тот же набор жестов. Жесты одной группы записывались друг за другом по 10 повторений подряд.

Одновременно с записью данных с электродов снимались данные с датчиковой перчатки CyberGlove II. Эта перчатка предназначена для регистрации угловых положений суставов пальцев руки и угла сгибания и отведения самого запястья. CyberGlove II оснащена тензодатчиками, позволяющими измерять изменение соответствующих углов. Изображение перчатки и схематичное расположение датчиков представлено на изображении (рисунок 3.4). Данные углов с перчатки синхронизированы с ЭМГ-сигналов с электродов.

Рисунок 3.4 – Перчатка CyberGlove II на руке участника эксперимента (фотография), схема устройства перчатки и расположения датчиков. [33-34]

3.2 Структура и обработка данных

Зарегистрированные данные хранятся в формате. mat (MATLAB). Каждый файл соответствует конкретному субъекту, включает в себя всю группу последовательно выполненных жестов группы с повторениями. В файлах в каждый момент времени соответственно записаны значения: restimulus (номер жеста в группе), repetition (номер повторения жеста конкретным субъектом), значения с ЭМГ датчиков и 22 значения перчатки CyberGlove II.

В настоящей работе был реализован код на Python для:

- визуализации полных исходных данных (Приложение А);
- создания отдельных сегментов данных по каждому конкретному жесту, каждому повторению (Приложение Б);
- сохранения углов суставов в отдельный CSV-файл для дальнейшего использования в OpenSim модели (Приложение Б);
- аппроксимации эмг сигнала для дальнейшего сравнения и численной оценки результатов (Приложение В).

Как уже было сказано выше было применена аппроксимация ЭМГ сигнала. Это необходимо также для устранения шумов и повышения интерпретируемости сигнала. Для этих целей были рассмотрены следующие методы [35]:

1. Скользящее среднее (moving average) (3.1):

$$y_i = \frac{1}{N}\sum_{j=i-N\div 2}^{i+N\div 2} x_j, \quad (3.1)$$

где y_i – сглаженное значение сигнала в точке с индексом i;

x_i – исходное значение сигнала в точке с индексом j;

N – длина окна усреднения (количество точек в окне).

2. Фильтр Хилберта (Hilbert transform) (3.2):

$$\hat{x}(t) = \frac{1}{\pi} \cdot P.V. \int_{-\infty}^{\infty} \frac{x(\tau)}{t-\tau} d\tau, \quad (3.2)$$

где $\hat{x}(t)$ – преобразованный сигнал;

$x(\tau)$ – исходный сигнал;

P.V. – главное значение интеграла (principal value).

3. Низкочастотная фильтрация с использованием фильтра Баттерворда (Hilbert transform) (3.3):

$$\left|\left(H(f)\right)\right| = \frac{1}{\sqrt{1+\left(\frac{f}{f_c}\right)^{2n}}}, \quad (3.3)$$

где f – текущая частота сигнала;

f_c - Частота среза (например, 10Гц);

n – порядок фильтра.

4. NK-фильтрация – метод с ядром типа «bozxen» (реализация при помощи встроенного метода nk.signal_smooth) (3.4)

$$y(t) = \sum_{\tau} x\,(t-\tau) \cdot k(\tau), \quad (3.4)$$

где t – текущий момент времени (или индекс текущей точки сигнала);

τ – смещение в пределах ширины ядра фильтра;

$x(t-\tau)$ – значение исходного сигнала в точке, смещенного на τ относительно t;

$k(\tau)$ – значение ядра фильтра в позиции τ.

После тестирования всех методов было решено использовать метод скользящего среднего. Он наиболее устойчив к шумам, сохраняет форму сигнала, что особенно важно при интерпретации мышечной активности (рисунок 3.5).

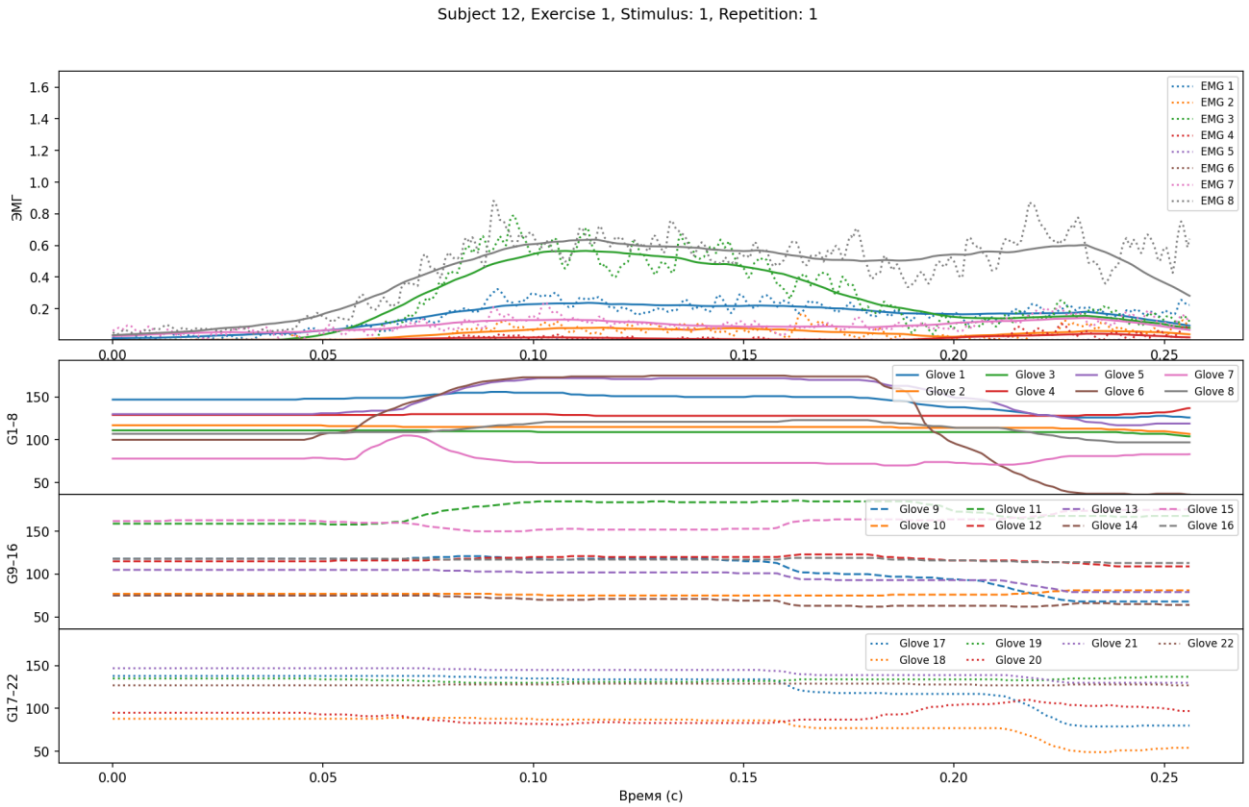


Рисунок 3.5 – ЭМГ-сигнал отдельно сегментированного жеста «сгибание указательного пальца» (пунктиром на верхнем графике – исходные данные. Сплошным – аппроксимированные). На трех нижних – значения углов перчаток

Стоит пояснить, что на графике представлены данные для записи пЭМГ с датчиков и углов суставов с перчатки (разделенных на три отдельных графика, для улучшения восприятия) для одного человека.

4 БИОМЕХАНИКА В OPENSIM

4.1 Исследование существующих моделей кисти в OpenSim

OpenSim – это открытое ПО для моделирования биомеханики. Оно позволяет симулировать и анализировать скелетно-мышечную систему человека. OpenSim широко используется в исследованиях связанных со всеми областями применения биомеханики: ортопедия, реабилитация, спортивная медицина. К примерам задач биомеханики, решаемых в OpenSim можно отнести анализ кинематики, оценку усилий мышц и напряжений в суставах, разработка персонализированных моделей суставов, создание персонализированных моделей походки человека и их анализ. [25-28]

В последних версиях OpenSim появилось Python API. Именно это API использовалось для исследования. Оно облегчает взаимодействие с модель, использование других библиотек, а также позволяет автоматизировать изменение параметров моделей. [29]

В учебных целях для работы с API предоставляется несколько моделей. В том числе базовая модель руки, реализованная в виде двузвенного маятника (рисунок 4.1). Однако данная модель абсолютно не соответствует анатомической структуре кисти и использовалась только для ознакомления взаимодействия с API.

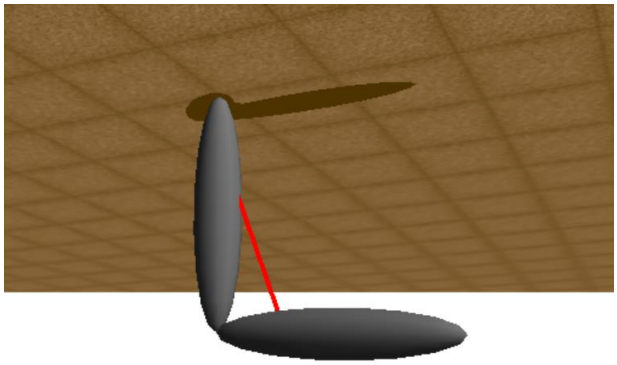


Рисунок 4.1 – Базовая демонстрационная модель руки (двузвенного маятника) в OpenSim

Также для OpenSim разработаны более подробные модели руки человека. Были найдены и проанализированы несколько анатомически корректных моделей (рисунок 4.2). Это были следующие модели:

1. Базовая костная модель верхней конечности. (Эта модель не подходит нам, так как в ней не получится смоделировать мышечную активность)

2. Модель с полным набором мышц. Однако в данной модели суставы кисти и пальцев были зафиксированы и мне не удалось вынести их в изменяемые параметры. Данная модель предназначается скорее для имитации таких движений, как сгибание и разгибание локтя, приведение и отведение руки в плечевом суставе.

3. Модель с объединенными сгибателями пальцев в единый мышечный элемент. (предоставлена Юровой А.С.) Эта модель была получена в результате совмещения нескольких других моделей кисти. Данная модель также не подойдет для нашего исследования, поскольку она симулирует движение при нестандартном наборе мышц предплечья, совмещенных в одну.

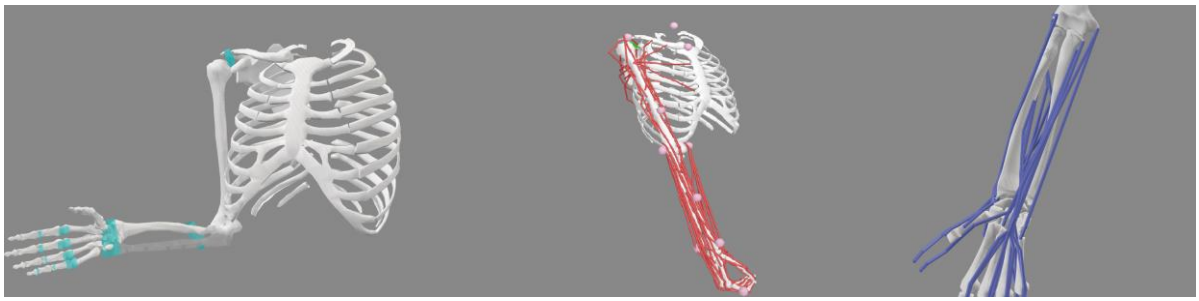


Рисунок 4.2 – Возможные модели руки для исследования

В результате поиска была подобрана наиболее подходящая модель с хорошей детализацией анатомии кисти (рисунок 4.3). В ней все кости предплечья и кисти представлены подвижными компонентами.

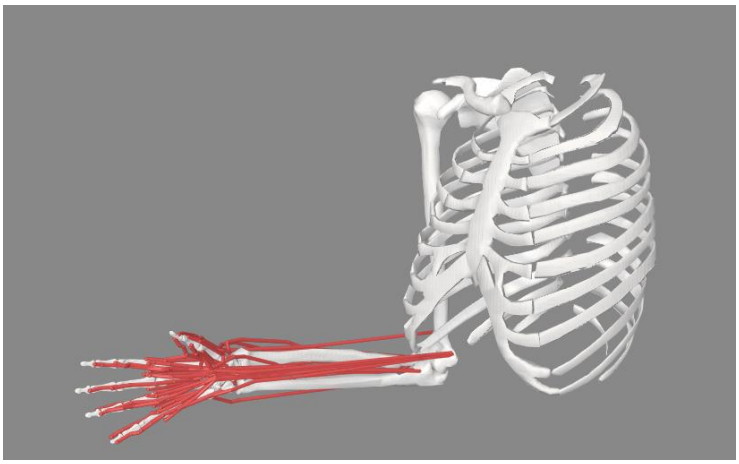


Рисунок 4.3 - Модель руки с подвижными суставами кисти, используемая в исследовании

В выбранной модели есть особенность, которую в дальнейшем мы будем учитывать при анализе полученных результатов. В ней Musculus extensor digitorum (разгибатель пальцев) представлен четырьмя отдельными объектами мышц (рисунок 4.4). Это связано с тем, что разные ее двигательные единицы соединены с разными пальцами и соответственно активируются по отдельности. Подобное поведение невозможно иначе отобразить в OpenSim модели. Также датчиками, которые использовались при снятии экспериментальных данных невозможно снять активность отдельных двигательных единиц.

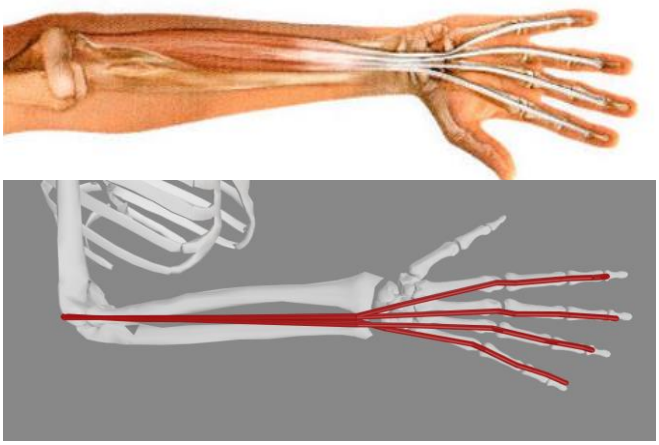


Рисунок 4.4 – Строение Musculus extensor digitorum (разгибатель пальцев) в реальности и в модели

4.2 Модель мышцы в OpenSim

Как было сказано выше, мы не можем представить в объекте одной мышцы отдельные двигательные единицы, активирующиеся отдельно друг от друга. Рассмотрим модели мышц.

OpenSim предоставляет несколько моделей, описывающих мышцы. Самая базовая из них – модель Хилла. Эта модель включает в себя контрактивный элемент (CE), пассивный эластичный элемент и последовательный сократительный элемент (рисунок 4.5). Активная сила формируется только в CE и зависит от длины мышцы и скорости ее сокращения (в соответствии с формулой 4.1):

$$\begin{aligned} F_{total} &= F_{CE} + F_{PE} \\ F_{CE} &= f_A \cdot f_L(l_{CE}) \cdot f_V(v_{CE}) \\ F_{PE} &= f_{passive}(l_{CE}), \end{aligned} \tag{4.1}$$

где F_{total} – суммарная сила, генерируемая мышцей;

F_{CE} – сила, создаваемая сократительным элементом (активная сила);

F_{PE} – сила, создаваемая пассивным (эластичным) элементом;

f_A – уровень активации мышцы, $0 \leq \mathrm{f_A} \leq 1$;

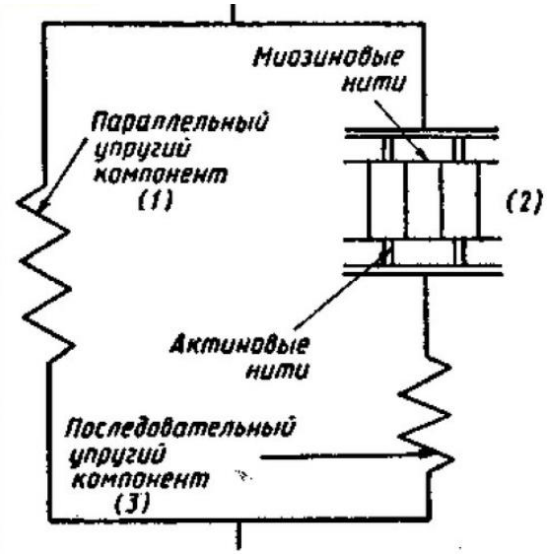
$f_L(l_{CE})$ – функция зависимости силы от длины сократительного элемента;

$f_v(v_{CE})$ - функция зависимости силы от скорости изменения длины;

$f_{passive}(l_{CE})$ – функция пассивной эластичности мышцы;

l_{CE} – длина сократительного элемента (мышечного волокна);

v_{CE} – скорость изменения длины мышечного волокна (сокращение или растяжение).



(1) - соединительные образования (оболочка мышечных волокон, фасция и тд), (2) - Сократительный элемент. Актино-миозиновые мостики, (3) - сухожилие и миофибриллы

Рисунок 4.5 – Модель Хилла

Модель мышцы Милларда (Millard2012EquilibriumMuscle) это усовершенствованная модель Хилла. Эта модель широко используется в биомеханике для симуляции мышечного сокращения. Она учитывает как активные, так и пассивные компоненты силы, а также влияние скорости сокращения и угла пеннации (рисунок 4.6). [30]

Угол пеннации – это угол между направлением мышечных волокон и направлением силы, передаваемой сухожилию. Он влияет на эффективную силу, передаваемую сухожилию, через cos(φ). При увеличении угла пеннации эффективная сила уменьшается.

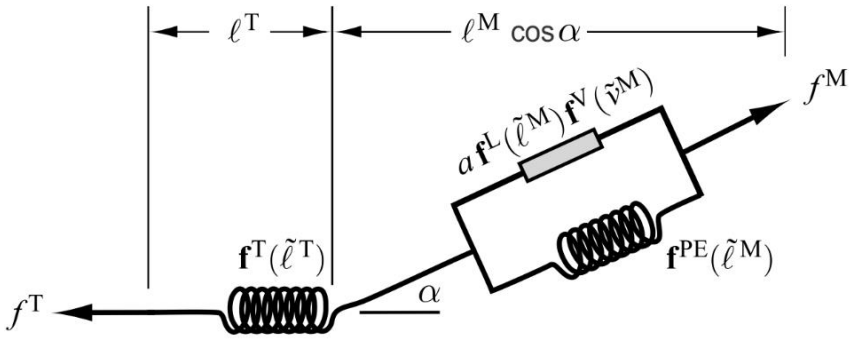


Рисунок 4.6 – Структура модели Милларда [31]

Именно эта модель используется в актуальных моделях мышц OpenSim,

в том числе и в используемой для исследования модели кисти руки и предплечья. Ниже опишем компоненты силы, формируемые моделью Милларда:

1. **Активный компонент силы [31]**

Этот компонент зависит от степени активации мышцы, её длины и скорости сокращения. Он моделируется с использованием следующих кривых:

а) **Кривая сила-длина (active force-length curve)**: описывает, как сила изменяется в зависимости от длины мышечного волокна (рисунок 4.7).

б) **Кривая сила-скорость (force-velocity curve)**: описывает зависимость силы от скорости сокращения или растяжения мышцы (рисунок 4.7).

2. **Пассивный компонент силы [31]**

Этот компонент возникает из-за эластичных свойств мышечных тканей и сухожилий, даже без активации мышцы. Он моделируется с использованием:

а) **Кривой пассивная сила-длина (passive force-length curve)**: описывает силу, возникающую при растяжении мышцы сверх её оптимальной длины (рисунок 4.7)

б) **Кривой сила-длина сухожилия (tendon force-length curve)**: описывает эластичность сухожилия при различных его длинах (рисунок 4.7).

Общая сила активации мышцы рассчитывается в соответствии с формулой 4.2:

$$\begin{gathered} F_{total} = F_{ISO} \cdot (F_{CE} + F_{PE} + F_V) \cdot cos(\varphi) \\ F_{CE} = f_A \cdot f_L(l_{CE}) \cdot f_V(v_{CE}) \\ F_{PE} = f_{PE}(l_{CE}) \\ F_V = \beta \cdot v_{CE}, \end{gathered} \tag{4.2}$$

где F_{total} – суммарная сила, генерируемая мышцей;

F_{CE} – сила, создаваемая сократительным элементом (активная сила);

F_{PE} – сила, создаваемая пассивно (из-за растяжения элемента);

F_V – сила вязкого сопротивления;

F_{ISO} – максимальная изометрическая сила мышцы;

f_A – уровень активации мышцы, $0 \leq f_A \leq 1$;

$f_L(l_{CE})$ – функция зависимости силы от длины сократительного элемента;

$f_v(v_{CE})$ – функция зависимости силы от скорости изменения длины;

$f_{PE}(l_{CE})$ – функция пассивной эластичности мышцы;

l_{CE} – длина сократительного элемента (мышечного волокна);

v_{CE} – скорость изменения длины мышечного волокна (сокращение или растяжение);

φ – *угол пеннации;*

β – *коэффициент вязкости.*

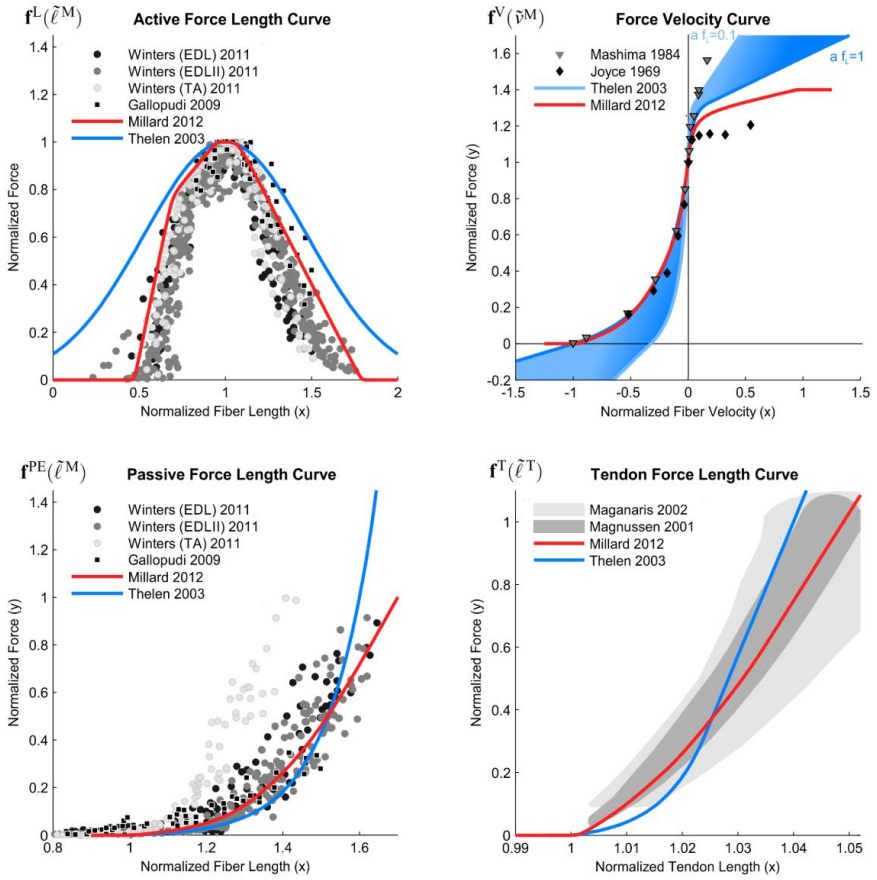


Рисунок 4.7 - Сравнительные графики компонент силы для разных моделей (Милларда и более ранней модели – Телена) [32]

В рамках работы с выбранной моделью были подробно изучены её основные компоненты, включая анатомическое строение костных сегментов и расположение мышц. Также был разработан программный код, позволяющий управлять мышечной активностью – отключать или изменять её параметры – а также модифицировать углы в суставах (Приложение Е).

5 ИНТЕГРАЦИЯ ДАННЫХ В МОДЕЛЬ КИСТИ OPENSIM

На этапе интеграции экспериментальных данных в биомеханическую модель в OpenSim была решена задача соответствия между измеренными углами суставов и переменными модели.

Данные перчатки представляют собой измерения углов сгибания (flexion) суставов всех пяти пальцев, углы сгибания (flexion) и отклонения (deviation) кисти, а также углы отведения/приведения (abduction) большого пальца и угол между пальцами (задан тоже как abduction).

Однако из-за существенных расхождений в определении нулевого положения (со значением в 0 градусов) углов между моделью и перчаткой необходимо было откалибровать эти углы. В частности, углы отведения (abduction) были зафиксированы на постоянных значениях для всех пальцев, кроме большого. Это было сделано по двум причинам:

- во-первых, данные углы мало варьируются при выполнении большинства жестов;
- во-вторых, различие в анатомической интерпретации и параметризации этих углов между моделью и устройством делают их точную передачу затруднительной: некоторые наборы значений углов между 2-5 пальцами можно интерпретировать двумя и более способами, пример двух различных положений при одних и тех же значениях углов между пальцами представлены на рисунке (рисунок 5.1).

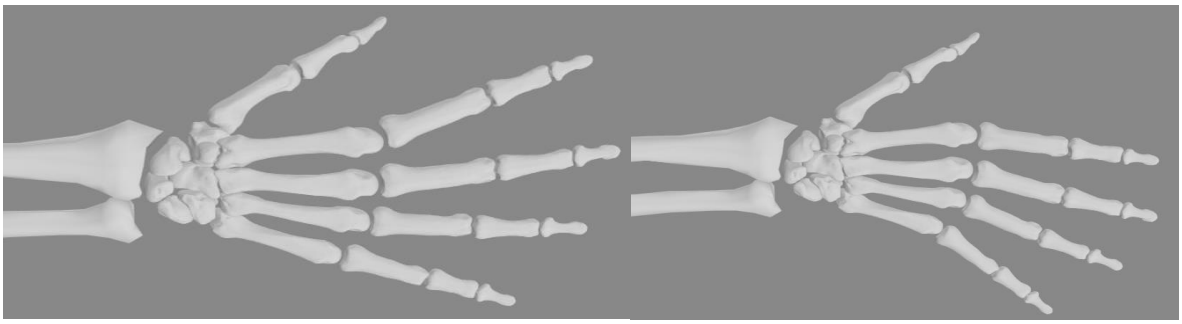


Рисунок 5.1 – разные положения кисти с одинаковыми углами между 2-4 пальцами

Полный перечень соответствий между параметрами модели и данными перчатки, кодом для моделирования жестов и визуализации движений приведён приложении (Приложение Г).

После обработки и калибровки углы были переданы в модель, на основе которых формировались состояния (states) руки, визуализируемые с помощью анимации движения. На снимке экрана представлено положение суставов и мышц для движения «взятие стакана» из третьей группы жестов БД NinaPro (рисунок 5.2). Красные линии – мышечные структуры, красные сферы на изображении – точки крепления мышц.

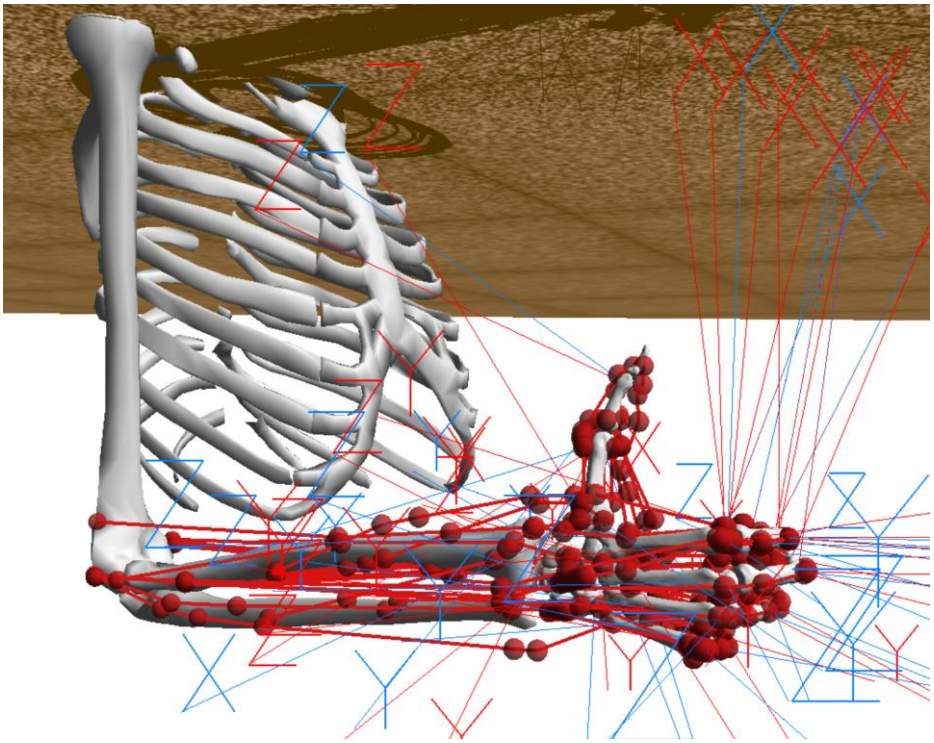


Рисунок 5.2 – Снимок экрана из анимации движения руки на основе экспериментальных данных (на примере движения «взятие стакана»)

Также был проведён анализ мышечной анатомии и структуры модели. В биомеханической модели используется 43 мышцы, в то время как в экспериментальных данных с ЭМГ задействовано только 8 каналов. Соответствие между мышцами, на брюшко которых ориентируются сенсоры, и мышцами в модели представлено в таблице 5.1:

Таблица 5.1 - Соотнесение анатомических названий мышц с названиями в модели

Название мышцы (на которой находится датчик)	**Имя в модели OpenSim**
m. brachioradialis	(отсутствует в модели)
m. flexor carpi radialis	FCR
m. palmaris longus	PL
m. flexor carpi ulnaris	FCU
m. extensor carpi ulnaris	ECU
m. extensor digitorum	EDCL, EDCR, EDCM, EDCI
m. extensor carpi radialis brevis	ECRB
m. extensor carpi radialis longus	ECRL

Extensor digitorum (разгибатель пальцев) в модели представлен в виде четырех отдельных мышц, о чем мы говорили ранее. В модели также отсутствует объект модели мышцы, соответствующий brachioradialis (плечелучевая мышца). Это связано с тем, что она участвует преимущественно в сгибании локтевого сустава и не имеет значительного вклада в движение кисти.

6 РАССЧЕТ МЫШЕЧНОЙ АКТИВНОСТИ ИЗ УГЛОВЫХ ДАННЫХ

Перед тем как переходить к расчетам уточним разницу между пассивной и активной составляющими мышечной силы:

- пассивная сила возникает без нейронной активации - при растяжении мышечно-сухожильного комплекса и появляется за счет его упругих свойств. (В модели Миларда управляющие обозначались как passive-force-length curve и tendon-force-length-curve). Эта сила фактически стремится вернуть биомеханическую систему в положения покоя. Такая сила может возникать: при растяжении расслабленной мышцы (если мы пассивно сгибаем сустав) и в мышцах-антагонистах во время активной работы мышц-агонистов;
- активная сила возникает при возбуждении мышцы мотонейроном, за счет сокращения мышечных волокон и зависит от уровня активации (частоты импульсов от мотонейронов), длины и скорости сокращения мышцы. Такая сила возникает только в активируемых мышцах. (В модели Милларда соответственно представлена совокупностью управляющих кривых active-force-length-curve и force-velocity-curve).

Рассмотрим также суммарную силу, генерирующуюся мышечно-сухожильным комплексом (без учета составляющей скорости) в зависимости от отношения длины мышечного волокна к его **оптимальной длине** (l/l_0) (рисунок 6.1). Оптимальная длина – длина мышцы, при которой она способна развить максимальное усилие (оптимальная длина НЕ ВСЕГДА РАВНА длине мышцы в расслабленном состоянии. Длина мышцы в расслабленном состоянии больше, чем оптимальная. Оптимальная длина достигается при частичном сокращении мышцы от ее «состояния покоя»).

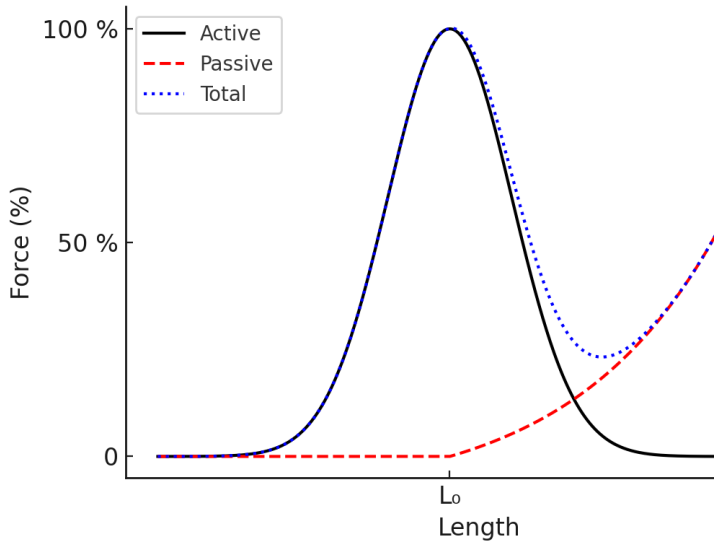


Рисунок 6.1 – график зависимости силы мышцы от ее длины относительно оптимальной длины

Рассмотрим подробнее компоненты графика:

1. Active – активная составляющая мышечной силы, генерируемая сократительным элементом (СЕ). Достигает пика при $l = l_0$.

2. Passive – пассивная сила, возникающая при растяжении мышцы и стремящаяся вернуть ее в исходное положение (например, в мышцах-антагонистах). Начинает расти при $l > l_0$. *В*ключает в себя как пассивную силу мышечной составляющей, но не пассивную силу, генерируемую растяжением сухожилия.

3. Total – суммарная сила активной и пассивной части.

Безусловно у растяжения мышечно-сухожильного комплекса есть предел, который также задается в модели. И получить большую силу только за счет пассивной составляющей силы нельзя, есть предел растяжения (который в модели выставляется отдельными параметрами). При перерастяжении мышечного-сухожильного комплекса в реальности рвутся ткани - надрыв или полный отрыв сухожилия.

6.1 Расчет пассивной составляющей мышечной силы

Положение суставов напрямую влияет на относительное удлинение мышц (за счет точек прикрепления). Соответственно при известном угловом положении суставов в конкретный момент времени можно рассчитать пассивную силу каждой мышцы.

Для расчета пассивных сил в каждый момент времени были автоматизированы (с использованием Python API) следующие действия (Приложение Д):

1. Для каждого момента времени устанавливается нужное значение углов (из ранее сегментированных файлов) в загруженную модель.
2. Пересчитывается состояние модели в соответствии с этими углами.
3. Для каждой мышцы вызывается встроенный метод computePassiveForce. (Полученная сила выражается в Ньютонах)
4. Временной ряд сохраняется в файл для построения графиков пассивной силы.

Так, например, представим график (рисунок 6.2) для мышц, длина которых изменяется при сгибании указательного пальца.

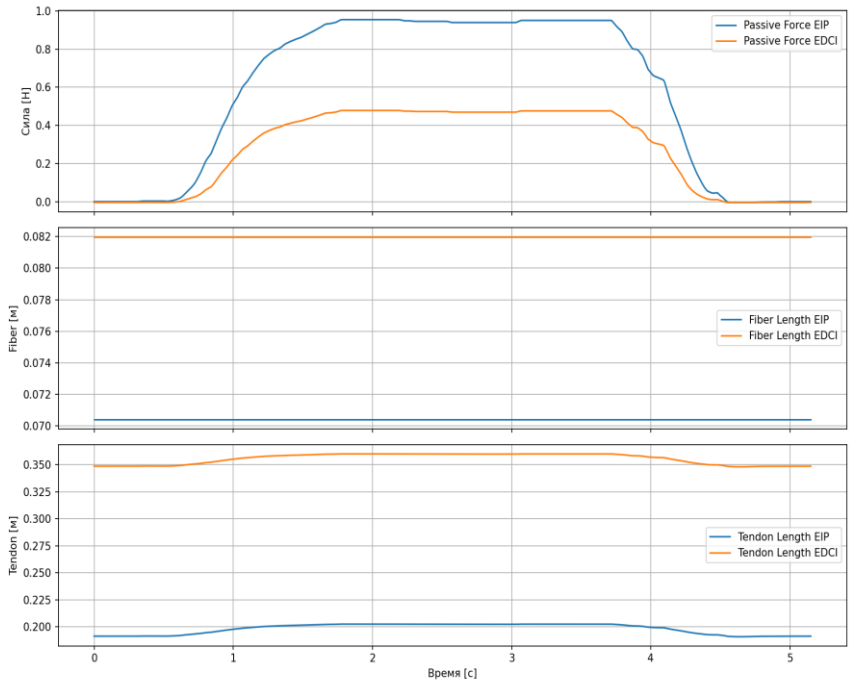


Рисунок 6.2 – Пассивная сила, длина мышц и сухожилий разгибателей указательного пальца EIP и EDCI. В случае нефиксированной длины сухожилия (на примере движения «сгибание указательного пальца»)

Расположение указанных мышц можно увидеть на рисунке (рисунок 6.3):

- EIP – Extensor Indicis Proprius (собственный разгибатель указательного пальца) короткая мышца на изображении.
- EDCI – Extensor Digitorum Communis to Index (часть двигательных единиц мышцы EDC общего разгибателя второго-пятого пальцев). Длинная мышца на изображении

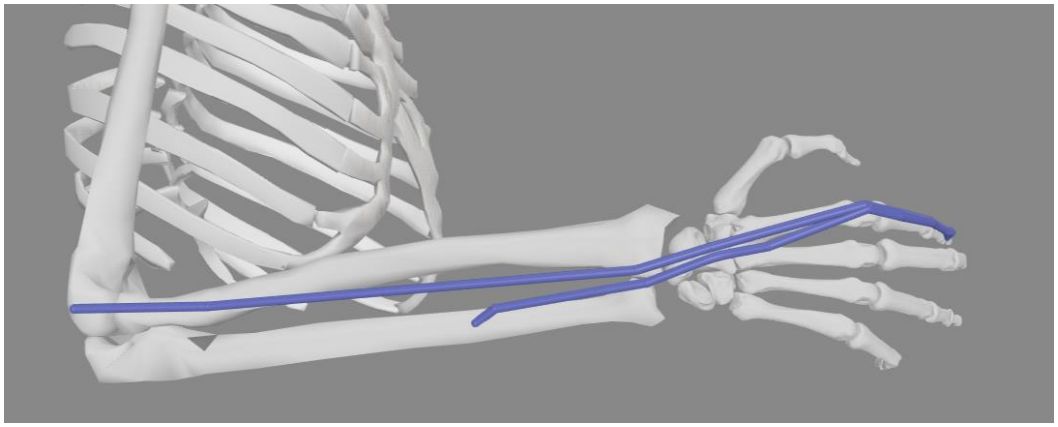


Рисунок 6.3 – расположение мышц-разгибателей указательного пальца EIP и EDCI

Эти данные позволяют оценить, какие мышцы испытывают пассивное растяжение во время жеста, как меняется пассивная сила от времени, сравнить набор сокращаемых и удлиняющихся мышц.

6.2 Обзор методов оптимизации в OpenSim

В OpenSim разработаны два встроенных метода для расчета активной составляющей мышцы: Static Optimization и Computed Muscle Control (CMC). Оба метода позволяют рассчитать активации мышц, необходимые для задаваемого движения. Оба метода основаны на инверсной динамике, соответственно решая следующие задачи [36-37]:

1. Static Optimization (SO) в каждый момент времени решает задачу минимизации функционала при том, что суммарный ответ от мышц должен соответствовать моменту, рассчитанному по инверсной динамике. Минимизируется целевая функция (6.1) при выполнении одного из условий: идеальные генераторы сил (6.2) или генераторы, ограниченные кривой сила-длина-скорость (6.3) [38].

$$J = min \sum_{m=1}^{n} (a_m)^p \tag{6.1}$$

$$\sum_{m=1}^{n} (a_m \cdot F_m^0) \cdot r_{m,j} = \tau_j \tag{6.2}$$

$$\sum_{m=1}^{n} [a_m \cdot f(F_m^0, l_m, v_m)] \cdot r_{m,j} = \tau_j, \tag{6.3}$$

где n – количество мышц в модели;

a_m – уровень активации мышцы m на каждом дискретном временном шаге;

p – степень целевой функции, чаще всего 2;

F_m^0 – его максимальная изометрическая сила;

l_m – ее длина;

v_m – скорость изменения ее длины;

$f(F_m^0, l_m, v_m)$ – кривые зависимости сила-длина-скорость;

$r_{m,j}$ – плечо момента относительно оси j-го сустава;

τ_j – обобщенная сила, действующая вокруг оси j-го сустава

Важное замечание. При использовании этого метода **длины сухожилий фиксируются неизменными** в соответствии с документацией [38].

2. Computed Muscle Control (CMC) решает динамическую задачу управления: находит такие активации, при которых модель воспроизводит заданное движение. (Похоже на обратную задачу управления, где динамика тела определяется дифференциальными уравнениями). Подробное описание работы этого метода опущено, в силу невозможности его использования (нужны дополнительные входные файлы кинематики). Однако подробно уравнения метода, а также их вывод можно изучить в документации. [38-39]

Сравнительные характеристики для SO и CMC представлены в таблице 6.2:

Таблица 6.2 – Сравнительные характеристики для методов Static Optimization и Computed Muscle Control

	Static Optimization (SO)	Computed Muscle Control (CMC)
Принцип действия	каждый кадр обрабатывается отдельно	расчет каждого кадра ориентируется на предыдущий
Учитывает скорости движения	нет	да
Учитывает внешние силы и контактные условия	нет	да
Чувствителен к параметрам модели	слабо	сильно
Выходные значения	.sto файлы с активациями мышц, активными и пассивными усилиями мышц .xml файл с управляющими сигналами	.sto файлы с активациями мышц, активными и пассивными усилиями мышц .xml файл с управляющими сигналами states.sto – положение, скорость и длина мышц

SO является более устойчивым и быстрым способом. Одновременно с

тем – менее физиологичным (так как не учитывает расчеты предыдущего временного кадра). В исследовании было принято решение использовать SO для расчета активной составляющей силы. Этот выбор обусловлен большим количеством сегментов движений для обработки, ограниченным временем на симуляцию, а также отсутствием внешних параметров CMC.

Однако в дальнейшем развитии исследования рекомендуется настроить метод CMC, поскольку его результирующие графики будут более физичными.

Подготовка данных движения: углы, полученные с CyberGlove II были сохранены в формате .mot, совместимый с OpenSim.

6.3 Корректировка модели для использования Static Optimization

Исключение растяжения сухожилий. Как уже было сказано в описании метода Static Optimization, его использование возможно только при условии фиксированной длины сухожилий (tendon). Это связано с тем, что метод основан на квазистатическом подходе и не учитывает динамическую деформацию этой составляющей мышечно-сухожильного комплекса. Поэтому в моделировании необходимо исключить влияние растяжимости сухожилий, чтобы обеспечить корректную работу алгоритма оптимизации.

В OpenSim данное ограничение реализуется путем установки специального флага для каждой мышцы, определяющего, будет ли учитываться растяжимость соответствующего сухожилия. Для этого в программном коде используется метод:

```
muscle.set_ignore_tendon_compliance(True)
```

После применения этого метода растяжение сухожилий при расчетах полностью игнорируется. Таким образом, все изменение длины мышечно-сухожильного комплекса приписывается только мышечному волокну. Подробная реализация данной настройки представлена в приложении G.

Также после применения данного флага изменятся и соответствующие графики пассивной силы, которая раньше генерировалась за счет удлинения

сухожилия (рисунок 6.2). Теперь в мышцах-антагонистах растяжение и формирование пассивной силы происходит за счет растяжения мышечного волокна (рисунок 6.4).

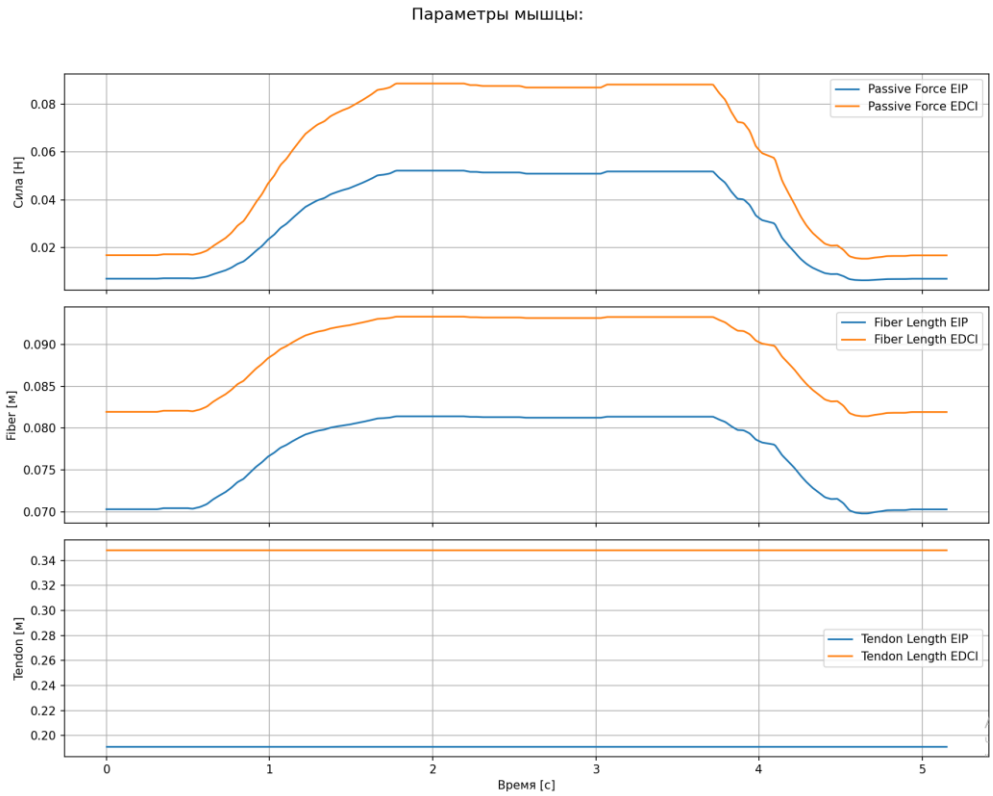


Рисунок 6.4 - Пассивная сила, длина сухожилия и мышц разгибателей указательного пальца EIP и EDCI, после фиксирования длины сухожилия (на примере движения «сгибание указательного пальца»)

Настройка мышц в соответствии с начальным положением. По умолчанию в используемой биомеханической модели исходное положение пальцев задано полностью разогнутым состоянием, при котором все межфаланговые и пястно-фаланговые суставы находятся в нейтральной (нулевой) позиции (рисунок 6.5). Такое положение анатомически возможно, но в реальности практически не встречается у здорового человека в состоянии мышечного покоя. В реальных условиях расслабленная кисть характеризуется умеренным сгибанием как в проксимальных, так и в дистальных суставах пальцев.

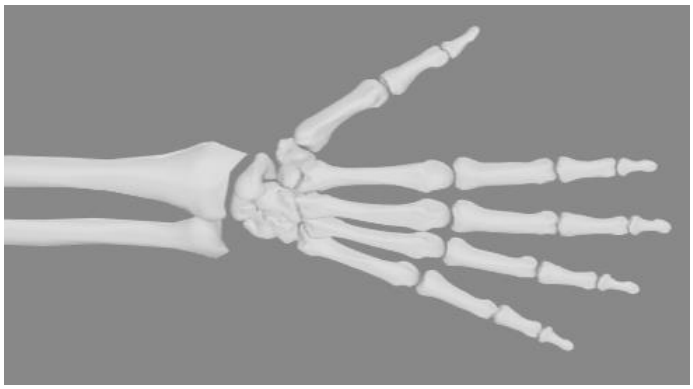


Рисунок 6.5 – Изначальное положение суставов в используемой модели

Такое расхождение между модельной и физиологической позицией состояния «покоя» приводит к некорректной инициализации параметров мышц, в частности длины мышечных волокон и сухожилий, а также их соотношения в мышечно-сухожильных комплексах. Это влияет на расчеты пассивных и активных сил и искажает результаты расчетов Static Optimization.

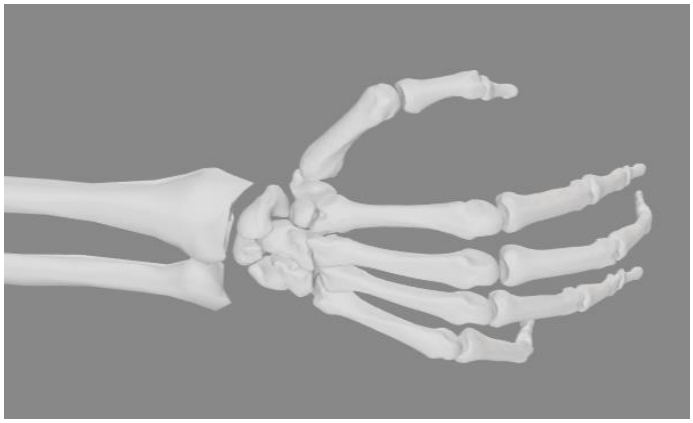


Рисунок 6.6 – Положение суставов в модели после корректировки (на примере движения «сгибание указательного пальца»)

Для более достоверного моделирования исходная поза была скорректирована с использованием данных с перчатки CyberGlove II. Углы сгибания во всей кисти были извлечены из первого кадра измерений, соответствующего естественной расслабленной позе испытуемого (рисунок 6.6).

На рисунке указана поза, взятая из первого кадра конкретного примера, однако состояние покоя перед и между движениями у человека одинаковая с незначительными отклонениями.

Полученные значения были заданы в модели как «дефолтные» (initial/default pose), после чего были пересчитаны параметры мышц (рисунок 6.7):

```
muscle.computeInitialFiberEquilibrium(state)
# Полная длина мышцы (сухожилие + волокно) для установленного положения
l_mt = muscle.getLength(state)
# Длина волокна для установленного положения
fiber_length = muscle.getFiberLength(state)
# Оптимальная длина волокна
l_opt = muscle.getOptimalFiberLength()
# сила и длина сухлжилия для установленного положения
t_length = muscle.getTendonLength(state)
t_slack_length = muscle.getTendonSlackLength()
# Исходная дефолтная длина
def_len = muscle.getDefaultFiberLength()
# угол пеннации
pennation_angle = muscle.getPennationAngleAtOptimalFiberLength(()
# минимальная возможное значение активации мышцы
min_activation = muscle.getMinControl()
muscle.setDefaultActivation(min_activation)

# новая дефолтная длина волокна в соответствии с установленным положением
fiber_length_new = (l_mt - t_slack_length)/math.cos(pennation_angle)
# установка новых значений параметров
muscle.setDefaultFiberLength(fiber_length_new)
muscle.setFiberLength(state, fiber_length_new)
muscle.setOptimalFiberLength(fiber_length*0.95)
```

Рисунок 6.7 – отрывок кода для пересчета базовых параметров мышц

Это обеспечивает соответствие модели реальному биомеханическому состоянию кисти и позволяет корректно инициализировать параметры мышц до выполнения последующих расчетов. Полный программный код можно изучить в приложении. (Приложение Е)

Для анализа достоверности получившихся длин был произведен сравнительный анализ соотношений длин сухожилий и мышечных волокон в модели с этими же значениями в реальности. Сравнительную таблицу можно увидеть в приложении (Приложение Ж).

Корректировка углов суставов. При первых попытках рассчитать активную составляющую мышечную силу при помощи SO возникли ошибки (нарушение ограничений constraint vioation и time out) связанные с тем, что модель не персонализирована. Как следствие:

- мышечные параметры обобщенные, могут отличаться от конкретного человека;

- точки крепление мышц могут быть не совсем точными.

Также есть и другие причины, порождающие вышеописанные проблемы:

- объем данных (слишком длинные по временным кадрам записи жестов);
- отсутствие компенсирующих сил;
- отсутствие резерва для компенсации ошибок модели.

Для решения и компенсации этих и других эффектов было приняты следующие меры:

- уменьшение частоты записи сегментированных отрезков (меньшее количество записей и, как следствие, меньшая нагрузка на оптимизатор для того же жеста);
- использование актуаторов.

В OpenSim актуаторы – это специальные элементы модели, которые создают силы или моменты и используются в следующих случаях:

- для управления движением;
- для компенсации ошибок;
- для поддержки мышц, когда те не справляются с задачей.

Важно понимать, что в идеале все движение должно обеспечиваться только мышцами, а актуаторы использоваться не должны. Однако в случае, если сила актуатора незначительна, то его эффект не заменит мышцы, а лишь компенсирует остатки, уравновешивает систему уравнений, благодаря чему система становится решаемой.

Для добавления актуаторов и выполнения оптимизации был написан программный код (Приложение Е). Для каждого сустава (с незафиксированным положением) добавлен CoordinateActuator с заданными ключевыми параметрами:

```
    actuator.setOptimalForce(opt_force)
    actuator.setMaxControl(1)
actuator.setMinControl(-1)
```

- optimal_force (минимальный/максимальный момент) – множитель управляющего сигнала control;
- Max/MinControl – ограничивают рамки допустимых усилий [40].

Актуаторы должны создавать малое усилие для стабилизации положения (по сравнению с мышцами, при этом система решения движений должна остаться стабильна:

- значение constraint violation <1e-6 (система имеет решение с допустимой погрешностью)
- нет ошибок оптимизации и solver не падает с ошибкой (Optimizer failed)

Согласно best practices, оптимальные optimal_force обычно меньше 10 Н·м, чтобы актуаторы не доминировали над мышцами и не искажали биомеханику [41]. Также важно проверить, что актуаторы не работают на максимальных значениях во всех DOF (degree of reedom – степени свободы) (иначе это означает, что мышцы слишком слабые)

Для подбора optimal_force мы запустили серию оптимизаций с разными значениями. При больших значениях (20, 10, 5) ошибок не было, значение constraint violation было параметра 1e-10. При значениях ниже (1.0, 2.0, 2.1) наблюдались ошибки solver'a. Значение 2.5, 3.0, 3.5, 4.0, 4.5 работали корректно лишь на некоторых из сегментированных записей. Значение 5.0 выбрали как минимальное стабильное, причем оно удовлетворяет best practices из документации.

При добавлении актуаторов и передачи в функцию StaticOptimization соответствующих файлов метод создает выходные файлы движения, по которым мы можем оценить степень участия актуаторов в формировании движения (рисунок 6.8). Полный программный код для реализации формирования актуаторов и запуска StaticOptimization можно посмотреть в приложении (Приложение Е).

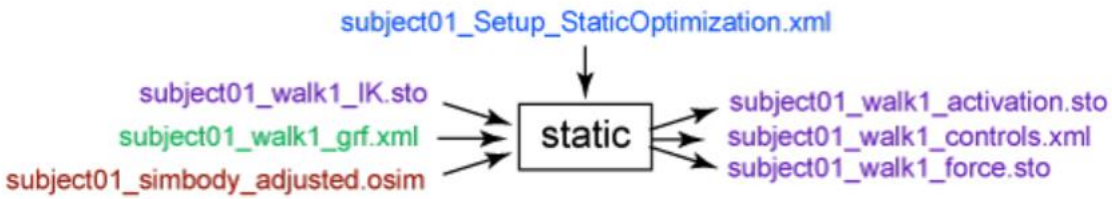


Рисунок 6.8 – Inputs and Outputs of the Static Optimization Tool [42]

Сформированный файл StaticOptimization_activation.sto содержит в себе активации мышц и control-сигналы резервных актуаторов. Файл StaticOptimization_force.sto соответственно усилия мышц.

Корректировка углов суставов большого пальца. Для анализа активации актуаторов при движении были выбраны 5 из них с наибольшими максимальными абсолютными значениями и отображены на графике (рисунок 6.9).

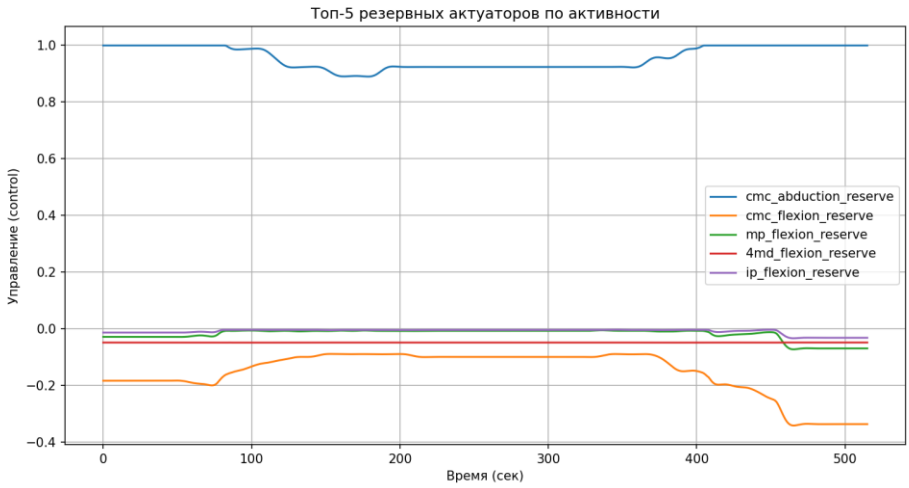


Рисунок 6.9 – значения активации актуаторов по субъекту 1 движению 1 (повторение 2) из группы 1 (сгибание указательного пальца)

Можем наблюдать наибольшие отклонения в значениях cmc_abduction_reserve и mp_flexion_reserve. Это актуаторы для суставов большого пальца. При этом в движение, соответствующее этому графику – сгибание указательного пальца. И большой в нем практически остается в покое.

Соответственно была выдвинута гипотеза, что углы приведения/отведения и сгибания большого пальца некорректны. Для

проверки этой гипотезы также были сформированы аналогичные файлы для других повторений этого же движения (рисунок 6.10).

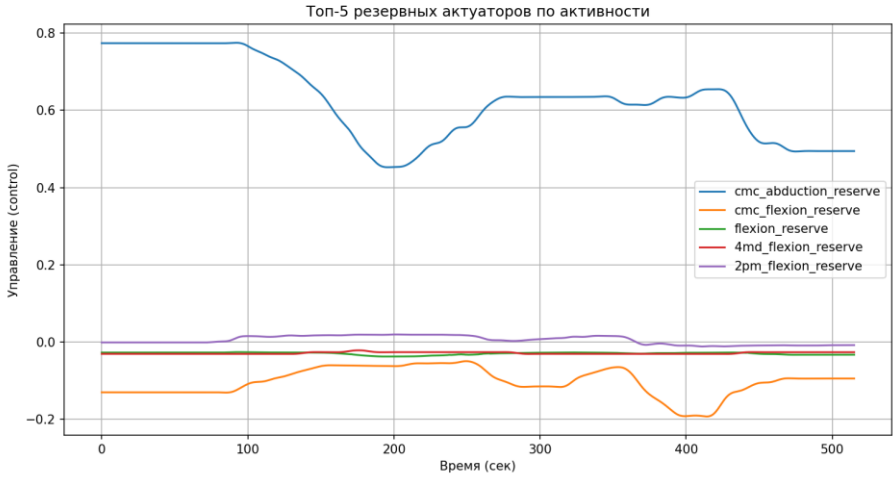


Рисунок 6.10 – значения активации актуаторов по субъекту 1 движению 5 (повторение 1) из группы 1 (сгибание указательного пальца)

В других движениях и повторениях данная ошибка также повторялась. После чего были пересмотрены параметры калибровки углов для преобразования из данных CyberGlove II в .mot файл и найдены ошибки в значении cmc_abduction и cmc_flexion (эти значения наиболее трудно калибруются среди всех углов пальцев). Были подобраны значения для каллибровки этих углов, которые делают движение более физичным. После чего заново пройдены этапы решения с помощью OpenSim и построения значений активации актуаторов, теперь они стали более приемлемые (рисунок 6.11).

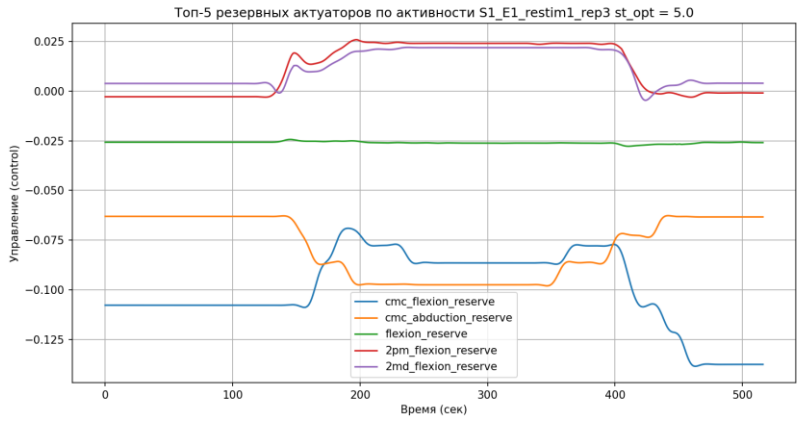


Рисунок 6.11 – значения активации актуаторов субъекта 1 жеста «сгибание указательного пальца» после пересмотра значений калибровки

Аналогичные графики были построены для других движений других групп. Например, для «жест указания», когда сгибаются все пальцы, кроме указательного, а он – выпрямляется (рисунок 6.12). Наибольшие активации все еще наблюдаются для сустава приведения/отведения большого пальца. Такие значения активации актуаторов являются допустимыми для дальнейшего проведения исследования.

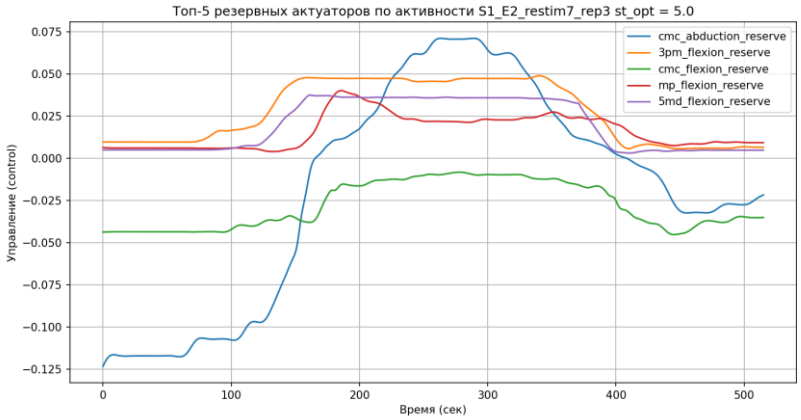


Рисунок 6.12 - значения активации актуаторов субъекта 1 ("жест указания") после пересмотра значений калибровки

Корректная кинематика важна для биомеханической правдоподобности модели. Калибровка углов большого пальца все еще требует пересмотра для лучшего результата. Однако уже можно оценить сходство сгенерированных Static Optimization EMG сигналов с исходными значениями.

7 РЕЗУЛЬТАТЫ

В рамках работы было проведено сопоставление рассчитанных методом Static Optimization значений активации мышц с экспериментальными данными из базы NinaPrro1. Целью сравнения являлась оценка достоверности симуляций мышечной активности в модели кисти, основанной на кинематических данных с перчатки CyberGloveII.

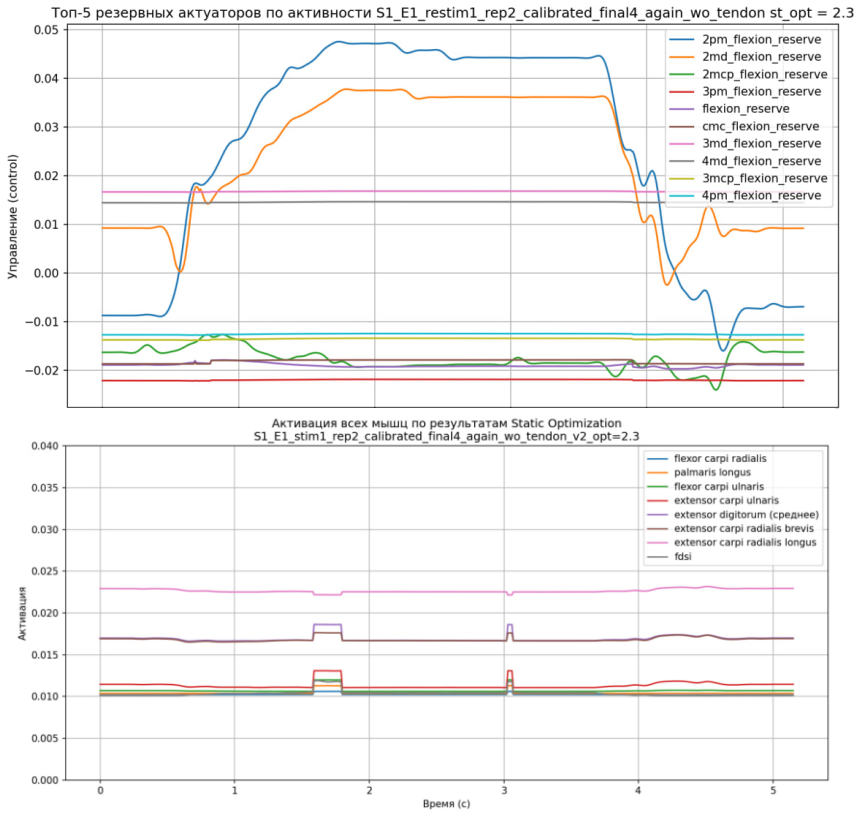


Рисунок 7.1 – значения активаций актуаторов и активации мышц при выполнении жеста «сгибание указательного пальца»

Изначально результаты моделирования показали, что основную нагрузку при выполнении движений брали на себя актуаторы, встроенные в модель для балансировки сил и возможности проведения расчетов методом Static Optimization (рисунок 7.1). Это привело к тому, что для большинства мышц значения симулироемой активации оставались близкими к нулю, несмотря на наличие экспериментально зарегистрированной активности.

Однако после дополнительного изучения документации OpenSim

модель была скорректирована в соответствии раздела 1.6.3 настоящего диплома и значения активации мышц уже также показали значения отличные от спокойных (рисунок 7.2). Однако вклад актуаторов суставов все еще остается больше.

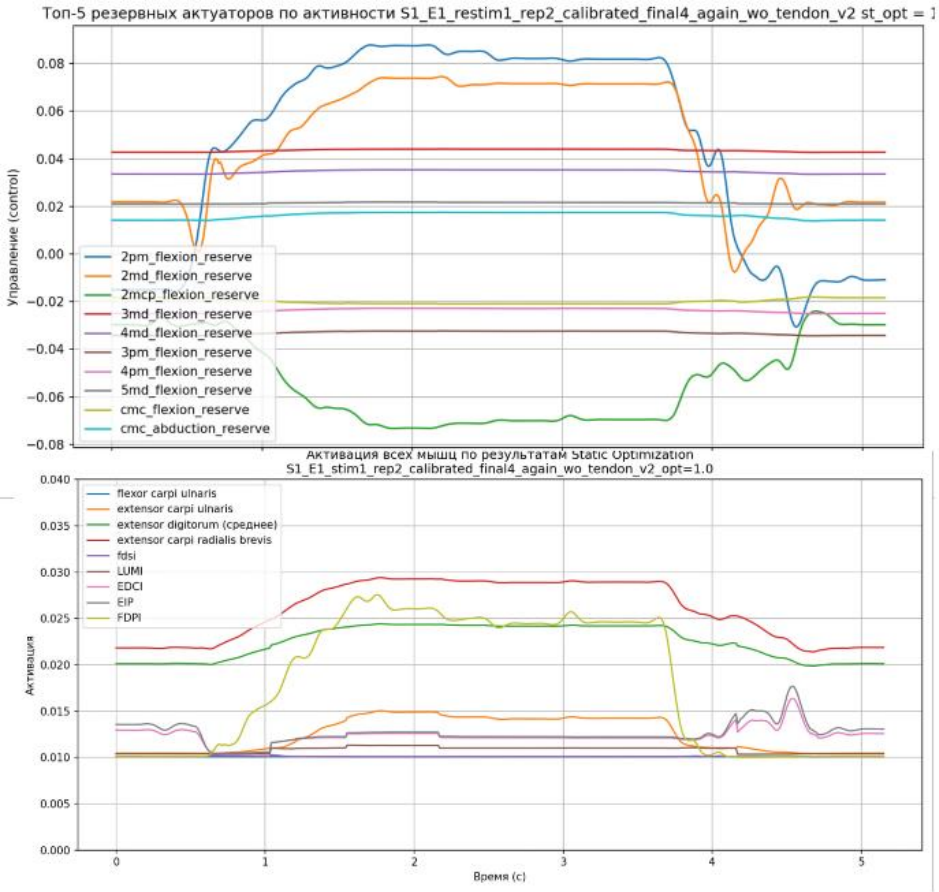


Рисунок 7.2 – значения активаций актуаторов и активации мышц при выполнении жеста «сгибание указательного пальца» после корректировки модели

Правдоподобные результаты получились также для мышц-антагонистов, которые подвергались пассивному растяжению. В этих случаях модель корректно отражала рост пассивных сил и регистрировала соответствующую активность (рисунок 7.2). Это хорошо согласуется с физиологическими механизмами пассивной коактации мышц.

Частично корректные результаты обусловлены недостаточной корректировкой модели, приводящей к искажению мышечной кинематики.

Таким образом текущее состояние не позволяет в полной мере воспроизводить паттерны мышечной активности, наблюдаемые в

экспериментальных данных ЭМГ. Однако совпадение формы акивации у некоторых мышц в условиях пассивного растяжения и наличие активных сил указывает на потенциал использования данной методики при более глубокой настройке модели и ограничении вмешательства вспомогательных элементов.

Среди предполагаемых причин при анализе модели и документации также выделили:

- приведение параметра fiber_damping к малому значении. fiber_dumbling отвечает за вязкое сопротивление мышечного волокна при изменении длины. Необходимо проверить что это сопротивление не имеет сверхвысоких значений;
- проверка Constraints параметров. Они отвечают за взаимосвязь значений суставов друг с другом;
- оценить вклад Other_Forces (прочих сил), исключить те, которые не являются мышечными и могут оказывать влияние на динамику модели.

Заключение

В ходе работы был разработан подход для обратной симуляции мышечной активности на основе экспериментальных данных, включающих измерения (с перчатки CyberGlove II) и сигналы поверхностной электромиографии (с 10 каналов на поверхностных мышцах предплечья), взятые из базы данных NinaPro DB1.

Для оценки возможностей воспроизведения мышечной активности в биомеханической модели была проведена настройка существующей модели кисти в OpenSim, включающая:

- предварительная обработка и сегментация экспериментальных данных на отдельные файлы по каждому субъекту, каждому движению и каждому повторению;
- импорт и преобразование угловых данных в формат движения .mot;
- корректировка начальных положений суставов модели для физиологичного положения «покоя»;
- выполнение статической оптимизации на основе построенного движения;
- извлечение и анализ рассчитанных значений активации и удлинения мышц;

Изначальные результаты показали, что активность мышц-агонистов, выполняющих движение отсутствовала, а основное движение выполнялось за счет вспомогательных актуаторов. После чего была проведена дополнительная настройки модели: устранения несоответствий в исходном положении суставов и параметрах мышц, благодаря чему удалось получить активации, отличающиеся от нулевых значений, а также реалистичное поведение мышц-антагонистов, подвергающихся пассивному растяжению. Это подтверждает корректную работу механизма пассивной коактивации в модели.

Тем не менее, симулируемая активность все еще не в полной мере воспроизводит характер ЭМГ-сигнала, что указывает на необходимость дальнейшего усовершенствования модели. Для доработки модели, а также для усовершенствования разработанного подхода:

1. Проведение повторную настройку модели с пересмотром параметров мышц (максимальная сила, оптимальная длина, сухожильные характеристики и др.), чтобы обеспечить появление реалистичных активаций у мышц-агонистов, без гиперкомпенсации актуаторами при расчетах методом Static Optimization.

2. Анализ и контроль влияния таких параметров как дополнительные силы (Other Forces), механические зависимости (Constraints) на производимые расчеты.

3. Внедрение декомпозиции ЭМГ-сигнала для более точной интерпретации экспериментально полученных значений поверхностной электромиографии.

4. Использование альтернативного метода расчета активаций – Computed Muscle Control (CMC), для динамической оценки управления мышцами.

Разработанный подход с текущими результатами уже демонстрирует потенциал использования биомеханического моделирования в качестве инструмента для анализа и верификации мышечной активности.

После отладки предварительных настроек модели для перед оптимизацией подход сможет представить собой альтернативу существующим методам моделирования движений кисти на основе поверхностной ЭМГ. На фоне традиционных решений, сосредоточенных на классификации жестов, используя ЭМГ, как абстрактный сигнал, не связанный с архитектурой мышц, предложенный метод использует биомеханическое моделирование с учетом анатомической структуры.

Таким образом разработанный подход может открыть путь к более интерпретируемым и физиологически обоснованным методам восстановления

движений и нейромышечной диагностики при решении как научных, так и клинических задач.

Список использованных источников

1. Robertson D.G.E., Caldwell G.E., Hamill J., Kamen G., Whittlesey S.N. Research Methods in Biomechanics. 2-е изд. Human Kinetics, 2013. 440 с. URL: https://www.humankinetics.com/products/research-methods-in-biomechanics-2nd-edition (дата обращения: 16.06.2025)

2. Уткин В.М. – Биомеханика физических упражнений. Москва, 1989. 272 с. URL: http://vlu-st.ru/BFU.pdf (дата обращения: 16.06.2025)

3. А.С. Бескровный, Л.В. Бессонов, А.А. Голядкина и др. Разработка системы поддержки принятия врачебных решений в травматологии и ортопедии. Биомеханика как инструмент предоперационного планирования // Российский журнал биомеханики. 2021. Т. 25, № 2. URL: https://cyberleninka.ru/article/n/razrabotka-sistemy-podderzhki-prinyatiya-vrachebnyh-resheniy-v-travmatologii-i-ortopedii-biomehanika-kak-instrument (дата обращения: 16.06.2025)

4. Ju F., Wang Y., Xie B., Mi Y., Zhao M., Cao J. The Use of Sports Rehabilitation Robotics to Assist in the Recovery of Physical Abilities in Elderly Patients with Degenerative Diseases: A Literature Review // Healthcare (Basel). 2023. Vol. 11, № 3. Article 326. DOI: 10.3390/healthcare11030326. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC9914201/ (дата обращения: 16.06.2025).

5. Koike Y., Kawato M. Estimation of dynamic joint torques and trajectory formation from surface EMG signals using a neural network model // Biological Cybernetics. 1995. Vol. 73. DOI: 10.1007/BF00201445 URL: https://link.springer.com/article/10.1007/BF00201445 (дата обращения: 16.06.2025)

6. Resnik L.J., Klinger S.L., Etter K. The DEKA Arm: Its features, functionality, and evolution during the Veterans Affairs Study // Prosthetics and Orthotics International. 2014. Vol. 38, № 6. DOI: 10.1177/0309364613506913 URL: https://journals.sagepub.com/doi/10.1177/0309364613506913 (дата

обращения: 16.06.2025)

7. Sabuba M., Ab-Alwafa R., Mhseen D., Zidan E., Zorba F., Qaddumi J. – Prevalence of Absence of the Palmaris Longus Muscle among Medical Students of An-Najah National University: a Cross-Sectional Study from Palestine // Palestinian Medical and Pharmaceutical Journal. 2021. Vol. 7, № 1. Article 9. DOI: 10.59049/2790-0231.1063. URL: https://pmpj.najah.edu/journal/vol7/iss1/9/ (дата обращения: 16.06.2025).

8. Иванов Д.В., Доль А.В. Биомеханическое моделирование: монография. Саратов: Изд-во СГУ, 2021. URL: https://old.sgu.ru/sites/default/files/education/method/2021/biomehanicheskoe_modelirovanie.pdf (дата обращения: 17.02.2025)

9. Schepers H.M., van Asseldonk E.H.F., Buurke J.H., Veltink P.H. Measurement of Ground Reaction Forces and Moments during Overground Walking Using Inertial Sensing // Sensors. 2017. Vol. 17, No. 9. DOI: 10.3390/s17092085. URL: https://www.mdpi.com/1424-8220/17/9/2085 (дата обращения: 15.01.2025)

10. Sy L.W.F., Viswanathan S., Linninger A.A. Estimating Lower Body Kinematics using a Lie Group Constrained Extended Kalman Filter and Reduced IMU Count // Sensors. 2020. DOI: 10.3390/s20236829. URL: https://www.mdpi.com/1424-8220/20/23/6829 (дата обращения: 15.01.2025)

11. Чадова М., Галло Л.М. Подходит ли OpenSim для анализа жевательной системы? // Российский журнал биомеханики. – 2013. – № 3. URL: https://cyberleninka.ru/article/n/podhodit-li-opensim-dlya-analiza-zhevatelnoy-sistemy (дата обращения: 15.01.2025)

12. Zeng X.S., Dwarakanath S., Lu W., Nakada M., Terzopoulos D. Neuromuscular – Control of the Face–Head–Neck Biomechanical Complex With Learning-Based Expression Transfer From Images and Videos // Computers & Graphics. 2021. DOI: 10.1016/j.cag.2021.01.004. URL:https://www.researchgate.net/publication/356198862_Neuromuscular_Control_of_the_Face-Head-Neck_Biomechanical_Complex_With_Learning-Based_Expression_Transfer_From_Images_and_Videos (дата обращения:

15.01.2025).

13. Long Y., Geng Y., Dai C., Li P. A Transfer Learning Based Cross-Subject Generic Model for Continuous Estimation of Finger Joint Angles from a New User // Sensors. 2023. Vol. 23, No. 13. Article 5952. DOI: 10.3390/s23135952. URL: https://www.researchgate.net/publication/366922788_A_Transfer_Learning_based_Cross-subject_Generic_Model_for_Continuous_Estimation_of_Finger_Joint_Angles_from_a_New_User (дата обращения: 15.01.2025).

14. Wei J., Meng Q., Badii A. Classification of human hand movements using surface EMG for myoelectric control // Advances in Intelligent Systems and Computing. – 2017. – Vol. 513. – DOI: 10.1007/978-3-319-46562-3_22. – URL: https://repository.lboro.ac.uk/articles/conference_contribution/Classification_of_human_hand_movements_using_surface_EMG_for_myoelectric_control/9403136 (дата обращения: 15.01.2025).

15. Rahimi F., Masson W., Leone M., Atzori M. Simultaneous Control of Human Hand Joint Positions and Grip Force via HD-EMG and Deep Learning // IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2024. Vol. 32. DOI: 10.1109/TNSRE.2024.3389987. https://arxiv.org/abs/2410.23986 (дата обращения 01.03.2025)

16. Wei W., Dai Q., Wong Y., Hu Y., Kankanhalli M., Geng W. Surface Electromyography-based Gesture Recognition by Multi-view Deep Learning // IEEE Transactions on Biomedical Engineering. – 2019. – DOI: 10.1109/TBME.2019.2899222. – URL: https://doi.org/10.1109/TBME.2019.2899222 (дата обращения: 15.06.2025).

17. Casu A., Rahimi F., Masson W., Atzori M. sEMG-Based Continuous Estimation of Finger Kinematics via Large-Scale Temporal Convolutional Network // Applied Sciences. 2021. Vol. 11, No. 10. DOI: 10.3390/app11104678. URL: https://www.mdpi.com/2076-3417/11/10/4678 (дата обращения: 19.01.2025)

18. Яковлев В.Н. Нормальная физиология: – Учебные модули для самостоятельной работы студентов / под ред. В.Н. Яковлева. – Воронеж: ИПФ

«XXI век», 2010. URL: https://academia-moscow.ru/ftp_share/_books/fragments/fragment_20294.pdf (дата обращения: 12.01.2025).

19. Попова Н.Н., Артемьева С.С. Физиология мышц / Министерство спорта Российской Федерации. – Москва: ВГИФК, 2022. URL: https://www.vgifk.ru/sites/default/files/docs/popova_n.n._artemeva_s.s._fiziologiya_myshc_2022.pdf (дата обращения: 12.01.2025).

20. Chen C., Li D., Xia M. – A motor unit action potential–based method for surface electromyography decomposition // J NeuroEng Rehabil. – 2025. – Т. 22, № 60. – DOI: 10.1186/s12984-025-01595-y URL: https://jneuroengrehab.biomedcentral.com/articles/10.1186/s12984-025-01595-y (дата обращения: 11.02.2025)

21. И.Н. Новикова и др. – Анатомия скелетных мышц: учебное пособие; Белорус. гос. мед. ун-т. – Минск: БГМУ, 2019. URL: https://rep.bsmu.by/bitstream/handle/BSMU/36858/978-985-21-1162-1.pdf (дата обращения: 11.02.2025).

22. Семенова Л.М., Куприянов С.В., Бочкарев С.В., Романова Л.П., Перина С.С. – Физиология возбудимых тканей: учебное пособие. – Чебоксары: Изд-во Чуваш. ун-та, 2014. URL: https://studfile.net/preview/21488527/page:6/ (Дата обращения 11.03.2025)

23. Lv Y., Zheng Q., Chen X., Hou C., An M. – Analysis on synergistic cocontraction of extrinsic finger flexors and extensors during flexion movements: A finite element digital human hand model // PLoS ONE. 2022. Vol. 17, № 5. Article № e0268137. DOI: 10.1371/journal.pone.0268137. URL:https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0268137 (Дата обращения: 12.04.2025)

24. Привес М.Г., Лысенков Н.К., Бушкович В.И. – «Анатомия человека». – Москва «Медицина» - 1985 URL: https://jasulib.org.kg/wp-content/uploads/2023/02/12.-%D0%9F%D1%80%D0%B8%D0%B2%D0%B5%D1%81-%D0%9C.%D0%93.-

%D0%90%D0%BD%D0%B0%D1%82%D0%BE%D0%BC%D0%B8%D1%8F-%D1%87%D0%B5%D0%BB%D0%BE%D0%B2%D0%B5%D0%BA%D0%B0.pdf (Дата обращения 30.09.2024)

25. Rukina N.N., Kuznetsov A.N., Borzikov V.V., Komkova O.V., Belova A.N. – Surface electromyography: its role and potential in the development of exoskeleton (review). Sovremennye tehnologii v medicine 2016; 8(2): 109–118, http://dx.doi.org/10.17691/ stm2016.8.2.15. URL: https://www.stm-journal.ru/en/numbers/2016/2/1253/pdf (Дата обращения: 12.04.2025)

26. Delp S. L. et al. OpenSim: open-source software to create and analyze dynamic simulations of movement. IEEE Transactions on Biomedical Engineering, 2007. URL: https://pubmed.ncbi.nlm.nih.gov/18018689/ (Дата обращения: 12.04.2025)

27. Seth A. et al. OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. PLOS Comput Biol, 2018. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006223 (Дата обращения: 12.04.2025)

28. Hamner S.R. et al. Muscle contributions to propulsion and support during running // Journal of Biomechanics, 2010. URL: https://pubmed.ncbi.nlm.nih.gov/20691972/ (Дата обращения: 12.04.2025)

29. OpenSim Python API documentation: https://opensimconfluence.atlassian.net/wiki/spaces/OpenSim/pages/53085346/Scripting+in+Python (дата обращения: 20.09.2024).

30. Millard, M., Uchida, T., Seth, A., & Delp, S.L. (2013) – Flexing computational muscle: modeling and simulation of musculotendon dynamics. Journal of Biomechanical Engineering, 135(2). URL: https://pubmed.ncbi.nlm.nih.gov/23445050/ (Дата обращения: 12.04.2025)

31. OpenSim Developer Documentation https://simtk.org/api_docs/opensim/api_docs31/classOpenSim_1_1ActiveForceLengthCurve.html (дата обращения: 19.03.2025).

32. Characteristic Musculotendon Curves. OpenSim Documentation. https://opensimconfluence.atlassian.net/wiki/spaces/OpenSim/pages/53089801/Characteristic+Musculotendon+Curves (дата обращения: 19.03.2025).

33. Atzori M., Gijsberts A., Castellini C., et al. Electromyography data for non-invasive naturally-controlled robotic hand prostheses. Scientific Data 1, 140053 (2014). DOI: 10.1038/sdata.2014.53 URL: https://www.nature.com/articles/sdata201453 (Дата обращения: 12.04.2025)

34. NinaPro Database DB1. Сайт проекта: https://ninapro.hevs.ch/instructions/DB1.html (Дата обращения: 21.02.2024)

35. Mousavi Hondori H., Ebrahimi S., Rostami A. et al. A Review of EMG Onset Detection Methods for Real-Time Applications. Bioengineering. 2023;10(10):1300. doi:10.3390/bioengineering10101300. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10594734/ (дата обращения: 10.01.2025).

36. Uhlrich SD, Klotz T, Negrello E, et al. OpenSim Moco: Musculoskeletal Optimal Control. PLoS Comput Biol. 2022;18(2):e1009732. doi:10.1371/journal.pcbi.1009732. PMID: 35500997 URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008493 (Дата обращения: 17.02.2025)

37. Thelen DG, Anderson FC, Delp SL. Generating dynamic simulations of movement using computed muscle control. Journal of Biomechanics. 2003;36(3):321-328. Доступно: https://simtk.org/projects/cmc/ (Дата обращения: 18.02.2025)

38. OpenSim Documentation. How Static Optimization Works [Электронный ресурс] // OpenSim Documentation – 2024. – URL: https://opensimconfluence.atlassian.net/wiki/spaces/OpenSim/pages/53089619/How+Static+Optimization+Works (дата обращения: 16.03.2025).

39. OpenSim Documentation. How CMC Works [Электронный ресурс] // OpenSim Documentation – 2024. – URL: https://opensimconfluence.atlassian.net/wiki/spaces/OpenSim/pages/53089706/Ho

w+CMC+Works (дата обращения: 01.05.2025).

40. CoordinateActuator – Class Reference // OpenSim Moco 1.1.0 API Reference. URL: https://opensim-org.github.io/opensim-moco-site/docs/1.1.0/html_user/classOpenSim_1_1CoordinateActuator.html (дата обращения: 13.06.2025).

41. Working with Static Optimization // OpenSim Documentation. URL: https://opensimconfluence.atlassian.net/wiki/spaces/OpenSim/pages/53085189/Working%2Bwith%2BStatic%2BOptimization (дата обращения: 13.06.2025).

42. Getting Started with Static Optimization // OpenSim Documentation. URL: https://opensimconfluence.atlassian.net/wiki/spaces/OpenSim/pages/53089624/Getting%2BStarted%2Bwith%2BStatic%2BOptimization (дата обращения: 13.06.2025).

43. Lieber RL, Fridén J, Botte MJ. Architecture of selected muscles of the arm and forearm. *J Neurophysiol*. 1990.

Приложение А

(справочное)

Программный код для визуализации полных исходных записей

```
# класс для загрузки параметров эксперимента

class Loader:
    def __init__(self):
        script_dir = os.path.dirname(os.path.abspath(__file__))
        self.parent_dir = os.path.abspath(os.path.join(script_dir, '..'))
        config_path = os.path.join(self.parent_dir, 'config.json')

        with open(config_path, 'r') as f:
            config = json.load(f)

        self.db = config.get("db")
        self.subject = config.get("subject")
        self.exercise = config.get("exercise")
        self.stimulus = config.get("stimulus")
        self.repetition = config.get("repetition")
        self.opt_force = config.get("opt_force")
        self.postfix = config.get("postfix")

    def __repr__(self):
        return (f"Config(db={self.db}, subject={self.subject}, exer-
cise={self.exercise})")

    def save_glove_dataframe(self, df: pd.DataFrame, restimulus_number: int,
rep_number: int):
        save_dir = os.path.join(self.parent_dir, 'data', 'glove_segments',
f'DB{self.db}')
        os.makedirs(save_dir, exist_ok=True)
        filename = f"glove_S{self.subject}_E{self.exercise}_restim{restimu-
lus_number}_rep{rep_number}.csv"
        file_path = os.path.join(save_dir, filename)
        df.to_csv(file_path, index=False, header=False)
        print(f"[INFO] Glove data saved to: {file_path}")

    def save_emg_dataframe(self, df: pd.DataFrame, restimulus_number: int,
rep_number: int):
        save_dir = os.path.join(self.parent_dir, 'data', 'emg_segments',
f'DB{self.db}')
        os.makedirs(save_dir, exist_ok=True)
        filename = f"emg_S{self.subject}_E{self.exercise}_restim{restimu-
lus_number}_rep{rep_number}.csv"
        file_path = os.path.join(save_dir, filename)
        df.to_csv(file_path, index=False, header=False)
        print(f"[INFO] EMG data saved to: {file_path}")


# класс для загрузки исходных данных из NinaPro
class NinaProData:
    def __init__(self):
        self.loader = Loader()
        self._load_data()
        self._compute_global_ranges()

    def _load_data(self):
        script_dir = os.path.dirname(os.path.abspath(__file__))
        grandparent_dir = os.path.abspath(os.path.join(script_dir, '..'))
```

```
        file_path = os.path.join(grandparent_dir, 'data', 'raw_data',
f'DB{self.loader.db}',
                                 f'S{self.loader.subject}_E{self.loader.exer-
cise}_A1.mat')

        data = scipy.io.loadmat(file_path)

        self.emg = data['emg'][:, :8]
        self.glove = data['glove']
        self.restimulus = data['restimulus'].flatten()
        self.repetition = data['repetition'].flatten()
        self.rerepetition = data['rerepetition'].flatten()

        if self.loader.db == 1:
            self.fs_emg = 100
        elif self.loader.db == 2:
            self.fs_emg = 2000

    def _compute_global_ranges(self):
        self.emg_min = np.min(self.emg)
        self.emg_max = np.max(self.emg)
        emg_range = self.emg_max - self.emg_min
        self.emg_min -= 0.05 * emg_range
        self.emg_max += 0.05 * emg_range

        self.glove_min = np.min(self.glove)
        self.glove_max = np.max(self.glove)
        glove_range = self.glove_max - self.glove_min
        self.glove_min -= 0.05 * glove_range
        self.glove_max += 0.05 * glove_range

        label_min = min(np.min(self.restimulus), np.min(self.repetition),
np.min(self.rerepetition))
        label_max = max(np.max(self.restimulus), np.max(self.repetition),
np.max(self.rerepetition))
        label_range = label_max - label_min if label_max != label_min else 1
        self.label_min = label_min - 0.05 * label_range
        self.label_max = label_max + 0.05 * label_range

# Функция для отображения данных
def plot_emg_and_glove(emg, glove, restimulus, repetition, rere, subject, ex-
ercise, time_emg, db,
                       emg_min, emg_max, glove_min, glove_max, label_min, la-
bel_max):
    # ЭМГ
    fig1, (ax1, ax2) = plt.subplots(2, 1, figsize=(15, 8), sharex=False)
    for i in range(emg.shape[1]):
        ax1.plot(time_emg, emg[:, i], label=f'EMG {i + 1}')
    ax1.set_ylabel('Амплитуда ЭМГ')
    ax1.set_ylim(emg_min, emg_max)
    ax1.set_title(f'Субъект {subject}, Упражнение {exercise}')
    ax1.legend(loc='upper right', ncol=4)

    ax2.plot(time_emg, restimulus, label='restimulus', linewidth=1)
    ax2.plot(time_emg, repetition, label='repetition', linewidth=1)
    ax2.plot(time_emg, rere, label='rerepetition', linewidth=1)
    ax2.set_ylabel('Метки')
    ax2.set_xlabel('Время (с)')
    ax2.set_ylim(label_min, label_max)
    ax2.legend(loc='upper right')

    plt.tight_layout()
    plt.show()
```

```
    # Перчатка
    fig2, (ax3, ax4) = plt.subplots(2, 1, figsize=(15, 8), sharex=False)
    for i in range(glove.shape[1]):
        idx = i + 1
        if True or idx != 11 or db == 1:
            style = sensor_plot_styles.get(idx, {"color": "gray", "lin-
estyle": '-'})
            ax3.plot(time_emg, glove[:, i], label=f'Glove {idx}',
                     color=style["color"], linestyle=style["linestyle"])
    ax3.set_ylabel('Сигнал перчатки')
    ax3.set_ylim(glove_min, glove_max)
    ax3.set_title(f'Субъект {subject}, Упражнение {exercise}')
    ax3.legend(loc='upper right', ncol=4)

    ax4.plot(time_emg, restimulus, label='restimulus', linewidth=1)
    ax4.plot(time_emg, repetition, label='repetition', linewidth=1)
    ax4.plot(time_emg, rere, label='rerepetition', linewidth=1)
    ax4.set_ylabel('Метки')
    ax4.set_xlabel('Время (с)')
    ax4.set_ylim(label_min, label_max)
    ax4.legend(loc='upper right')

    plt.tight_layout()
    plt.show()

def plot_from_to(t_start, t_end, data):

    emg_start = int(t_start * data.fs_emg)
    emg_end = int(t_end * data.fs_emg)

    time_emg = np.arange(emg_start, emg_end) / data.fs_emg

    plot_emg_and_glove(
        data.emg[emg_start:emg_end],
        data.glove[emg_start:emg_end],
        data.restimulus[emg_start:emg_end],
        data.repetition[emg_start:emg_end],
        data.rerepetition[emg_start:emg_end],
        data.loader.subject,
        data.loader.exercise,
        time_emg,
        data.loader.db,
        data.emg_min, data.emg_max,
        data.glove_min, data.glove_max,
        data.label_min, data.label_max
    )

# пример вызова для отображения данных
if __name__ == "__main__":
    data = NinaProData()

    for t_start in range(0, 3000, 30):
        t_end = min(t_start + 30, 2014) # 2014 это для конкретного примера
        plot_from_to(t_start, t_end, data)
```

Приложение Б

(справочное)

Программный код для сегментации записей и сохранение сегментов для дальнейшей интеграции в модель

```
def process_and_save_segments():
    data = NinaProData()

    # Маска активных точек: где хотя бы одно значение не ноль
    active_mask = (data.restimulus != 0) | (data.repetition != 0) |
(data.rerepetition != 0)

    segments = {}
    current_indices = []

    for i in range(len(active_mask)):
        if active_mask[i]:
            current_indices.append(i)
        else:
            if current_indices:
                # Первый индекс (где restimulus и repetition > 0)
                valid_start_index = next(
                    (idx for idx in current_indices
                     if data.restimulus[idx] > 0 and data.repetition[idx] >
0),
                    None
                )
                if valid_start_index is not None:
                    key = (data.restimulus[valid_start_index], data.repeti-
tion[valid_start_index])
                    segments[key] = current_indices.copy()
                current_indices.clear()

    # если файл не закончился нулями
    if current_indices:
        valid_start_index = next(
            (idx for idx in current_indices
             if data.restimulus[idx] > 0 and data.repetition[idx] > 0),
            None
        )
        if valid_start_index is not None:
            key = (data.restimulus[valid_start_index], data.repeti-
tion[valid_start_index])
            segments[key] = current_indices.copy()

    # Сохраняем все сегменты
    for (stim, rep), indices in segments.items():
        segment_emg = pd.DataFrame(data.emg[indices, :])
        segment_glove = pd.DataFrame(data.glove[indices, :])

        data.loader.save_emg_dataframe(segment_emg, restimulus_num-
ber=int(stim), rep_number=int(rep))
        data.loader.save_glove_dataframe(segment_glove, restimulus_num-
ber=int(stim), rep_number=int(rep))

    print(f"[INFO] Обработано {len(segments)} сегментов субъекта
{data.loader.subject} exercise {data.loader.exercise}")
```

Приложение В

(справочное)

Программный код для аппроксимации ЭМГ-сигнала для дальнейшего сравнения и численной оценки результатов

```
def plot_emg_segment(emg_csv_path):
    # скользящее среднее (moving average)
    def smooth_emg(signal, window_size=50):
        window = np.ones(window_size) / window_size
        return np.apply_along_axis(lambda x: np.convolve(x, window,
mode='same'), axis=0, arr=signal)


    emg_df = pd.read_csv(emg_csv_path, header=None)
    emg_df_smoothed = smooth_emg(emg_df)

    plt.figure(figsize=(15, 6))
    for i in range(emg_df.shape[1]):
        plt.plot(emg_df.iloc[:, i], label=f'EMG {i + 1}', color=emg_sen-
sor_colors[i+1], linestyle='-', alpha=0.5)
        plt.plot(emg_df_smoothed[:, i], label=f'smoothed EMG {i + 1}',
color=emg_sensor_colors[i+1], linestyle='--')
    plt.title("EMG Segment")
    plt.xlabel("Время (мсек)")
    plt.ylabel("mV")
    plt.legend(ncol=4, loc='upper right')
    plt.tight_layout()
    plt.show()
```

Приложение Г

(справочное)

Программный код для выставления углов и визуализации движений

```
muscles_to_keep = {
    'ECRL',  # extensor carpi radialis longus
    'ECRB',  # extensor carpi radialis brevis
    'ECU',   # extensor carpi ulnaris
    'FCR',   # flexor carpi radialis
    'FCU',   # flexor carpi ulnaris
    'PL',    # palmaris longus
    'EDCL', 'EDCR', 'EDCM', 'EDCI'  # extensor digitorum components
    # 'brachioradialis' - если есть в модели
}

length_data = {name: [] for name in muscles_to_keep}

#Углы перчатки:
glove_angles = {
    # большой палец
    1: 'cmc_flexion',
    2: 'mp_flexion',
    3: 'ip_flexion',
    4: 'cmc_abduction',
    # указательный
    5: '2mcp_flexion',
    6: '2pm_flexion',
    7: '2md_flexion',
    # средний
    8: '3mcp_flexion',
    9: '3pm_flexion',
    10: '3md_flexion',
    11: '', # разница между '2mcp_abduction' и '3mcp_abduction'
    # безымянный
    # '4mcp_abduction'
    12: '4mcp_flexion',
    13: '4pm_flexion',
    14: '4md_flexion',
    15: '', # разница между '3mcp_abduction' и '4mcp_abduction'
    # мизинец
    16: '5mcp_flexion',
    17: '5pm_flexion',
    18: '5md_flexion',
    19: '', # разница между '4mcp_abduction' и '5mcp_abduction'
    20: '', # '4cmc_flexion'
    # запястье
    21: 'flexion', # radians
    22: 'deviation' # radians
}

#########################################################################
# импорт модели и библиотек еще
#########################################################################


model = osim.Model(model_path)

# Включаем визуализатор до инициализации
model.setUseVisualizer(True)
```

```
# Только после этого инициализируем систему
state = model.initSystem()

# Получение всех координат
coords = model.getCoordinateSet()

flag = False
for glove_test_angles_line in glove_test_angles[:1000:5]:
    for key, val in glove_angles.items():
        if val !=  '':
            # Установка нового значения для одного из суставов
            coord_name_to_modify = val
            new_angle_deg = glove_test_angles_line[key]
            if val in ['deviation']:
                new_angle_deg = 120 - new_angle_deg
            elif val in ['flexion']:
                new_angle_deg = new_angle_deg - 180
            elif val in ['cmc_flexion']:
                new_angle_deg = new_angle_deg - 180
            elif val in ['cmc_abduction']:
                new_angle_deg = 180 - new_angle_deg
            else:
                new_angle_deg = new_angle_deg - 80
            new_angle_rad = math.radians(new_angle_deg)

            if coords.contains(coord_name_to_modify):
                coord = coords.get(coord_name_to_modify)
                coord.setLocked(state, False)  # если вдруг заблокировано
                name = coord.getName()
                min_angle = coord.getRangeMin()
                max_angle = coord.getRangeMax()
                # print(f"{idx}: {name} - диапазон: от {math.degrees(min_an-
gle):.1f}° до {math.degrees(max_angle):.1f}°")
                # coord.setRangeMin(min(new_angle_rad - 1.0, math.radi-
ans(min_angle)))  # можно уточнить допустимый диапазон
                coord.setRangeMin(min(0.0, math.radians(min_angle)))  # можно
уточнить допустимый диапазон
                # coord.setRangeMax(max(new_angle_rad + 1.0, math.radi-
ans(max_angle)))
                coord.setRangeMax(max(2.0, math.radians(max_angle)))
                # print(f"\nУгол '{coord_name_to_modify}' установлен в
{new_angle_deg}°.")
                coord.setValue(state, new_angle_rad)

            else:
                ...
    # Пересчитываем состояние
    model.realizePosition(state)

    # Показываем в визуализаторе
    model.getVisualizer().show(state)
    if not flag:
        # time.sleep(5)
        flag = True
    else:
        time.sleep(0.0001)
```

Приложение Д

(справочное)

Программный код для расчета пассивных сил

```
loader = Loader()


script_dir = os.path.dirname(os.path.abspath(__file__))
model_path = os.path.abspath(os.path.join(script_dir, 'model',
f'Hand_Wrist_Model_for_development{loader.postfix}.osim'))
mot_path = os.path.abspath(os.path.join(script_dir, 'model', 'ge-
nered_mot_files', f"S{loader.subject}_E{loader.exercise}_restim{loader.stimu-
lus}_rep{loader.repetition}_v2.mot"))


# === Загрузка модели ===
model = osim.Model(model_path)
state = model.initSystem()
coord_set = model.getCoordinateSet()
muscle_set = model.getMuscles()


# Список интересующих мышц
# muscles_to_plot = ['FCR', 'FCU', 'ECRL', 'ECRB', 'FDSI']

# muscles_to_plot = [muscle_set.get(i).getName() for i in range(mus-
cle_set.getSize())][1:48]
muscles_to_plot = ['EIP', 'EDCI']#, 'LUMI', 'FDSI', 'FDPI']

with open(mot_path, 'r') as f:
    lines = f.readlines()

for i, line in enumerate(lines):
    if line.strip().lower() == 'endheader':
        header_index = i + 1
        break

column_names = lines[header_index].strip().split('\t')

df = pd.read_csv(mot_path, sep='\t', skiprows=header_index + 1, names=col-
umn_names)
time_array = df['time'].values

forces = {m: [] for m in muscles_to_plot}
fiber_lengths = {m: [] for m in muscles_to_plot}
tendon_lengths = {m: [] for m in muscles_to_plot}
musculotendon_lengths = {m: [] for m in muscles_to_plot}
optimal_fiber_lengths = {m: [] for m in muscles_to_plot}

for t_idx in range(len(time_array)):
    row = df.iloc[t_idx]
    t = row['time']

    # Установка координат
    for coord_name in df.columns:
        if coord_name == 'time':
            continue
        if coord_set.contains(coord_name):
            coord = coord_set.get(coord_name)
```

```
            coord.setValue(state, math.radians(row[coord_name]))  # если в
градусах
            # coord.setValue(state, row[coord_name])  # если в радианах

    # Обновление модели
    model.realizePosition(state)
    model.realizeDynamics(state)

    # Расчёт пассивных сил
    for m in muscles_to_plot:
        muscle = osim.Millard2012EquilibriumMuscle.safeDownCast(mus-
cle_set.get(m))
        forces[m].append(muscle.getPassiveFiberForce(state)/100)
        fiber_lengths[m].append(muscle.getFiberLength(state))
        tendon_lengths[m].append(muscle.getTendonLength(state))
        musculotendon_lengths[m].append(muscle.getLength(state))
        optimal_fiber_lengths[m].append(muscle.getOptimalFiberLength())
        if t_idx == 0 and m == "LUMI":
            print(forces[m], fiber_lengths[m], tendon_lengths[m], musculoten-
don_lengths[m], optimal_fiber_lengths[m])

# === Визуализация ===
plt.figure(figsize=(8, 5))

fig, axs = plt.subplots(3, 1, figsize=(12, 10), sharex=True)
fig.suptitle(f'Параметры мышцы:', fontsize=14)
for m in muscles_to_plot:

    axs[0].plot(time_array, forces[m], label=f'Passive Force {m}')
    axs[0].set_ylabel('Сила [Н]')
    axs[0].legend()
    axs[0].grid(True)

    axs[1].plot(time_array, fiber_lengths[m], label=f'Fiber Length {m}')
    axs[1].set_ylabel('Fiber [м]')
    axs[1].legend()
    axs[1].grid(True)

    axs[2].plot(time_array, tendon_lengths[m], label=f'Tendon Length {m}')
    axs[2].set_ylabel('Tendon [м]')
    axs[2].legend()
    axs[2].grid(True)
    axs[2].set_xlabel('Время [с]')

    # остается константой
    # axs[3].plot(time_array, musculotendon_lengths[m], label=f'Muscle-Tendon
Length {m}')
    # axs[3].set_ylabel('L_mt [м]')
    # axs[3].legend()
    # axs[3].grid(True)

    # остается константой
    # axs[4].plot(time_array, optimal_fiber_lengths[m], label=f'Optimal Fiber
Length {m}')
    # axs[4].set_ylabel('L_opt [м]')
    # axs[4].set_xlabel('Время [с]')
    # axs[4].legend()
    # axs[4].grid(True)

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

Приложение Е

(справочное)

Программный код для предварительной настройки модели перед запуском симуляции

```
################################################
# редактирование параметров мышц
################################################
# Настройка путей
os.add_dll_directory("C:/Opensim 4.5/bin")  # Только для Windows
import opensim as osim
from Loader import Loader

loader = Loader()
script_dir = os.path.dirname(os.path.abspath(__file__))

model_path = os.path.join(script_dir, 'model', 'Hand_Wrist_Model_for_develop-
ment.osim')
mot_path = os.path.join(script_dir, 'model', 'genered_mot_files',
                        f"S{loader.subject}_E{loader.exercise}_res-
tim{loader.stimulus}_rep{loader.repetition}.mot")
output_path = os.path.join(script_dir, 'model', f'Hand_Wrist_Model_for_devel-
opment{loader.postfix}.osim')

# Загрузка модели
model = osim.Model(model_path)

model.setUseVisualizer(True)

# Отключение tendon у всех мышц
muscles = model.getMuscles()
state = model.initSystem()
print("*"*100)
print("BEFORE MAKE 1st mot snapshot")
print("*"*100)
for i in range(muscles.getSize()):
    muscle = osim.Millard2012EquilibriumMuscle.safeDownCast(muscles.get(i))
    if muscle:
        t_length = muscle.getTendonSlackLength()
        muscle.setTendonSlackLength(abs(t_length))
    # Полная длина мышцы (сухожилие + волокно)
        l_mt = muscle.getLength(state)
        # Длина волокна
        fiber_length = muscle.getFiberLength(state)
        # Оптимальная длина волокна
        l_opt = muscle.getOptimalFiberLength()
        t_length = muscle.getTendonLength(state)

        print(f"{muscle.getName()}: Lmt={l_mt:.4f}, fiber={fiber_length:.4f},
optimal={l_opt:.4f}, tendon = {abs(t_length):.4f}")

# считывание первого кадра из .mot
motion = osim.Storage(mot_path)
labels = motion.getColumnLabels()
state_vector = motion.getStateVector(0)
data = state_vector.getData()

mot_values = {labels.get(i+1): data.get(i) for i in range(data.size())}
```

```
# Установка координат в state
coord_set = model.getCoordinateSet()
for i in range(coord_set.getSize()):
    coord = coord_set.get(i)
    name = coord.getName()

    if name not in mot_values:
        print(f"[WARNING] Координата '{name}' не найдена в .mot. Пропущена.")
        continue

    value_deg = mot_values[name]
    value_rad = math.radians(value_deg)
    coord.setValue(state, value_rad)
    coord.setDefaultValue(value_rad)

    print(f"[OK] {name}: {value_deg:.4f}° → {value_rad:.4f} rad")

print("*"*100)
print("*"*100)
print("AFTER MAKE 1st mot snapshot")
print("*"*100)
# Обновление положения
model.realizePosition(state)
model.realizeDynamics(state)


for i in range(muscles.getSize()):
    # Получаем длины
    muscle = osim.Millard2012EquilibriumMuscle.safeDownCast(muscles.get(i))
    name = muscle.getName()

    # muscle.computeInitialFiberEquilibrium(state)


    # Полная длина мышцы (сухожилие + волокно) для установленного положения
    l_mt = muscle.getLength(state)
    # Длина волокна для установленного положения
    fiber_length = muscle.getFiberLength(state)
    # Оптимальная длина волокна
    l_opt = muscle.getOptimalFiberLength()
    # сила и длина сухлжилия для установленного положения
    t_length = muscle.getTendonLength(state)
    t_slack_length = muscle.getTendonSlackLength()
    # Исходная дефолтная длина
    def_len = muscle.getDefaultFiberLength()

    # угол пеннации
    pennation_angle = muscle.getPennationAngleAtOptimalFiberLength()
    # минимальная возможное значение активации мышцы
    min_activation = muscle.getMinControl()
    muscle.setDefaultActivation(min_activation)

    # новая дефолтная длина волокна в соответствии с установленным состоянием
"покоя"
    fiber_length_new = (l_mt - t_slack_length)/math.cos(pennation_angle)
    # установка новых параметров
    muscle.setDefaultFiberLength(fiber_length_new)
    muscle.setFiberLength(state, fiber_length_new)
    muscle.setOptimalFiberLength(fiber_length*0.95)

    muscle.set_ignore_tendon_compliance(True)
```

```
model.realizeDynamics(state)


# set = set()

import numpy as np
import matplotlib.pyplot as plt
for i in range(muscles.getSize()):
    # Выбор мышцы
    muscle = osim.Millard2012EquilibriumMuscle.safeDownCast(model.getMus-
cles().get(i))
    passive_force = muscle.getPassiveFiberForce(state)
    print(f"[{muscle.getName()}] Passive force: {passive_force}")


    name = muscle.getName()
    # Настройка состояния: зафиксируем активацию = 0
    muscle.setActivation(state, 1)

    # Получение кривой полной силы от длины волокна
    fiber_curve = muscle.getFiberForceLengthCurve()

    # Отдельно можно получить:
    active_curve = muscle.getActiveForceLengthCurve()
    tendon_curve = muscle.getTendonForceLengthCurve()
    velocity_curve = muscle.getForceVelocityCurve()


    lengths = np.linspace(0.1, 2, 300)

    # Вычисляем значения кривых
    active_vals = [active_curve.calcValue(l) for l in lengths]
    tendon_vals = [tendon_curve.calcValue(l) for l in lengths]
    fiber_vals  = [fiber_curve.calcValue(l) for l in lengths]
    velocity_vals  = [velocity_curve.calcValue(l) for l in lengths]
    passive_force = [fiber_vals[i] - active_vals[i] for i in range(300)]

    # Построение графика
    plt.plot(lengths, active_vals, label="Active Force-Length")
    plt.plot(lengths, tendon_vals, label="Tendon Force-Length")
    plt.plot(lengths, fiber_vals, label="Total Fiber Force-Length", lin-
estyle='--')
    plt.plot(lengths, velocity_vals, label="Velocyty Force")
    plt.plot(lengths, passive_force, label="Total - ACTIVE Force", color =
'pink')

    plt.xlabel("Normalized Fiber Length")
    plt.ylabel("Normalized Force")

    plt.ylim(-1, 2)
    plt.title(f"Force-Length Curves [{name}]")
    plt.legend()
    plt.grid(True)
    # plt.show()


# Отображение
model.getVisualizer().show(state)
input()

model.finalizeFromProperties()
model.printToXML(output_path)
print(f"\n Модель сохранена в: {output_path}")
```

```
##################################################
# Установка актуаторов и запуск static optimization
##################################################


def add_actuators(model, opt_force):
    coord_set = model.getCoordinateSet()
    for i in range(coord_set.getSize()):
        coord_name = coord_set.get(i).getName()
    #    print(coord_set.get(i).getMotionType())
        if coord_set.get(i).get_locked() == False:
            actuator_name = coord_name + "_reserve"
            actuator = osim.CoordinateActuator()
            actuator.setName(actuator_name)
            actuator.setCoordinate(coord_set.get(i))
            actuator.setOptimalForce(opt_force)
            # actuator.setMaxControl(math.inf)
            # actuator.setMinControl(-math.inf)
            actuator.setMaxControl(1)
            actuator.setMinControl(-1)
            # actuator.setMaxControl(0.2)
            # actuator.setMinControl(-0.2)

            model.addForce(actuator)
    state = model.initSystem()
    return model

def static_optimization2(model, ik_res, res_dir, res_name):
    sto = osim.Storage(ik_res)
    time = osim.ArrayDouble()
    sto.getTimeColumn(time)
    sTime = time.get(0)
    fTime = time.get(time.getSize() - 1)

    static_opt = osim.StaticOptimization()
    static_opt.setStartTime(sTime)
    static_opt.setEndTime(fTime)
    static_opt.setUseModelForceSet(True)
    static_opt.setUseMusclePhysiology(True)
    static_opt.setActivationExponent(2)

    muscle_analysis = osim.MuscleAnalysis()

    model.addAnalysis(static_opt)
    model.addAnalysis(muscle_analysis)

    analysis = osim.AnalyzeTool()
    analysis.setModel(model)
    analysis.setInitialTime(sTime)
    analysis.setFinalTime(fTime)
    analysis.setLowpassCutoffFrequency(6)
    analysis.setCoordinatesFileName(ik_res)
    analysis.setResultsDir(res_dir)
    analysis.setName(res_name)
    analysis.setLoadModelAndInput(True)

    analysis.getAnalysisSet().cloneAndAppend(static_opt)
    analysis.getAnalysisSet().cloneAndAppend(muscle_analysis)

    analysis.run()
    print(f"[OK] Static Optimization выполнена. Результаты в: {res_dir}")
```

```
if __name__ == "__main__":

    loader = Loader()

    opt_force = loader.opt_force

    # Получаем путь к текущему скрипту
    script_dir = os.path.dirname(os.path.abspath(__file__))
    model_path = os.path.abspath(os.path.join(script_dir, 'model',
f'Hand_Wrist_Model_for_development{loader.postfix}.osim'))
    model = osim.Model(model_path)
    model_act = add_actuators(model, opt_force)

    ik_res = os.path.abspath(os.path.join(script_dir, 'model', 'ge-
nered_mot_files', f"S{loader.subject}_E{loader.exercise}_restim{loader.stimu-
lus}_rep{loader.repetition}_v2.mot"))
    res_dir = os.path.abspath(os.path.join(script_dir, 'so_res'))
    res_name = f"S{loader.subject}_E{loader.exercise}_stim{loader.stimu-
lus}_rep{loader.repetition}_static_opt{opt_force}{loader.postfix}"

    static_optimization2(model_act, ik_res, res_dir, res_name)
```

Приложение Ж

(справочное)

Сравнительная таблица получившихся длин сухожилий и мышечных волокон в модели и в реальности

Имя мышцы в модели	F/M в модели (отношение длины мышечного волокна к общей длине)	Анатомическое название	F/M из литературы
ECRL	0.2102	Extensor carpi radialis longus	0.22
ECRB	0.2158	Extensor carpi radialis brevis	0.22
ECU	0.2368	Extensor carpi ulnaris	0.24
FCR	0.1726	Flexor carpi radialis	0.18
FCU	0.1167	Flexor carpi ulnaris	0.17
PL	0.1581	Palmaris longus	0.16
FDSL	0.1356	Flexor digitorum superficialis (lateral)	0.18
FDSR	0.1922	Flexor digitorum superficialis (ring)	0.18
FDSM	0.204	Flexor digitorum superficialis (middle)	0.18
FDSI	0.2357	Flexor digitorum superficialis (index)	0.18
FDPL	0.1994	Flexor digitorum profundus (lateral)	0.2
FDPR	0.2157	Flexor digitorum profundus (ring)	0.2
FDPM	0.2152	Flexor digitorum profundus (middle)	0.2
FDPI	0.1962	Flexor digitorum profundus (index)	0.2
EDCL	0.2037	Extensor digitorum communis (lateral)	0.21
EDCR	0.1889	Extensor digitorum communis (ring)	0.21
EDCM	0.2134	Extensor digitorum communis (middle)	0.21
EDCI	0.1904	Extensor digitorum communis (index)	0.21
EDM	0.2076	Extensor digiti minimi	0.21
EIP	0.2692	Extensor indicis proprius	0.27