ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
АВТОМОБИЛЬНО-ДОРОЖНЫЙ ИНСТИТУТ

А.В. Химченко, Н.И. Мищенко

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ТЕХНИЧЕСКИХ СИСТЕМ

УЧЕБНОЕ ПОСОБИЕ

Горловка
2018

УДК 004.942(075)
Х 469

Рекомендовано к изданию решением учебно-методической комиссии факультета «Дорожно-транспортный» Автомобильно-дорожного института ГОУВПО «Донецкий национальный технический университет».
Протокол № 5 от 16.05.2018.

Рецензенты:
Вовк Л. П. — доктор технических наук, профессор, заведующий кафедрой «Математическое моделирование» АДИ ГОУВПО «ДОННТУ»
Самисько Д. Н. — кандидат технических наук, доцент кафедры «Транспортные технологии» АДИ ГОУВПО «ДОННТУ»

Химченко, А.В.
Х 469 Компьютерное моделирование технических систем : учебное пособие [Электронный ресурс] / А.В. Химченко, Н.И. Мищенко. — Горловка: АДИ ГОУВПО «ДонНТУ», 2018. — 93 с.

Учебное пособие предназначено для студентов технических специальностей, обучающихся по программам магистратуры и специалитета. Целью пособия является оказание помощи студентам в освоении компьютерного моделирования технических систем на примерах конкретного использования моделирования для решения практических задач. Целевой аудиторией являются студенты, для которых компьютерное моделирование является прикладной задачей. Это обуславливает выбор в качестве инструмента моделирования программных продуктов Mathworks Matlab и Simulink.

В данной редакции пособия рассматривается имитационное моделирование в среде Simulink с помощью инструмента Simscape Multibody и применение для моделирования простейших нейронных сетей с использованием Matlab.

Пособие предназначено для обучающихся по направлениям подготовки: 23.04.03 «Эксплуатация транспортно-технологических машин и комплексов» и 08.04.01 «Строительство» квалификационного уровня «магистр», специализирующихся в области автомобильного транспорта и дорожного строительства, а также 23.05.01 «Наземные транспортно-технологические средства».

Текст изложен в авторской редакции.

УДК 004.942(075)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Научные исследования современном мире должны проводиться достаточно быстро и качественно для внедрение результатов в промышленность и другие сферы жизнедеятельности человека. Этого требует достаточно жёсткой конкуренции между различными корпорациями, государственными и частными предприятиями.

Компьютерное моделирование это тот инструмент, который существенное ускоряет процессы разработки, создания и исследования различных технических объектов. Кроме того, имитационное моделирование дает возможность ответить на вопросы, для которых классические подходы, выражающиеся в натурном эксперименте или невозможны, или слишком затратны. Такие условия создают особую важность применения компьютерного моделирования в научных исследованиях и модельно-ориентированного проектирования при разработке и отладке работы сложных технических систем.

Современный специалист, учёный или инженер должен обладать умениями и навыками работа с современным инструментом. Важным инструментом в научной и инженерной деятельности является программное обеспечение, позволяющее быстро и качественно решать различного рода задачи.

Одним из таких инструментов являются системы имитационного моделирования, как например, Simulink. Поэтому в курсе компьютерного моделирования технических систем рассматриваются приёмы моделирования с использованием Mathworks Matlab и Simulink.

Данное учебное пособие имеет основную задачу облегчить освоение студентами применения методов математического компьютерного моделирования, создания имитационных моделей, поиска оптимальных решений и так далее. Пособие не является учебником по работе в Matlab, хотя и содержит некоторые сведения находящиеся в справочной документации по Matlab и Simulink [2], но только в той мере, в которой они необходимы для пояснения решения тех или иных задач.

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ В SIMULINK SIMSCAPE MULTIBODY

1.1 Принципы и особенности физического моделирования с использованием Simscape

Моделирование в Simscape имеет те же основные принципы, что и моделирование в Simulink. Особенностью Simscape является моделирование физических элементов с описанием их физических свойств. Инженеру и исследователю нет необходимости подробно описывать физические явления, по крайней мере, если они не отличается от моделей предусмотренных блоками Simscape. Есть возможность сконцентрироваться на описании взаимосвязей, учёте геометрии и материалов объектов и, на основе более простых действий, получить сложные математические модели. Simscape самостоятельно составит математические уравнения и проведёт поиск путей их решения при моделировании.

1.1.1 Представление физических моделей в Simscape

Основой компьютерной симуляции физических систем являются математические модели. Модели представляют собой систему уравнений. Это могут быть дифференциальные, дифференциально-алгебраические и алгебраические системы уравнений.

Как правило математическое представление физической системы содержит обыкновенные дифференциальные уравнения (ОДУ — ODEs), алгебраические уравнения или и то, и другое. В зависимости от наличия тех или иных уравнений системы можно разделить на несколько типов:

1. Система с ОДУ, которые определяют скорости изменения системных переменных и содержат некоторые или все производные системных переменных по времени. Без алгебраических ограничений система является дифференциальной (ОДУ).

2. Система алгебраических уравнений, которые определяют функциональные ограничения среди системных переменных, но не содержат временных производных системных переменных. Без ОДУ система является алгебраической.

3. Система с ОДУ и алгебраическими ограничениями — система смешанно-дифференциально-алгебраическая (DAE).

В зависимости от наличия в различных уравнениях системная переменная является дифференциальной или алгебраической. Дифференциальной

переменную считают если она появляется в форме производной по времени в системных уравнениях.

При представлении модели Simscape необходимо определить дискретные и непрерывные компоненты, которые могут прерваться во время моделирования. Эти компоненты будут определять тип системы дифференциальных уравнений.

Следует помнить что физическая модель Simscape представляет собой сеть из различных компонентов. Это могут быть готовые блоки Simscape и пользовательские блоки, написанные на языке Simscape. Физическая модель также может содержать математическую модель Simulink для описания, например взаимодействия между отдельными элементами, законов изменения силы воздействия на физическую модель. Следует помнить и учитывать то, что физическая модель оперирует переменными, имеющими единицы измерения, а математическая модель в Simulink только численными значениями. Таким образом, при совместном использовании в моделировании различных моделей Simscape и Simulink необходимо приведение единиц к одинаковым размерностям.

При составлении дифференциальных уравнений, описывающих модель, в зависимости от условий и связей, могут быть получены жёсткие системы дифференциальных уравнений. В определённых случаях их решения могут привести к колебаниям. Изменение параметров моделирования, выбор точности решения, выбор решателя и т. д. позволяют получить правильное качественное решение системы и определить адекватное поведение модели в сложных переходных режимах.

Моделирование в Simscape и Simulink поддерживает принципы дискретно-событийного моделирования. В основном моделирование физических систем и моделирование в Simulink осуществляется на принципах плавного изменения тех или иных переменных. Однако некоторые переменные являются дискретными, а закон изменения непрерывных переменных так же может изменяться дискретно. То есть фактически в модель вводятся триггеры (переключатели), которые в зависимости от условия (события) изменяют состав системы уравнений, определяющих модель. Это позволяет задавать и кусочные функции.

То есть события являются прерывистыми изменениями состояний системы или ее динамики по мере того, как система функционирует во времени: например, открытие клапана или жёсткий упор.

Особый тип события — это пересечение нуля. Он представляет смену знака математической функции. Переменные и шаговые решатели принимают более мелкие шаги, когда обнаруживают пересечения нуля. Меньшие шаги

помогают зафиксировать динамику, которая вызывает пересечение нуля, но они также значительно замедляют симуляцию.

В Simscape и Simulink имеется возможность отключение обнаружения перехода через 0 на отдельных блоках или на всей модели. Это обнаружение часто улучшает точность моделирования, но может существенно замедлить его скорость.

Matlab предлагает различные методы обнаружения и анализа пересечения нуля. Их применение поможет добиться правильного баланса между скоростью и точностью моделирования.

Так применение алгоритма адаптивного определение пересечения 0, позволяет уменьшить количество таких пересечений. Это, соответственно, позволяет избежать преждевременной остановки симуляции.

Настройка применения этого алгоритма находится в окне конфигурация модели в разделе Solver (рис. 1.1).

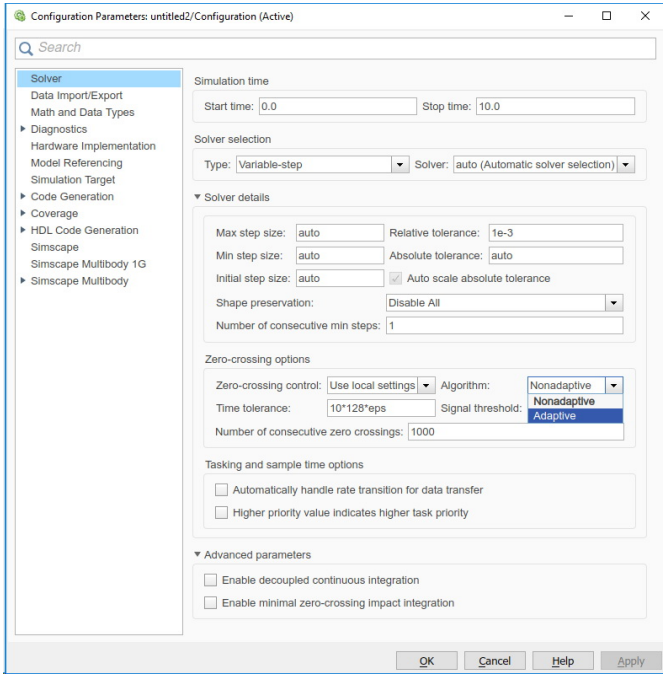


Рисунок 1.1 — Окно настройки параметров решателя модели

1.1.2 Принципы работы физического имитационного моделирования в Simscape

Имитационная модель выполняется в виде схемы (блок-схемы, кинематической схемы и т. д), объединяющей определённым образом различные блоки. Процесс моделирования имеет следующие основные этапы:

— проверка модели;
— построение сети;
— создание системы уравнений;
— определение начальных условий;
— инициализация переменных во времени;
— решение системы уравнений во времени.

На начальном этапе Simscape проверяет конфигурацию модели и элементы данных из блока диалоговые окна.

Проверка осуществляется по следующим признакам:

1. Все блоки Simscape в схеме должны быть подключены в одной или нескольких физических сетях.

2. Каждая топологически отличная физическая сеть на диаграмме требует только одного блока конфигурации решателя.

3. Если модель содержит гидравлические элементы, каждая топологически отличная гидравлическая схема на диаграмме должна соединяться с блоком пользовательских жидкостей  (или блоком жидкости, доступным с библиотеками блоков Simscape Fluids™). Эти блоки определяют свойства текучей среды, которые действуют как глобальные параметры для всех блоков одного гидравлического контура. Если в сети не установлен блок жидкости, гидравлические блоки в этом контуре сети используют жидкость по умолчанию. Более одного блока жидкости в контуре является ошибкой.

4. Если модель содержит газ, то свойства газа по умолчанию — это свойства сухого воздуха. Если подключается блок *Свойства газа*  к топологически собственному контуру, можно изменить свойства газа для всех блоков сети. Более чем один блок *Свойства газа* в контуре сети недопустим.

5. Сигнал единиц, указанных в блоке преобразователя Simulink-PS, должен соответствовать входному типу, ожидаемому блоком Simscape. Аналогичным образом единицы, указанные в блоке преобразователя PS-Simulink должны соответствовать типу физического сигнала, ожидаемого на выходе.

После проверки модели решатель Simscape создаёт физическую сеть на основе следующих принципов:

1. Два напрямую связанных порта имеют одинаковые значения для всех их переменных.

2. Любая переменная, передаваемая по линии физического соединения, при наличии ветвления делится между несколькими компонентами, связанными этими ветвями. Для каждой переменной сумма всех ее значений, втекающих в точку ветвления, равна сумме всех ее вытекающих значений.

На основе конфигурации сети, известных глобальных переменных, свойств жидкостей и тел решатель составляет систему уравнений. После анализа системы уравнений переменные, которые не участвуют в получении решения, отбрасываются.

Если в конфигурации отсутствуют или не были установлены начальные условия моделирования из стационарного состояния, то решатель вычисляет начальные условия для всех системных переменных, которые удовлетворяют всем модельным уравнениям. Удовлетворение условиям осуществляется с учётом приоритета уравнений и допускаемой погрешности.

1.1.3 Основные этапы построения физической модели в Simscape и работы с ней

Рассматривая основные этапы построение физической модели с помощью инструмента Simscape Simulink, компания Mathworks выделяет следующие основные ключевые шаги:

Шаг 1. Создание новой модели интерактивно или с использованием `ssc_new`.

Шаг 2. Сборка физической сети.

Шаг 3. Настройка параметров блоков и целевых переменных.

Шаг 4. Добавление источников сигналов.

Шаг 5. Добавление датчиков.

Шаг 6. Подключение Simscape-диаграмм к источникам и областям Simulink с интерфейсными блоками.

Шаг 7. Запуск имитации модели.

Шаг 8. Просмотр и анализ результатов моделирования.

Шаги отражают рабочий процесс не только построения, но и получения результатов моделирования. Применительно к примеру простейшей физической модели основные шаги рабочего процесса показаны на рисунке 1.2.

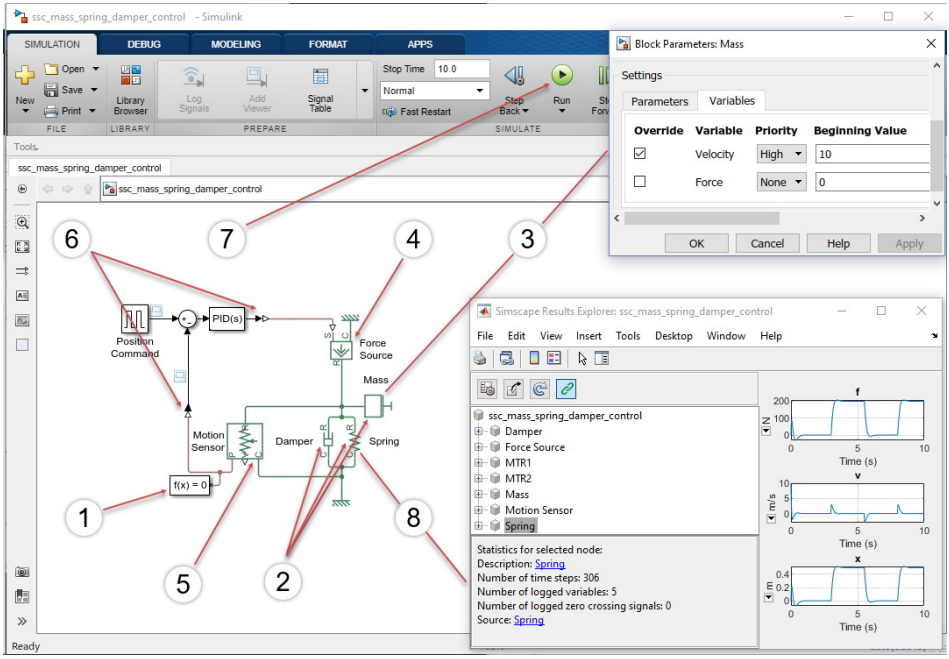


Рисунок 1.2 — Основные шаги рабочего процесса имитационного моделирования в Simscape

1.2 Моделирование с помощью Simscape Multibody (твердотельное моделирование)

Simscape™ Multibody™ представляет собой инструмент Simulink для твердотельного моделирования трёхмерные механических систем. Он может быть интересен для моделирования работы подвески транспортного средства, строительной техники и тому подобное. Инструмент имеет в своём составе блоки, позволяющие моделировать: трёхмерные тела и их взаимосвязи, преобразовывать системы координат и применять ограничения, прикладывать силы и устанавливать датчики.

Matlab позволяет импортировать в Simscape Multibody готовые 3D модели, подготовленные в системах твердотельного моделирования. При этом в модель переносятся как параметры отдельных деталей, так и виды их соединений. При моделировании Matlab визуализирует поведение объекта

твердотельного моделирования и при желании пользователя записывает это в видеофайл.

На данный момент имеется возможность импортировать модели CAD, URDF и Robotics System Toolbox. Модели должны удовлетворять требованиям к формату в зависимости от родительского программного обеспечения. Так модели САПР должны быть в подходящем формате XML. Для моделей сборки САПР используется плагин Simscape Multibody Link CAD для Autodesk® Inventor®, Creo™ Parametric или SolidWorks® моделей.

Могут импортироваться и отдельные файлы в формате step или stl не зависимо от родительского программного обеспечения [6] с дальнейшим определением видов их соединений уже в Simscape Multibody.

1.2.1 Основы создания модели

При запуске Simulink предлагает ряд шаблонов. Выберем раздел Simscape. Из предложенных шаблонов Simscape (рис. 1.3) выберем шаблон в Multibody.

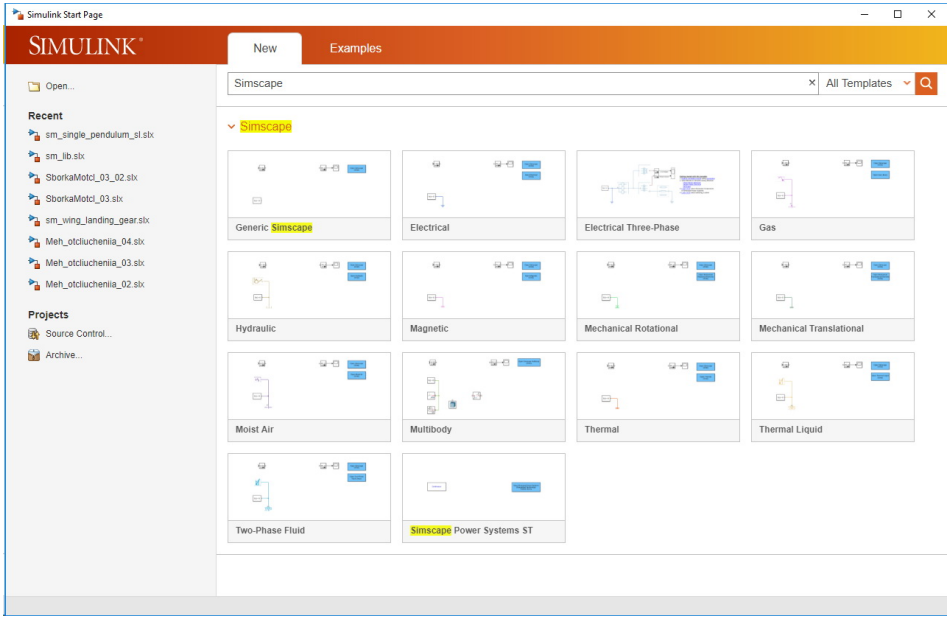


Рисунок 1.3 — Выбор шаблона Simscape

В результате мы получаем базовые блоки Simscape Multibody в составе модели (рис. 1.4). В данном случае это тот минимум, который необходим для реализации модели. Итак рассмотрим состав обязательных блоков и их базовые возможности на примере построения простейшей модели.

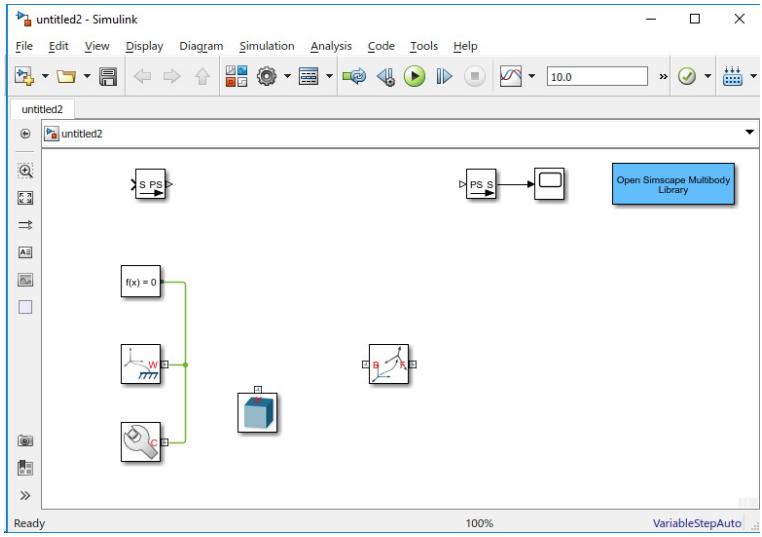


Рисунок 1.4 — Шаблон Simscape Multibody

Одним из основных элементов является *«Solid»* (Твёрдое тело)  . Данный блок представляет собой единое твердое тело. Он определяет геометрию, инерцию и массу тела.

Свойства блока позволяют делать очень широкие его настройки. В простейших моделях может быть определена геометрия блока в виде простейших трехмерных объектов: параллелепипеда, шара, цилиндра, эллипсоида, куба, призмы или тела, полученного вращением простейшей геометрической фигуры на некоторый угол (рис. 1.5). При наличии трёхмерной модели, подготовленной в каком-либо графическом редакторе, данный файл может быть прикреплён и использован в моделировании.

Кроме того, в свойствах твёрдого тела также определяется плотность материала, которая может быть использована для расчёта инерции из геометрии твёрдого тела. Для определения инерции есть варианты задания массы тела, для последующего вычисления инерционных характеристик, или зада-

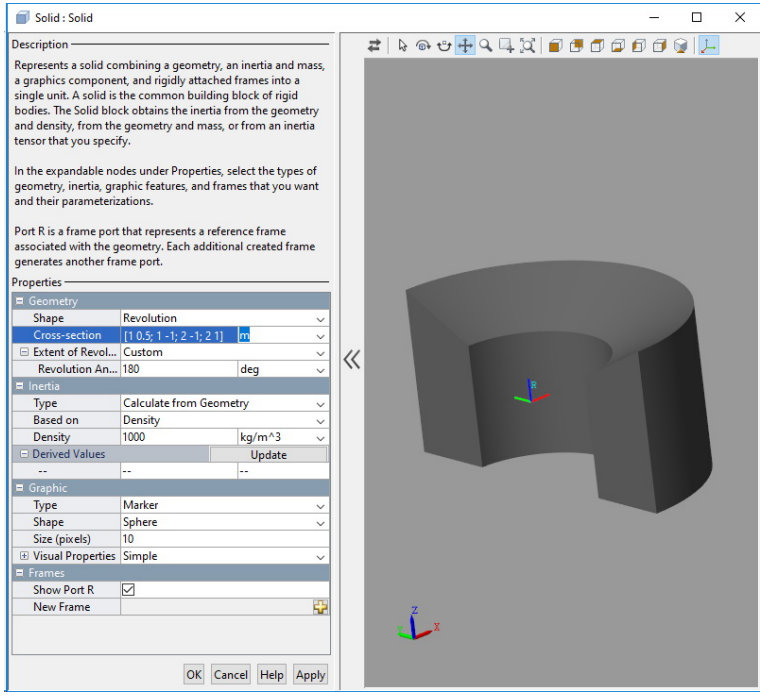


Рисунок 1.5 — Свойства твердого тела

ние тензора инерции. Тело может быть представлено и как материальная точка.

Отдельный раздел отвечает за отображение геометрии твёрдого тела в модели при обрисовке в процессе анимации. Возможны варианты отображения в виде маркера или в виде тела на основе его геометрии.

При изменении геометрических параметров твёрдого тела клавиша «F5» позволяет обновить изображение в окне свойств. При необходимости инструментальная панель позволяет выбрать необходимый ракурс и масштаб отображения.

Система координат твёрдого тела может определяться через порт и согласовываться с общей системой координат на входе. Кроме того системы координат могут быть настроены индивидуально и выбираться в зависимости от задач. Здесь следует обратить внимание на то, что в Simulink по умолчанию ось X направлена вправо, ось Z — вверх, ось Y — за экран. То есть оси X и Y являются горизонтальными, а ось Z — вертикальной.

При выборе параметров в окне свойств тела следует обращать внимание на единицы измерения.

Для возможности выполнения расчётов модель должны в обязательном порядке быть внесены такие блоки: как  — *World Frame* (глобальная система координат),  — *Solver Configuration* (конфигурация решателя) и  — *Mechanism Configuration* (конфигурация механизма). Эти блоки должны быть связаны между собой.

Глобальная система координат является неинерциальной и не требует каких-либо настроек.

Блок параметров конфигурация решателя (рис. 1.6) позволяет определить такие параметры для симуляции как:

— начало моделирования из устойчивого состояния механизма;
— допуск согласованности;
— использование локального решателя;
— количество итераций в режиме реального времени;
— метод решения уравнений линейной алгебры;
— способ задания гармонических функций;
— буфер памяти при задержках;
— фильтрацию для объектов из библиотек предыдущих версий.

Когда этот флажок *«Начать симуляцию из устойчивого состояния»* выбран, решатель пытается найти устойчивое состояние, которое может возникнуть, если входы в систему поддерживались постоянными в течение достаточно большого времени, начиная с начального состояния. Затем начинается симуляция с этого устойчивого состояния.

Параметр *«Допуск согласованности»* влияет на нелинейный решатель, используемый для вычисления начальных условий и для временной инициализации. Он определяет, насколько точно должны выполняться алгебраические ограничения в начале моделирования и после каждого дискретного события (например, разрыв, возникающий в результате открытия клапана, жёсткая остановка и т. д.). Чтобы получить более точное и надёжное моделирование параметр следует уменьшить. Однако это увеличит время вычисления.

Использование локального решателя позволяет настраивать моделирование на основе выборки. В этом случае все состояния физической схемы, которые в остальном являются непрерывными, становятся представленными Simulink в виде дискретных состояний.

В зависимости от задач, могут понадобиться и другие настройки.

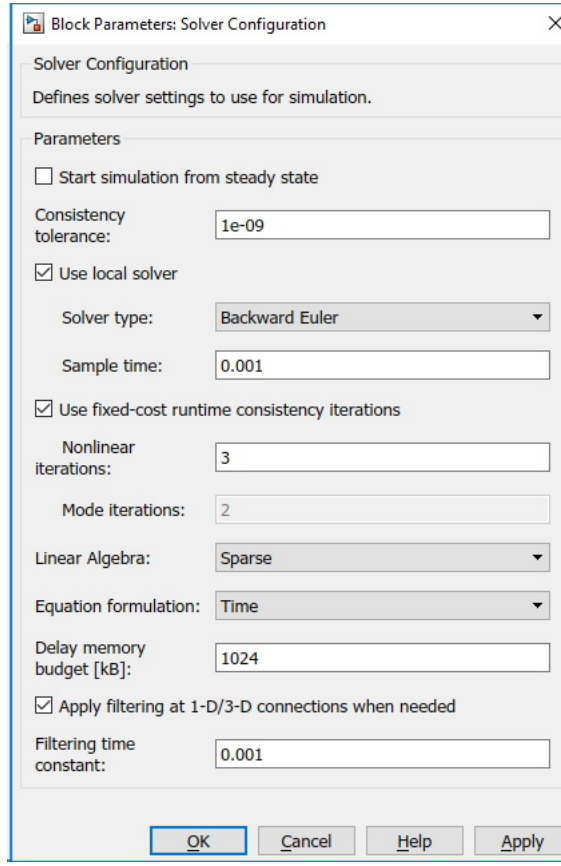


Рисунок 1.6 — Параметры конфигурации решателя

Использование блока *«Solver Configuration»* является обязательным, так как перед выполнением симуляции должны быть определены требования к решению. Порт блока может быть подключён в любой точке модели.

Блок конфигурации механизма (*«Mechanism Configuration»*) определяет параметры гравитации, воздействующей на механизм, и ошибку линеаризации для вычисления численных частных производных. Гравитация может быть задана несколькими способами. При выборе параметры *None* — гравитация отсутствует. В случае выбора постоянной гравитации (*Constant*) необходимо указать гравитационное ускорение как вектор, который остаётся

неизменным в пространстве и во времени. Он определяется проекцией на оси в декартовой системе координат (рис. 1.7).

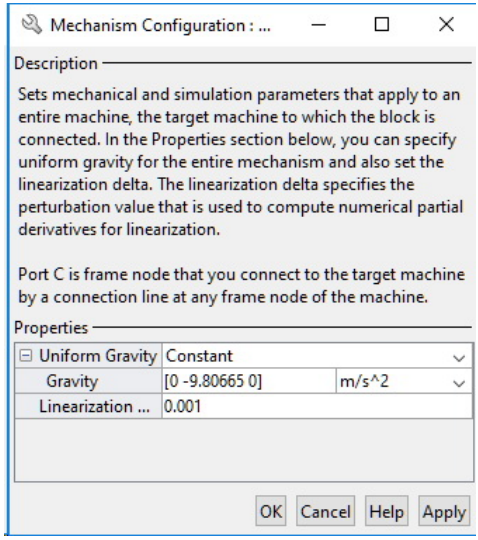


Рисунок 1.7 — Параметры блока конфигурации механизма

Есть возможность также задать гравитацию переменную во времени. Для этого следует выбрать параметр *Time-Varying*. В этом случае в блоке появится вход, который может быть использован для определения изменения гравитации во времени.

При задании вектора гравитации следует помнить, расположение осей в Matlab Simulink — $[XYZ]$. Вектор гравитации по умолчанию [0 0 –9.80665]. То есть направление к центру Земли определяется по оси $-Z$.

Создадим модель простейшего математического маятника.

Модели будет состоять из трёх деталей: пальца, относительно оси которого должен колебаться маятник; тонкого стержня, на котором закреплён груз; груза в виде шара.

Пусть палец закреплён неподвижно в глобальной системе координат. Стержень соединён с шаром шарниром, позволяющим совершать вращение вокруг пальца. А шар жёстко соединён со стержнем, например стержень вкручен в шар.

Палец фактически является цилиндром. При создании цилиндра ось Z располагается вдоль него. Для того, чтобы обеспечить колебания маятника относительно продольной оси пальца его придётся переориентировать.

Для этого нам понадобится блок преобразования координат  *«Rigid Transform»*. Этот блок производит независимые от времени преобразования координат между двумя системами координат. Например между глобальной и локальной или двумя локальными системами координат. Таким образом, он позволяет согласовать положение различных тел в пространстве.

Так как справочная система Matlab не русифицирована и имеется возможность использовать лишь автоматический перевод последней актуальной справочной информации на сайте https://www.mathworks.com/help/, пользователи в процессе создания модели неизбежно и многократно будут сталкиваться с понятием «Frame»[1].

Данное слово имеет много близких и сходных значений, но в контексте применения в Simscape «frame» следует понимать как систему координат (локальную или глобальную) или точку тела со своей локальной системой координат.

Речь идёт об ортогональных декартовых системах координат  , которые определяют положение и ориентацию данных в твердотельных 3–D моделях. Каждая система состоит из трёх перпендикулярных осей с общим началом. Начало определяет положение системы координат, а положение осей определяет ориентацию системы. Оси имеют цветовую маркировку. Ось x отображается красным, ось y зелёным, а ось z синим цветами. По умолчанию оси x и y лежат в горизонтальной плоскости, а ось z направлена вертикально. Собственно и преобразования происходят с этими локальными системами координат.

Определение места соединения деталей определяется местоположением точек присоединения, то есть положением локальной системы координат точки контакта относительно другой локальной системы или относительно глобальной системы координат.

Блок *«Rigid Transform»* позволяет осуществлять вращение системы координат, ее осей, и перемещение начала координат. Параметр *Rotation* позволяет выбрать метод поворота. Параметр *Translation* — метод перемещения.

Параметр *Rotation* может иметь следующее значения:

— None — отсутствие преобразования;

— Aligned Axes — выравнивание осей совмещением осей по выбору;

— Standard Axis — вращение относительно заданной оси на заданный угол;

— Arbitrary Axis — определяет новое положение системы координат вращением на углы, заданные матрицей, вокруг соответствующих осей $[x, y, z]$;

[1] Frame — рамка, кадр, рама, каркас, структура, оправа, система, конструкция, остов, станина, тело

— Rotation Sequence — определяет систему координат вращением как последовательность трех элементарных вращений на разные углы (преобразование Эйлера);

— Rotation Matrix — преобразование происходит с помощью матрицы вращения между базовой и конечной системой координат. Матрица должна быть ортогональная с определителем +1. По умолчанию используется матрица — [1 0 0; 0 1 0; 0 0 1].

В нашем случае удобно применить метод Aligned Axes, так как фактически необходимо ось Z направить вдоль оси Y (рис. 1.8). Ось X оставим без изменений.

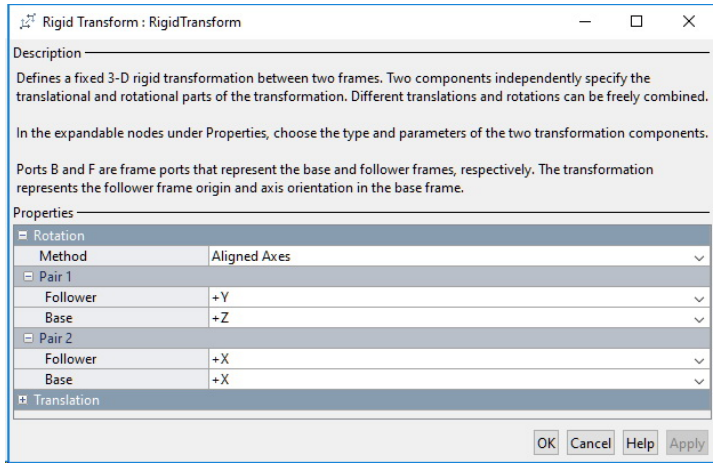


Рисунок 1.8 — Параметры блока преобразования координат

Так как для получения положения пальца перемещать его систему координат нет необходимости, она находится в центре пальца на его оси, перенесём рассмотрение процесса перемещения системы координат несколько ниже.

Для обеспечения колебания маятника относительно пальца, необходим шарнир. Библиотека шарниров Simscape Multibody достаточно разнообразна (рис. 1.9). Она включает в себя шарниры с различными степенями свободы (табл. 1.1), то есть различными ограничениями перемещений. Выбор шарнира зависит от типа механизма, который пытается моделировать инженер-исследователь. Фактически тип шарнира должен повторять тип механизма.

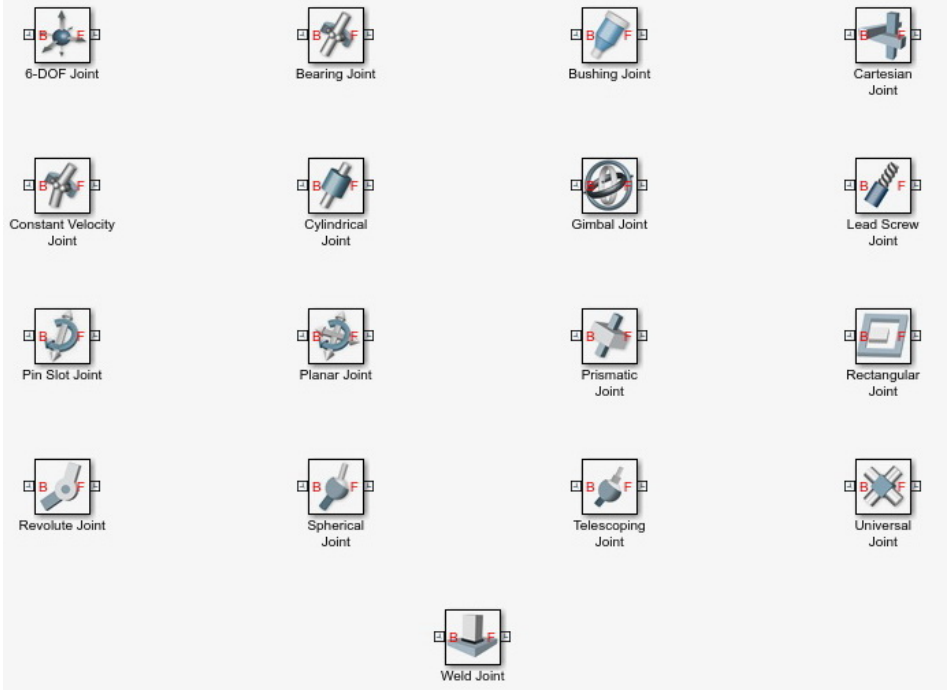


Рисунок 1.9 — Библиотека шарниров Simscape Multibody

Работая с шарнирами, следует помнить, что ограничение степеней свободы осуществляется относительно конкретных базовых осей. В отдельных случаях речь идёт не о конкретных перемещениях, а о видах движения, поэтому используется понятие примитивов[2].

В шарнирах Simscape Multibody используются следующие примитивы:

— призматический или поступательный — допускает перемещаться вдоль осей x, y или z. Обозначается буквой P с индексом соответствующей оси (P_x, P_y, P_z);

— вращательный — допускает вращения вокруг одной из осей x, y или z. Обозначается буквой R с индексом соответствующей оси (R_x, R_y, R_z);

— сферический допускает вращения вокруг любой 3–D оси, $[x, y, z]$. Обозначается буквой S;

[2]Примитив — это вид движения, представляющий элементарное перемещение в виде: вращения относительно оси, поступательного движения вдоль оси, совместного связанного поступательного и вращательного движения и так далее.

Таблица 1.1 — Ограничения степеней свободы, которые обеспечивают различные блоки шарниров

Шарнир	Перемещение	Вращение	Всего
6-DOF Joint	3	3	6
Bearing Joint	1	3	4
Bushing Joint	3	3	6
Cartesian Joint	3	0	3
Constant Velocity Joint	0	2	2
Cylindrical Joint	1	1	2
Gimbal Joint	0	3	3
Leadscrew Joint	1 (связанная)		1
Pin Slot Joint	1	1	2
Planar Joint	2	1	3
Prismatic Joint	1	0	1
Rectangular Joint	2	0	2
Revolute Joint	0	1	1
Spherical Joint	0	3	3
Telescoping Joint	1	3	4
Universal Joint	0	2	2
Weld Joint	0	0	0

— примитив ходового винта — допускает связанное вращательно-поступательное движение относительно одной из осей, например z. Обозначается LS;

— примитив равенства угловых скоростей — допускает вращение на постоянной скорости, соединённых под произвольным углом валов. Обозначается буквами CV. Это примитив шарнира равных угловых скоростей.

В таблице 1.2 приведены примитивы движений для разных шарниров доступных в библиотеке (рис. 1.9).

По своей сути работа со всеми шарнирами аналогична. В выбранном нами примере необходим шарнир, который имеют только одну степень свободы — допускает вращения относительно оси пальца. Это *«Revolute Joint»*. Соответственно рассмотрим соединение деталей на основе этого шарнира, его настроек и правильной ориентации.

Согласно справочной системе Matlab Simulink шарнир *«Revolute Joint»* позволяет вращать тело относительно оси Z системы координат, присоединённой к базовому входу шарнира. То есть при соединении шарнира необходимо обратить внимание на положение оси Z в базовой для него в системе координат. Располагая палец маятника, ось Z была направлена горизонтально, а система координат развёрнута относительно оси X. То есть система ко-

Таблица 1.2 — Примитивы, по которым допускается движение в шарнире

Шарнир	Примитивы шарнира								
6-DOF Joint	P_x	P_y	P_z				S		
Bearing Joint			Pz	Rx	R_y	R_z			
Bushing Joint	P_x	P_y	P_z	R_x	R_y	R_z			
Cartesian Joint	P_x	P_y	P_z						
Constant Velocity Joint								CV	
Cylindrical Joint			P_z			R_z			
Gimbal Joint				R_x	R_y	R_z			
Leadscrew Joint									LS_z
Pin Slot Joint	P_x					R_z			
Planar Joint	P_x	P_y				R_z			
Prismatic Joint			P_z						
Rectangular Joint	P_x	P_y							
Revolute Joint						R_z			
Spherical Joint							S		
Telescoping Joint			P_z				S		
Universal Joint				R_x	R_y				
Weld Joint									

ординат, в которой находится палец, как раз и соответствует необходимым условиям. Не производя преобразований, мы имеем возможность присоединить шарнир *«Revolute Joint»* в сеть в месте соединения пальца. Таким образом шарнир будет иметь с пальцем одинаковую ориентацию. Шарнир необходимо присоединять портом «B», как показано на рисунке 1.10.

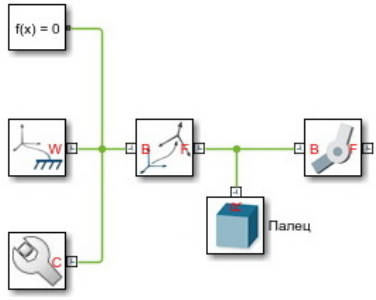


Рисунок 1.10 — Присоединение шарнира *«Revolute Joint»* к модели

Для следующей детали поворот системы координат может не понадобиться. Шарнир находится в преобразованной локальной системе. В этом

случае следует учитывать положение новой локальной системы координат при дальнейших преобразованиях или провести обратные преобразования, определив точку дальнейшего подключения в глобальной системе координат.

Для возвращения к базовой системе координат после шарнира добавим в сеть блок преобразования координат *«Rigid Transform»*, расположив его в зеркальном отражении. То есть портом *Follower* подключим к порту *Follower* шарнира. Зададим в параметрах преобразования обратные, выполненным ранее. В результате получим то, что порт *Base* блока *«Rigid Transform»* будет находиться в начале базовой глобальной системы координат.

Настроим шарнир *«Revolute Joint»* для получения желаемого результата. Откроем окно параметров шарнира двойным щелчком по иконке или через контекстное меню через правую клавишу мыши.

Параметры данного шарнира имеют в настройках два раздела: *Revolute Primitive* и *Composite Force/Torque Sensing*. В зависимости от типа шарнира и количества степеней свободы количество разделов типа *Revolute Primitive* может быть различным. Этот раздел настроек определяет целевые параметры положение равновесия, внутреннюю механику шарнира и силы и моменты, активирующие данные шарнир. Кроме того, в этом разделе находятся датчики фиксирующие кинематические параметры шарнира.

Выбирая параметры *Specify Position Target* (начальное положение) и *Specify Velocity Target (Начальная скорость)* имеется возможность задавать положение механизма в начальный момент моделирования, в момент времени равный нулю. Вместе с начальными условиями задаётся и приоритет их выполнения. Следует помнить, что при высоком приоритете выполнения различных условий система уравнений может не найти решения. Для того, чтобы маятник под действием силы тяжести начал движение, зададим в качестве начальных условий угол поворота шарнира относительно оси Z равный 45 градусам.

В разделе *Internal Mechanics (Внутренняя механика)* определяются параметры взаимодействия между собой отдельных частей шарнира. Здесь имеется возможность в случае наличия упругого элемента в шарнире установить его жёсткость и положение равновесия, а при наличии демпфирующих свойств, силы трения установить коэффициент линейного затухания (коэффициент демпфирования). Наличие коэффициента демпфирования позволяет определить внутреннее сопротивление шарнира взаимному перемещению его частей. Это может позволить создать условия движения в шарнире с постоянной скоростью.

Зададим коэффициент демпфирования. Подбор и его значения позволит получить затухающие колебания математического маятника. Полученные настройки шарнира показаны на рис. 1.11.

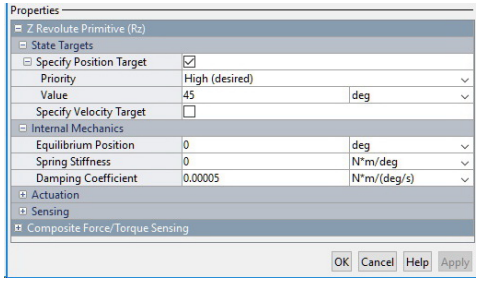


Рисунок 1.11 — Параметры шарнира *«Revolute Joint»*

Раздел *Actuation (Привод)* оставим по умолчанию. В этом случае параметры движения и усилия в шарнире будут рассчитываться в результате моделирования. Если к шарниру приложена сила или крутящий момент, то в данном разделе можно активировать порт, через который будут подаваться значение в данном случае крутящего момента. В отдельных случаях при моделировании необходимо задать какое-либо движение. Оно может быть задано с помощью внешних сигналов через порты *Привода*, которые также активируются в этом разделе. Движение задаётся значением перемещения или угла поворота.

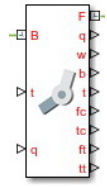
В разделе *Sensing* при необходимости можно активировать порты, для получения информации о положение шарнира, скорости, ускорений и приводных сил и моментов, приложенных в шарнире для обеспечения заданного перемещения или поворота в порту *Actuation*. В отличие от следующего раздела параметров датчик момента показывает то значение, которое подаётся в порт — приводной момент, или рассчитывается из условия обеспечения заданного перемещения или поворота. В других шарнирах в разделах настроек поступательных примитивов на месте датчика приводного момента определён датчик приводных сил. Это усилия внутреннего взаимодействия отдельных деталей шарнира, или внешние силы и моменты, приложенные к ним, рассчитанные по параметрам *Actuation*.

Реакции взаимодействия деталей в шарнире получают, активируя порты в разделе *Composite Force/Torque Sensing*.

Раздел *Composite Force/Torque Sensing* открывает порты для получение информации о реакциях в шарнире и суммарных силах: силах, учитывающих сопротивление, упругое взаимодействие, усилия активации и силы, действующие при наличии примитивов, определяющих другие степени свободы.

Блок шарнира *«Revolute Joint»* с активированными входами и датчиками показан на рис. 1.12.

Входы: B — базовая деталь; t — крутящий момент активации; q — изменения угла поворота шарнира.



Выходы: F — последующая деталь; q — изменения угла поворота шарнира; w — угловая скорость в шарнире; b — угловое ускорение в шарнире; t — крутящий момент активации; fc — силы реакции в шарнире; tc — моменты реакции в шарнире; fc и tc — суммарные силы и моменты, действующие в шарнире.

Рисунок 1.12 — Блок шарнира *«Revolute Joint»* с активированными входами и выходами

При выводе результатов моделирования реакций в шарнире следует не забывать об определении направления этих реакций. Это направление задаётся в разделе *Composite Force/Torque Sensing* параметром *Direction (Направление)*. Так как силы взаимодействия в шарнире между базовым элементом и последующем определяются из третьего закона Ньютона, их значение по модулю будут равными, а направления противоположными и иметь знак в зависимости от направления воздействия. В параметре *Direction* необходимо определить это направление. При выборе значения *Follower on Base* будет определяться сила, с которой тело, присоединённое к следящему порту, воздействует на базовое тело. В случае выбора значения *Base on Follower* направления действия сил изменятся на противоположные. Таким образом, будут выведены значения сил воздействия тела, присоединённого к базовому порту, на тело соединённое со следящим портом.

Параметр *Resolution Frame* определяет в какой системе координат следует проводить измерения. При наличии вращения результаты измерений в системах координат *Base* и *Follower* будут отличаться. Это связано с тем, что следящий порт шарнира вращается относительного базового, а проекции силы взаимодействия на соответствующие оси зависит от направления силы и направления оси.

Наличие входов физических величин в шарнирах требуют особого внимания к определению единиц измерения. Эти единицы измерения задаются в свойствах портов выхода соответствующих блоков.

Блок *«Rigid Transform»* является одним из важнейших блоков, который позволяет связывать между собой в нужных местах отдельные детали механизма. Поэтому частота его применения достаточно высока.

Рассмотрим включение в модель маятника — стержня длиной 100 мм.

Стержень может быть определён как простейший примитив в блоке *«Solid»*. В этом случае центр его системы координат находится на пересечении его главных осей. Ось Z направлена вдоль оси стержни. Стержень должен

быть соединён одним краем с шарниром, а другим с грузом в виде шара. Очевидно его локальная система координат должна быть смещена относительно центра шарнира наполовину длины стержня.

Смещение детали относительно предыдущей, базовой, осуществляется несколькими методами:

Cartesian — 3-D перемещение в системе декартовых координат;

Standard Axis — определяет 1-D перемещение вдоль оси X, Y или Z;

Cylindrical — 3-D перемещение в системе цилиндрических координат.

В случае с примером маятника достаточным является перемещение вдоль одной оси (рис. 1.13).

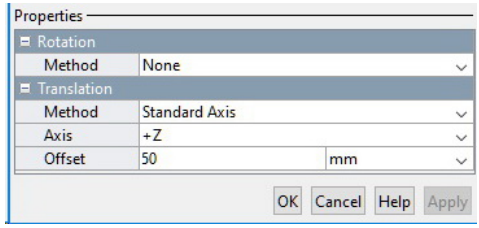


Рисунок 1.13 — Смещение стержня относительно шарнира

Присоединив стержень к сети и добавив после него такое же преобразование координат, получим точку смещения крепления шара на конце стержня. Остаётся создать с помощью блока *«Solid»* из примитива шар и подключить его к сети модели. Модель маятника готова (рис. 1.14). В модели, показан-

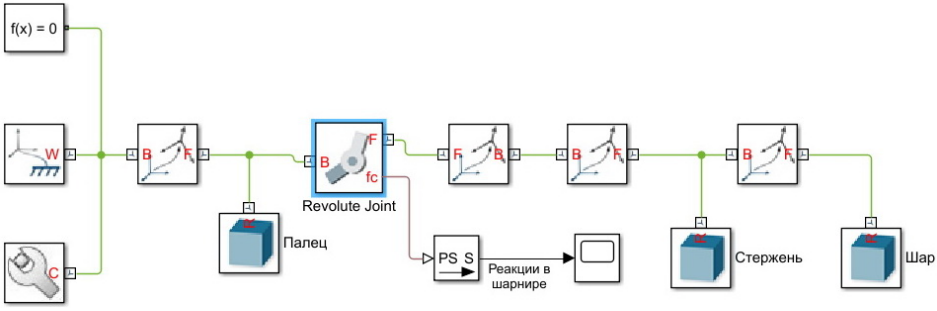


Рисунок 1.14 — Модель маятника

ной на данном рисунке присутствуют ещё два блока, которые будут описаны ниже.

Запустим имитацию модели маятника, оставив параметры по умолчанию. В открывшемся окне (рис. 1.15) можно видеть анимацию модели маятника. Инструменты Simscape Multibody позволяют замедлять скорость анимации, изменять масштаб и ракурс просмотра.

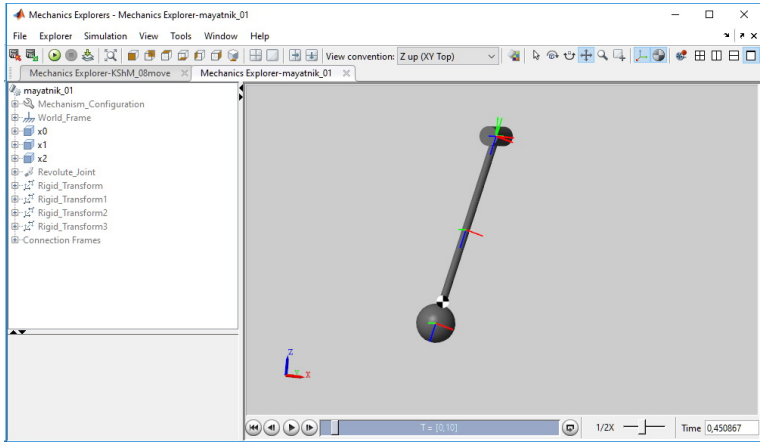


Рисунок 1.15 — Модель маятника

С одной стороны в результате моделирования мы видим прорисовку физической модели, что может дать любая система 3D моделирования, и ее анимацию. Особенность анимации в отличии от аналогичных роликов полученных с помощью, например, SolidWorks или Компас 3D заключается в том, что она не просто показывает возможные перемещения, а моделирует движение в зависимости от сил действующих на механизм и внутри него. В данном примере мы имеем маятник, на который действует сила тяжести вдоль оси Z. Кроме того в единственном присутствующем шарнире присутствует сила трения. Моделирование начинается из начального положения маятника отклонена во от оси Z на 45°. В результате получаем затухающие колебания маятника. Что чётко видно на анимации.

В случае Simscape Multibody анимация позволяет визуально наблюдать поведение механизма при изменении силового взаимодействия. Это является существенной помощью для инженера исследователя. Анимация позволяет увидеть нестандартные ситуации, в отдельных случаях определить необходимые начальные условия. Однако это лишь вспомогательный и в какой-то степени маркетинговый механизм. То есть Simscape Multibody позволяет создавать анимационное видео и в качестве презентационных роликов для проведённых разработок.

Как уже говорилось ранее, сенсорные выходы блоков шарниров позволяют определять реакции, действующие в шарнирах. Для их определения и визуализации к каждому выходу необходимо подключить осциллограф из базовых блоков Simulink. При этом необходимо помнить, что базовые блоки Simulink работают с числами, то есть с безразмерными величинами. На выходе датчика с реакцией в шарнире мы будем иметь величину с единицами измерения силы или момента. Очевидно, такой датчик не может быть подключён напрямую к осциллографу.

Воспользуемся блоком преобразования физических величин.

Блоки  *«PS-Simulink Converter»* и  *«Simulink-PS Converter»* предназначены для конвертирования физических величин Simscape в единицы Simulink. Эти блоки расположены в в библиотеке Simscape Utilities. Достаточно в параметрах только указать физическую величину, которую желательно получить на выходе или ту, которая подаётся на вход.

Установив параметры блока *«PS-Simulink Converter»* N (Ньютоны) достаточно присоединить его ко входу в осциллограф. В результате моделирования на осциллографе получаем графики изменения реакций в шарнире (рис. 1.16). На осциллографе отразилось 3 графика при подключении к одному выходу. Очевидно, что датчик шарнира выводит значения реакции по трём осям в декартовой системе координат в последовательности x, y, z. При необходимости отображение ненужных реакций можно отличить.

1.2.2 Моделирование работы механизма с использованием внешних 3D моделей

Пример моделирование работы маятника, приведённый ранее в пункте 1.2.1, является достаточно простым в том числе и для понимания. Использование внутренних примитивов для создания модели возможно и может служить различным целям в основном теоретического характера, однако это далеко не всегда возможно в практическом применении. Реальные детали механизмов достаточно сложны и рисовать их в Simscape Multibody нецелесообразно. Поэтому незаменимым свойствам данного пакета является возможность подключение в блоке *«Solid»* 3D моделей деталей из внешнего источника.

В качестве примера рассмотрим создание, а в дальнейшем и исследование работы, кривошипно-шатунного механизма двигателя внутреннего сгорания.

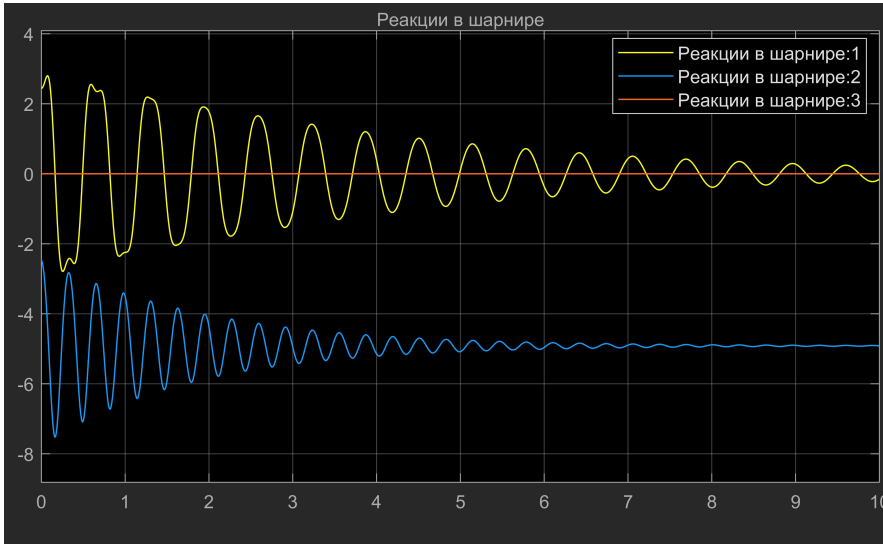


Рисунок 1.16 — Графики изменения реакций в шарнире при качании маятника

1.2.2.1 Создание 3D модели кривошипно-шатунного механизма в Simscape Multibody

Допустим мы имеем созданные в программах 3D моделирования модели поршня, пальца, шатуна и коленчатого вала (рис. 1.17). Приведённые модели созданы специально для данного примера и являются несколько упрощёнными, что не является принципиальным.

Блок Simscape Multibody *«Solid»* умеет считывать файлы в форматах STL и STEP. Поэтому предварительно подготовленные файлы с 3D моделями были импортированы в формат STEP. Указать путь к файлу с 3D моделью является несложной задачей даже на начальном этапе моделирования. Однако, в подключении к модели механизма есть «подводные камни», на которые необходимо обратить внимание.

Рассмотрим подробнее задание свойств для первой детали кривошипно-шатунного механизма: коленчатого вала (рис. 1.18). В разделе *Geometry* следует указать, что геометрия должна быть считана из файла, указать расширение файла модели и указать имя файла и полный путь к нему. Файл 3D модели может быть выбран с помощью диалогового окна. Для того, чтобы модель отразилась в окне просмотра следует нажать клавишу «F5». В

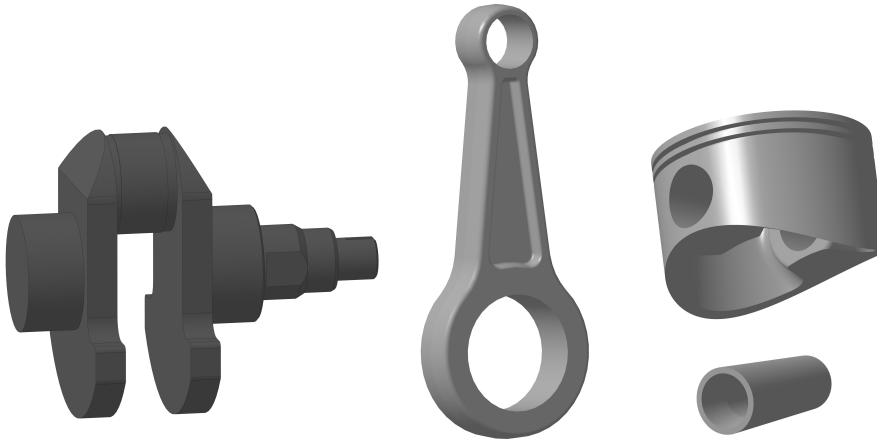


Рисунок 1.17 — 3D модели деталей кривошипно-шатунного механизма

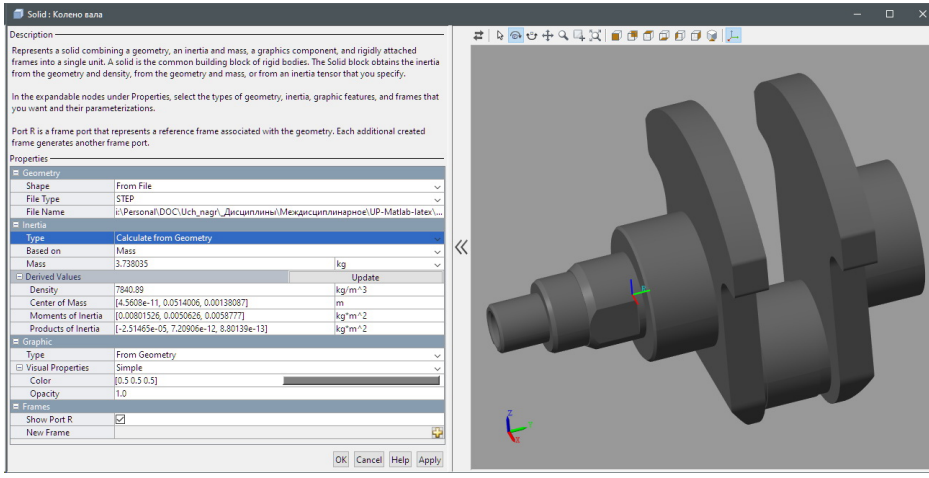


Рисунок 1.18 — Свойства коленчатого вала в модели кривошипно-шатунного механизма

дальнейшем необходимо помнить, что информации из файл 3D модели не помещается в модель Simscape Multibody, а читается непосредственно из файла во время компиляции модели. То есть при переносе модели выполненной в Simscape Multibody на другой персональный компьютер необходимо также

перенести и файлы с 3D моделями деталей механизма. Кроме того, в свойствах детали необходимо будет изменить путь к файлу 3D модели.

Блок *«Solid»* поддерживает и относительный путь к файлу. Этот путь можно записать в окне редактирования непосредственно вручную, но невозможно выбрать с помощью диалогового окна. Корректировка пути на относительный путь, например, ссылка на подкаталог, расположенной в каталоге проекта, позволяет переносить модель на другие ПК вместе со всеми файлами проекта и запускать её без редактирования.

Выберем вариант задание инерционных характеристик детали в разделе *Inertia* (рис. 1.18). Возможные варианты выбора: расчёт, исходя из геометрии, точечная масса и ручной вариант. Если мы хотим получить реальную картину при моделировании работы механизма, целесообразно выбирать расчёт, исходя из геометрии. Расчёт может основываться на массе детали и на плотности материала. Следует помнить, что из файла 3D модели передаются только геометрические характеристики детали. Поэтому есть необходимость заглянуть в редактор, в котором создавалась деталь, и, определившись с материалом, выбрать его плотность или массу, рассчитанную в этом программном продукте. В нашем случае указываем массу, которая составила 3,738 кг. Следует обращать внимание на единицы измерения, которые находятся справа от численных значений массы.

В подразделе *Derived Values* (Производные значения), после нажатия на поле обновления, появляются: плотность, моменты инерции и элементы тензора инерции. Они могут быть скопированы, но введение других значений ограничено.

Раздел *Graphic* управляет отображением модели при анимации. Не будем останавливаться подробно, но отметим, что в этом разделе можно настроить цвет и прозрачность детали, а также отключить отображение детали.

В общем-то на этом можно было бы остановиться. Последний раздел *Frames* управляет системой координат модели. Обратим внимание, что, в случае отсутствия преобразования системы координат, просмотр порта R должен быть включён. Этот раздел необходим для подключения 3D модели к системам координат модели Simscape Multibody. Преобразование координат может производиться и вне модели. Настройка преобразования системы координат была рассмотрена выше в пункте 1.2.1.

Раздел *Frames* позволяет упростить в целом ряде случаев преобразование системы координат указывая точки входа и выхода (места присоединения деталей) непосредственно на 3D объекте. Наиболее целесообразно применение этих преобразований систем координат в случае, если точка соединение деталей связана с какими-либо их геометрическими характеристиками. Учтём

эту возможность при создании модели двигателя с кривошипно-шатунным механизмом.

Определим положение коленчатого вала в пространстве и местоположение его соединений с другими деталями. Коленчатый вал обычно имеет подшипники скольжения расположенные на коренных шейках. Эти подшипники совместно с упорным оставляют только одну степень свободы для коленчатого вала. Таким образом, коленчатый вал должен быть соединён с блоком цилиндров двигателя с помощью шарнира, позволяющего осуществлять вращение. Пусть неподвижная система координат модели будет также связана с блоком цилиндров двигателя. Следовательно шарнир может быть прикреплён непосредственно к глобальной системе координат модели. С помощью блока преобразования координат необходимо согласовать системы координат точек соединения коленчатого вала и шарнира. Это связано с тем, что вращение может осуществляться только относительно оси Z, а в 3D модели коленвала вдоль него направлена ось Y. Второй шарнир представляет собой аналогичный подшипник вращения, расположенный на шатунной шейке. Таким образом, мы можем использовать тот же шарнир, однако он должен быть расположен на шатунной шейке. То есть необходимо повторное преобразование системы координат связанное с перемещением точки привязки к центру шатунной шейки. В результате получаем часть модели обозначенную зоной **A** на рисунке 1.19.

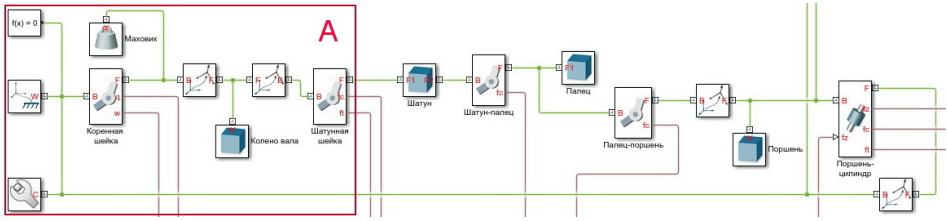


Рисунок 1.19 — Определение положения коленчатого вала и его соединений в модели кривошипно-шатунного механизма

Непосредственно после шарнира в модель может быть добавлена точечная масса, как показано на рисунке. Момент инерции этой массы относительно оси Z будет представлять собой момент инерции маховика. Изменяя его в процессе моделирования можно предварительно получить необходимые значения момента инерции для будущего маховика в зависимости от необходимых характеристик и оценить поведение кривошипно-шатунного механизма и двигателя.

Для определения точек соединения шатуна следует отключить порт по умолчанию и создать два порта в необходимых местах. При этом необходимо изменить направление осей, как показано на рисунке 1.20. Выберем в

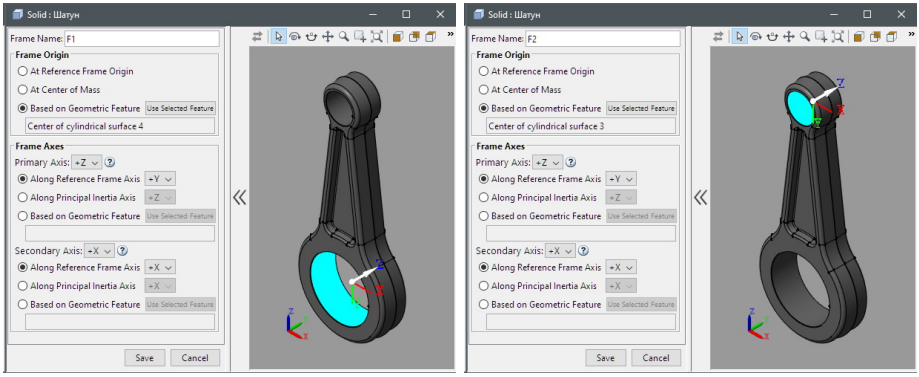


Рисунок 1.20 — Определение положения шатуна с помощью портов с новыми системами координат

разделе *Frame Origin* пункт *Based on Geometric Feature* (базироваться на особенностях геометрии). Это позволит определить местоположение локальной системы координат в зависимости от геометрических особенностей детали. После чего следует указать геометрический объект, который нам необходим. Так как подшипники расположены в отверстиях шатуна, соответственно эти цилиндрические отверстия и указаны в качестве базы для новой локальная система координат.

Как видно из рисунка 1.19, шатун соединён с шарнирами именно этими портами.

Введение в модель пальца является ещё более простой задачи. По условиям закрепления палец может быть плавающий и иметь возможность качания относительно поршня. То есть мы имеем шарнир. Этот шарнир имеет ту же ось вращения, что и шарнир соединяющий палец и шатун. В таком варианте палец может иметь только один порт соединения с моделью. И он может быть настроен непосредственно в окне параметров тела (рис. 1.21). Учитывая особенности 3D модели, расположение ее системы координат, достаточно только повернуть эту систему координат, оставив в той же точке.

В случае если палец в поршне закреплён неподвижно, шарнир между пальцем и поршнем может отсутствовать. Однако наличие шарнира позволяет снимать значения сил, действующих в направлениях с отсутствием степеней свободы. В том числе значения крутящих моментов. Поэтому для

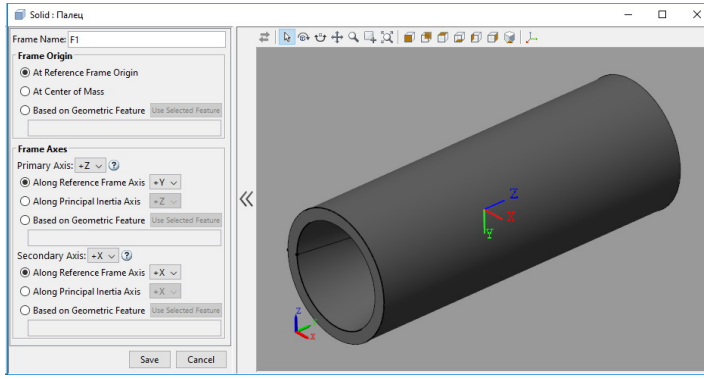


Рисунок 1.21 — Настройка порта с новой системой координат для подсоединения модели пальца

получения взаимодействия между поршневым пальцем возможно применение шарнира *«Weld Joint»* . Данный шарнир имеет только датчики сил взаимодействия деталей.

Вернёмся к модели, описываемой в примере.

Присоединение поршня более целесообразно с помощью блока преобразования координат *«Rigid Transform»*. Это связано с тем, что в поршне нет поверхности, геометрию которой можно было бы использовать для определения положение точки присоединения к пальцу. Поэтому установим положение поршня (рис. 1.19), задавая поворот осей координат и смещение. Это достаточно легко при получении некоторого навыка работы с преобразованием координат.

Последняя связь, которая будет замыкать цепь модели, это связь между поршнем и блоком цилиндров. Так как поршень движется в цилиндре, он имеет две степени свободы: движение вдоль оси цилиндра и вращение относительно этой же оси. В библиотеке шарниров присутствует *«Cylindrical Joint»* . Цилиндрический шарнир допускает перемещение и вращение относительно оси Z. Так как ось Z поршня направлена вдоль направления его движения, то преобразование координат в данном случае является лишним и шарнир может быть подключён непосредственно к точке присоединения поршня. А этой точкой в данном случае является центр пальца.

Остаётся только соединить следящий порт цилиндрического шарнира с корпусом двигателя, то есть с базовой глобальной системой координат. Так как система координат точки соединения цилиндрического шарнира

существенно смещена от центра глобальной системы координат и переориентирована по осям, необходимо добавить блок с обратными суммарными преобразованиями систем координат.

В противном случае появится неопределенность. Поскольку цепь замкнута определение точки присоединения поршня может осуществляться при движении по цепи в разных направлениях от блока *«World Frame»*. В результате вычислений должны получаться одни и те же координаты. В противном случае модель не пройдёт проверку и уравнения не смогут быть составлены.

Модель, показанная на рисунке 1.19, завершена. Можно запускать выполнение моделирования. Рекомендуется изначально установить желаемый максимальный шаг моделирования. Иногда это позволяет избежать грубых ошибок и практически во всех случаях, при моделировании в Simscape Multibody, позволяет получать более плавную анимацию механизма.

Для получения анимации необходимо наличие сил и начальных условий определяющих положение и движение начальный момент времени. Как правило, для получения движение достаточно в блоке Mechanism Configuration определить направление силы тяжести. Например, оставив его по умолчанию.

Если 3D модель, в открывшемся окне (рис. 1.22), соединена так как и предполагалось и начато движение, значит задачу удалось выполнить. Модель есть и она работает.

В случае если желаемый результат не достигнут, подсвечивание систем координат в точках соединения деталей, подсвечивание мест соединения при выделении детали, позволяет упростить поиск и устранение ошибок.

1.2.2.2 Определение условий работы механизма

Получение анимации при моделировании в Simscape Multibody не является самоцелью. Основная задача моделирования получить численные значения кинематических и динамических характеристик работы механизма в необходимых инженеру и исследователю точках. Хотя анимация в первую очередь и выполняет контрольную функцию, она позволяет в отдельных случаях визуально увидеть нежелательные явления в работе механизма.

Для визуализация работы и получения искомых данных моделирования необходимо определить и задать условия работы механизма. К ним можно отнести начальные условия движения и внешние силы, действующие на детали механизма.

Так как в качестве примера принят кривошипно-шатунный механизм двигатель внутреннего сгорания со стороны цилиндра на поршень этого механизма действует сила давления газов. Именно это сила создаёт крутящий момент двигателя. Для того, чтобы частота вращения коленчатого вала была

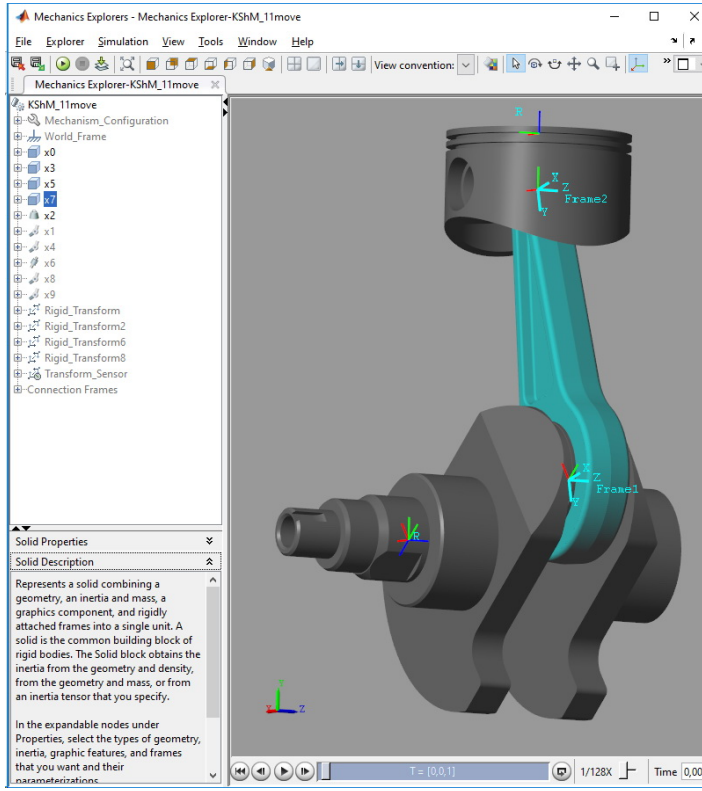


Рисунок 1.22 — Окно анимации 3D модели кривошипно-шатунного механизма

относительно постоянной или имела незначительные изменения во времени, к коленчатому валу должен быть приложен момент сопротивления.

Введем силу давления газов в модель двигателя. Так как давление газов определяется в результате расчёта рабочего процесса двигателя по достаточно сложным зависимостям, создание такой модели рабочего процесса для нашего примера нецелесообразно. Однако значение давления газов может быть получено по результатам предварительного моделирование рабочего процесса. В этом случае хорошее прикладное значение могут иметь например аппроксимирующие нейронные сети.

При проведении научных исследований авторами проводилось моделирование рабочего процесса бензинового двигателя в среде Simulink. Среди

данных, полученных в результате имитационного моделирования, были получены давления в цилиндре двигателя при различных углах поворота коленчатого вала в ряде рабочих циклов. Эти данные были использованы для обучения нейронной сети. Подробно о создании и обучении аппроксимирующих нейронных сетей рассказано в подразделе 2.1. Здесь только заметим, что в результате обучения была создана сеть, которая в зависимости от угла поворота коленчатого вала выдаёт абсолютное давление газов в цилиндре двигателя на режиме работы близком к номинальному. Сеть была сохранена в виде Simulink диаграммы.

Для получения силы давления газов абсолютное давление в цилиндре приводится к манометрическому и умножается на площадь поршня. Эти преобразования реализованы средствами Simulink и размещены в отдельной области, показанной на рис. 1.23. На этом же рисунке можно увидеть результат, выдаваемый нейронной сетью.

Следует обратить внимание: нейронная сеть на входе получает безразмерные числовые значения. То есть работает в формате Simulink. Модуль Simscape является модулем физического моделирования и имеет в выходных величинах размерность. Поэтому полученные данные об угле поворота коленчатого вала должны быть предварительно конвертированы в радианы. Подобную конвертацию необходимо выполнить и перед приложением силы давления газов к поршню в цилиндре двигателя. Впрочем соединение портов Simulink и Simscape без предварительной конвертации невозможно.

Текущий угол поворота коленчатого вала можно получить с помощью датчиков шарнира поворота имитирующего коренную шейку коленчатого вала. Там же снимается крутящий момент с двигателя и определяется начальное положение механизма. Поэтому в свойствах шарнира (рис. 1.24) определим следующие значения:

— начальное положение коленчатого вала задано в одном из положений в верхней мёртвой точке;

— начальная угловая скорость коленчатого вала принята равной $500\,\mathrm{c}^{-1}$;

— коэффициент сопротивление подобран экспериментально для получения равномерного движения;

— включены датчики кинематических параметров коленчатого вала, то есть датчики положения и угловой скорости вращения в шарнире.

Данные полученные с помощью датчика положения, после конвертации в радианы, передаётся на вход нейронной сети. Результаты выдаваемые нейронной сетью должны поступить на порт «fz» (рис. 1.19). Это порт силы, действующей в шарнире вдоль оси Z.

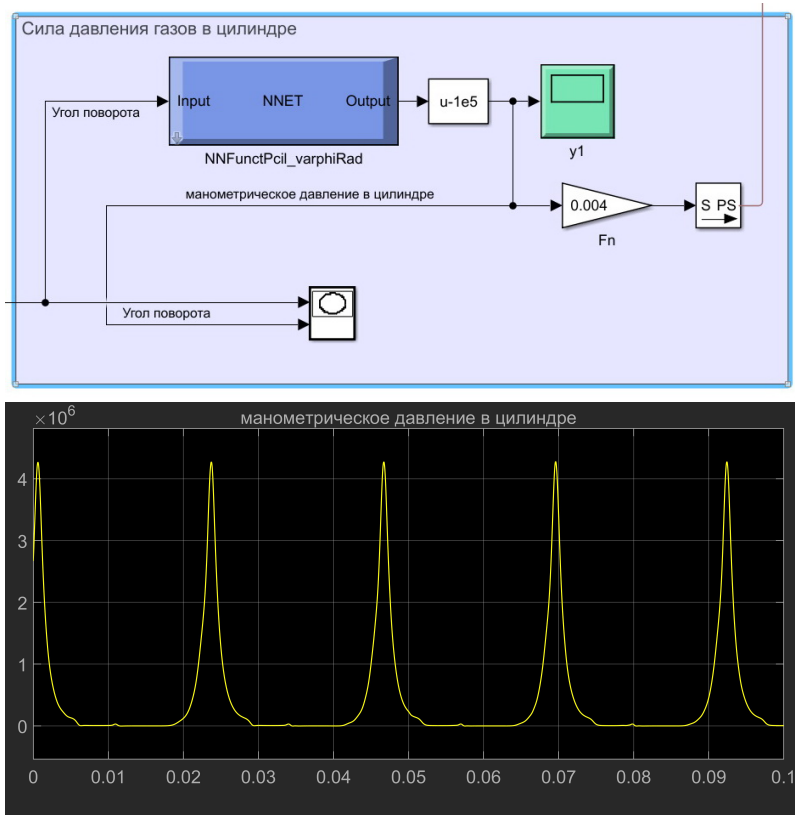


Рисунок 1.23 — Определение силы давления газа в модели

1.2.2.3 Получение результатов моделирования

Ранее мы уже выводили результаты моделирования в Simscape Multibody на экран осциллографа (рис. 1.16). Поэтому основные и необходимые для просмотра результатов моделирования действия нам известны. В зависимости от того, как предполагается обрабатывать в дальнейшем результаты моделирования, могут быть использованы различные инструменты Matlab Simulink.

Помня о том, что движение относительно, можно разделить получаемые результаты на группы: силы и кинематические параметры в шарнирах, относительные кинематические параметры между различными точками механизма.

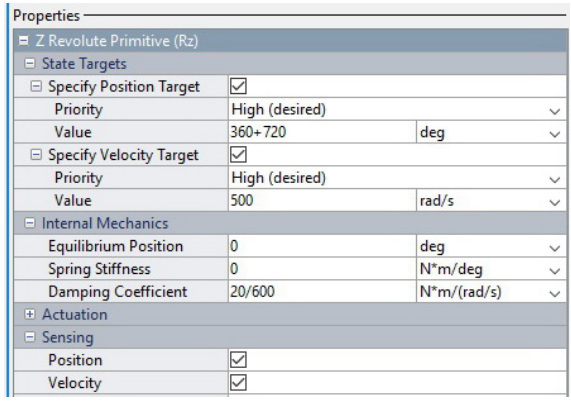


Рисунок 1.24 — Начальные условия и условия работы шарнира коренной шейки

Для получения сил и кинематических параметров в шарнирах необходимо:

— включить датчики кинематики в разделе соответствующего примитива шарнира;

— включить необходимые датчики сил и моментов, действующих в шарнире;

— определить параметры направления действия сил и моментов;

— выходные порты шарниров соединить с блоками конвертации физических единиц в числа Simulink;

— в зависимости от предполагаемых дальнейших действий направить сигнал в осциллограф, рабочее пространство или записать в файл.

Для оценки результатов моделирования, выведем на осциллографы те силы и реакции, которые обычно рассчитываются в теории двигателя при анализе динамики. Пример подсоединения осциллографов показан на рис. 1.25.

В теории двигателей внутреннего сгорания расчёт обычно проводят при постоянной частоте вращения коленчатого вала [4; 8; 9; 13] и для одного цикла. В случае имитационного моделирования можно говорить только о получении усреднённой постоянной частоты вращения коленчатого вала. Как видно из данных угловой скорости в шарнире коренной шейки (рис. 1.26, а) при отсутствии маховика частота вращения коленчатого вала постоянной не является.

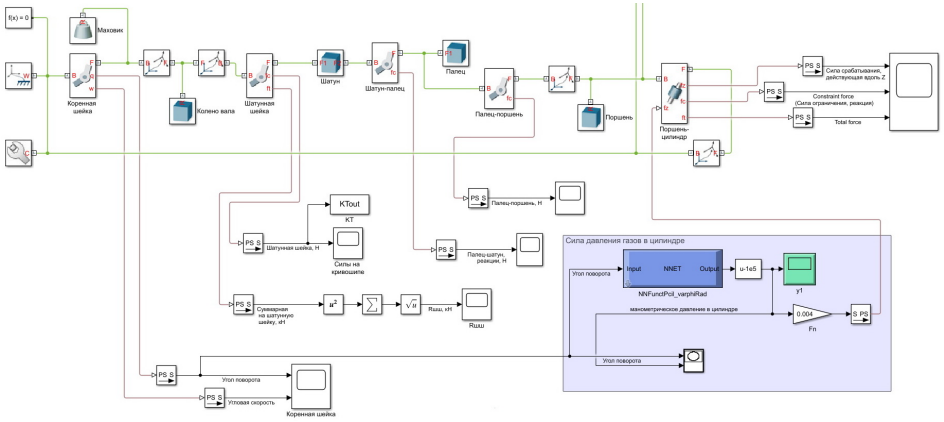
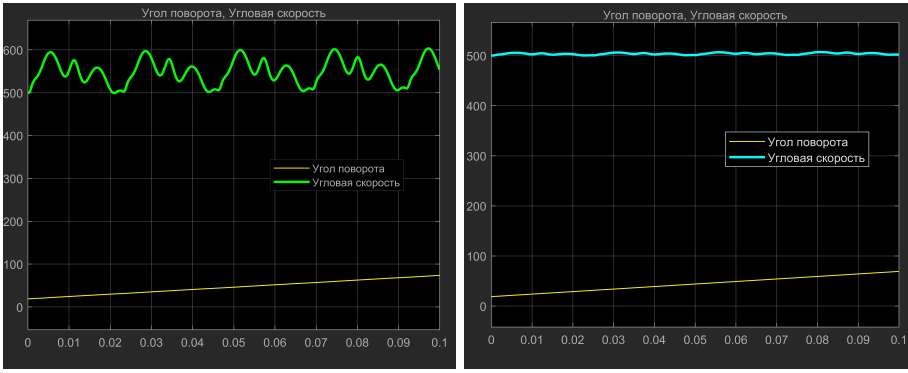


Рисунок 1.25 — Подключение осциллографов к портам вывода датчиков в шарнирах кривошипно-шатунного механизма



а) при отсутствии маховика, б) при сглаживании за счёт увеличения инерционной массы маховика

Рисунок 1.26 — Изменение угловой скорости коленчатого вала

Подбор момента инерции маховика позволяет существенно сгладить пульсации частоты вращения и получить практически постоянную частоту вращения коленвала (рис. 1.26, б).

Как уже было показано ранее, результаты выведенная в осциллограф могут быть сохранены в качестве фигуры (рисунка) Matlab. Ещё раз отметим, что рисунок Matlab хранит в своем формате все данные для построения

графика. Это позволяет в дальнейшем редактировать рисунки, при необходимости масштабировать и выводить в удобном для анализа виде.

Приведём некоторые графики сил, предварительно увеличив их масштаб по оси времени до размеров одного цикла.

Силы, действующие между пальцем и шатуном, показанные на рис. 1.27, свидетельствуют о получении результатов совпадающих с классическими. В этом легко можно убедиться, сравнив формы графиков с аналогичными, приведёнными в специальной литературе [4; 8; 9; 13].

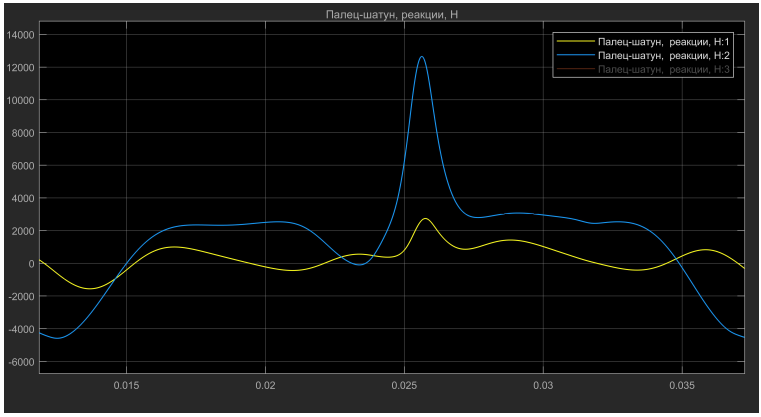


Рисунок 1.27 — Силы, действующие на шатун

Тоже можно сказать и о силах, действующих на шатунную шейку (рис. 1.28). Полученные на осциллографе графики силы T совпадают с классическим представлением, а другая сила несколько смещена относительно оси абсцисс. Что абсолютно верно, так как это, полученная в результате имитационного моделирования, реакция уже учитывает все силы инерции действующие в шарнире. То есть она несколько отличается от силы K.

Полученные с датчиков силы после конвертации могут быть обработаны средствами Simulink. Так геометрическая сумма сил (рис. 1.29), действующих на шатунную шейку, даёт суммарную реакцию, график которой можно увидеть на рисунке 1.30.

В случае если необходимо получить фазовые диаграммы или, например, полярную диаграмму сил, действующих на шатунную шейку, результаты расчёта могут быть сохранены в рабочее пространство Matlab (рис. 1.31), а полученные переменные использованы для построения необходимых графиков (рис. 1.32).

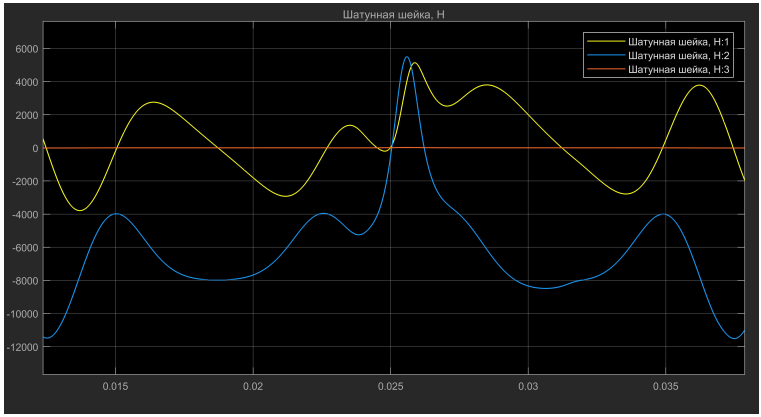


Рисунок 1.28 — Силы, действующие на шатунную шейку

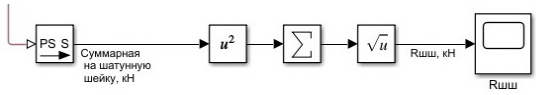


Рисунок 1.29 — Расчет суммарной силы, действующей на шатунную шейку

Приведённые примеры позволяют убедиться, что моделирование с помощью Simscape Multibody позволяет получать корректные результаты в различной удобной или привычной форме. Иногда привидение моделирования к условиям классических расчётов позволяет проверить модель, убедиться в правильности ее составления. И лишь после этого имеет смысл исследовать режимы, которые невозможно получить в результате обычных расчётов. И в этом случае имитационное моделирование имеет полное преимущество перед классическими техническими и научными расчётами.

В описанном здесь примере для получения постоянной частоты вращения использовался маховик с достаточно большим моментом инерции. Подбор маховика является одной из инженерных задач, которая решается при проектировании двигателей. Зачастую на двигатели спортивных автомобилей устанавливают маховики с меньшей инерционной массой. В модели достаточно легко отключить маховик и посмотреть, какие силы будут действовать при неравномерной частоте вращения (рис. 1.32, а).

Как видим, диаграмма несколько поменяла форму, но кроме этого, возросло и максимальное значение силы, действующей на шатунную шейку. Это надо учитывать при проектировании и проведении прочностных расчётов.

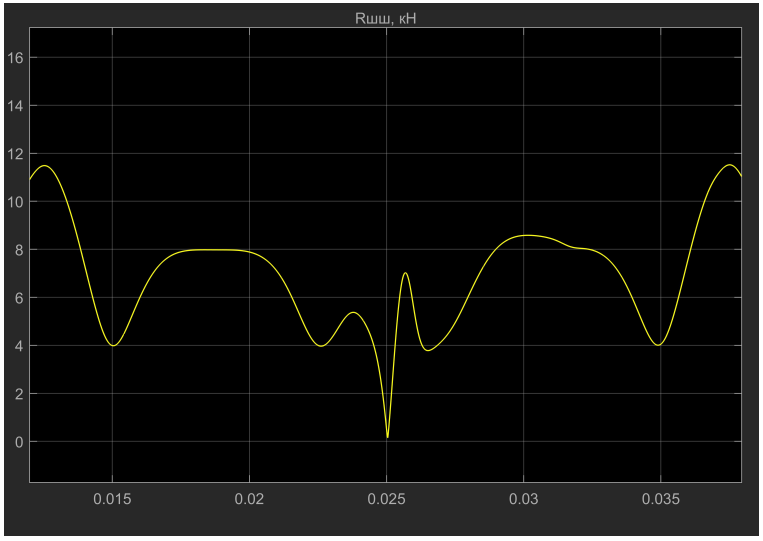


Рисунок 1.30 — График суммарной силы, действующей на шатунную шейку

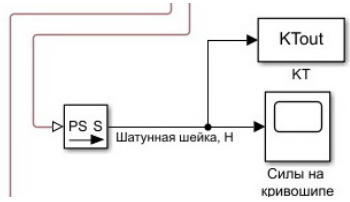


Рисунок 1.31 — Сохранение данных для построения полярной диаграммы

То есть данные, полученные в результате моделирования, могут быть исходными для принятия решений. И имитационное моделирование механизмов с помощью Simscape Multibody позволяет получить неизвестные ранее данные или такие, получение которых обычным способом сильно осложнено.

Очевидно, что силовое взаимодействие происходит в месте соединение деталей, то есть шарнирах, поэтому именно в шарнирах и расположены датчики силового взаимодействия и кинематики шарниров. Однако эти датчики не позволяют определить кинематику произвольной точки механизма.

Для этого существует блок *«Transform Sensor»*. Этот датчик позволяет выводить относительные кинематические параметры между двумя точками механизма. Рассмотрим использование такого датчика на примере

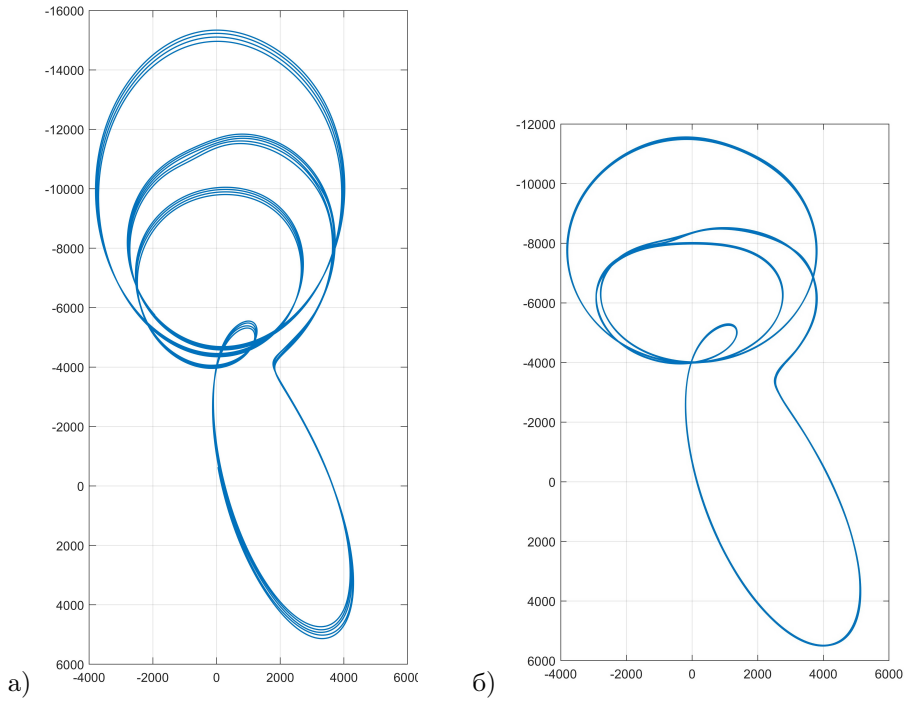


Рисунок 1.32 — Полярные диаграммы, сил действующих на шатунную шейку: а) без маховика; б) с маховиком

определения параметров движения поршня. Соединим один порт с точкой в сети модели, для которой необходимо определить параметры, а второй порт с глобальной системой координат (рис. 1.33). В настройках датчика включим порты для тех кинематических величин, которые необходимо получить. В данном случае это линейные перемещения, скорости и ускорения (рис. 1.34). В свойствах блока датчиков следует выбрать систему координат, в которой желательно получить результат. Порты выхода после преобразования физических величин подключаются к соответствующим осциллографам.

Датчики покажут кинематические параметры точки, к которой присоединён порт выхода, относительно точки, соединённой с базовым портом. При показанном выше соединении будут получены значения перемещения, скорости и ускорения поршня относительно цилиндра (рис. 1.35). Очевидно, что при отсчёте от верхней мёртвой точки значения перемещения будут отрицатель-

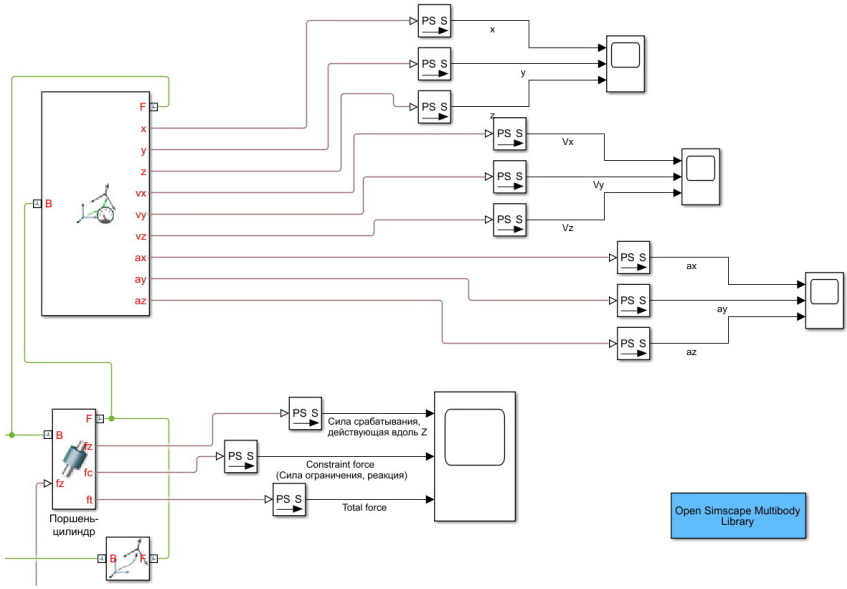


Рисунок 1.33 — Подключение *«Transform Sensor»* для определения кинематических параметров движения поршня в глобальной системе координат

ными. Поскольку в глобальной системе координат с учётом трансформаций ось Y направлена вверх. А так как цилиндрический шарнир ограничивает перемещение вдоль других осей, кинематические параметры по другим осям равны 0.

Свойства блока *«Transform Sensor»* являются в определённом смысле уникальными. Совместно с возможностями блоков твёрдых тел по установлению портов вывода, датчик кинематики даёт возможность определить параметры в любой точке тела. Например, если создать дополнительный порт на шатуне, связанный с его центром тяжести, появляется возможность построить траекторию движения центра тяжести шатуна. На рисунке 1.36 (а), показано подключение датчика к центру тяжести шатуна и получение его координат в полярные в системе. Траектория движения центра тяжести относительно оси вращения коленчатого вала, показана на рисунке 1.36 (б).

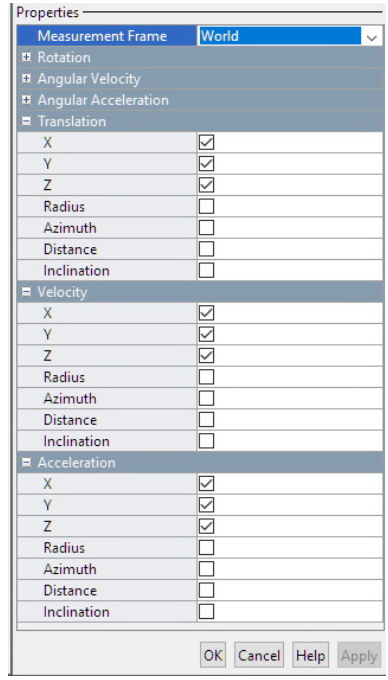


Рисунок 1.34 — Включение портов выхода датчика *«Transform Sensor»*

1.2.3 Другие возможности Simscape Multibody

Говоря о моделирование с помощью Simscape Multibody, мы попытались на примерах показать возможности пакета и познакомить с особенностями применения тех или иных блоков. Конечно охватить все возможности в рамках данного пособия невозможно. Кроме того, такая цель и не ставилась. Довольно сложно найти пример, который бы использовал большую часть возможностей. Кроме того, он бы имел значительную сложность и не дал бы возможность как объяснить, так и понять процесс создания и настройки модели. Конечно в очередной раз здесь следует отправить читателя к справочной системе Matlab. Но, очевидно, с пониманием базовых моментов моделирования, освоение применения других блоков не должно вызывать существенных проблем.

Тем не менее попытаемся кратко указать на возможности данного пакета.

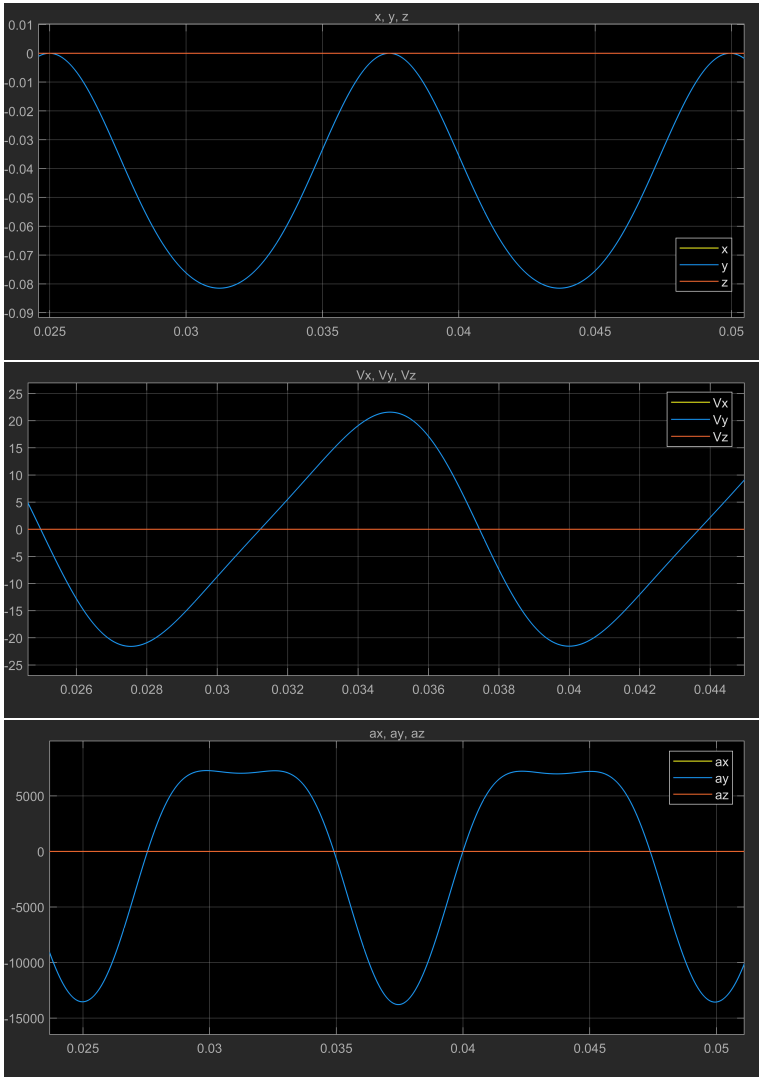
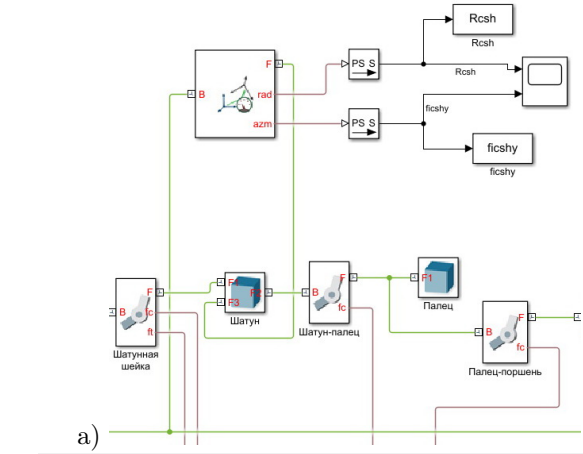
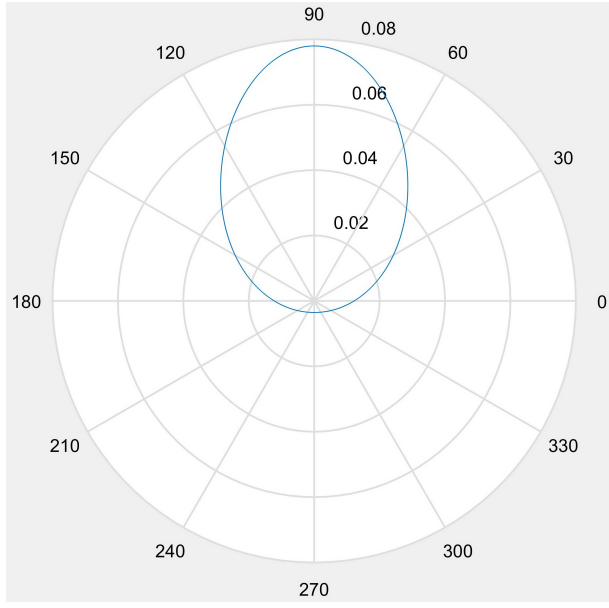


Рисунок 1.35 — Кинематика поршня относительно цилиндра

В целом библиотека Simscape Multibody включает в себя (рис. 1.37): библиотеку тел, библиотеку пространственных линий и поверхностей, библиотеку преобразования система координат, шарниров, ограничений, элементов



а)



б)

Рисунок 1.36 — Траектория центра тяжести шатуна в полярной системе координат

подъемно-транспортного оборудования таких как блоков и кабелей, библиотеку зубчатых зацеплений и библиотеку приложения сил и моментов.

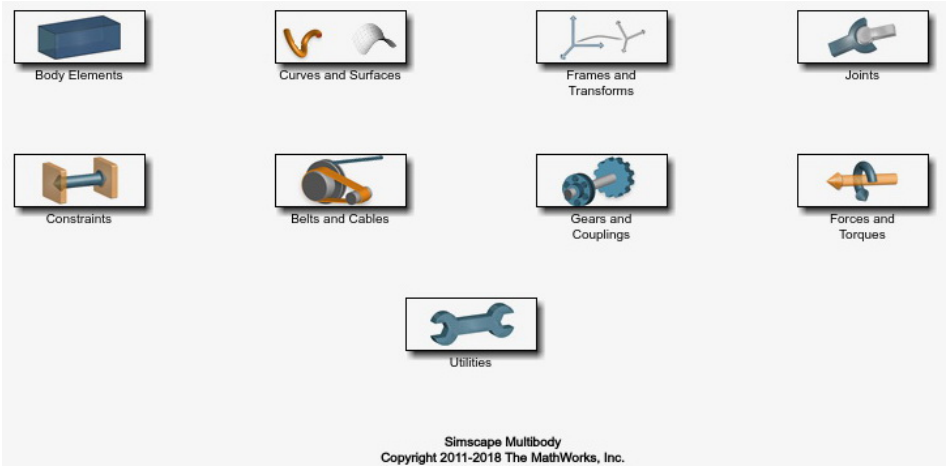


Рисунок 1.37 — Библиотеки Simscape Multibody

Некоторые из библиотек Simscape Multibody нам уже знакомы. Так, например, в инструментах находится блок конфигурации модели, и достаточно подробно мы говорили о шарнирах. Имеем опыт применения некоторых блоков из библиотеки тел. С системами координат и их преобразованиями, также знакомы. Но прежде чем перейти к тем частям библиотеки, которые в предыдущих примерах нам не понадобились, дополним информацию об уже знакомых библиотеках.

В библиотеке систем координат и преобразований имеется блок *«Reference Frame»*. Блок определяет в сети модели локальную неинерциальную систему координат. Этот блок может быть интересен если необходимо, например, производить измерения относительно какой-либо точки механизма. Положение глобальной системы координат, ее центра и ориентации, может быть неудобно для наглядного представления результаты моделирования. В этом случае локальная неинерциальная система координат может оказаться незаменимой.

Следует обратить внимание на возможные варианты задания тел, входящих механизм. На рисунке 1.38 показаны блоки, входящие библиотеку элементов тела.

Знание возможностей и имеющихся блоков позволяет более эффективно решать поставленные перед инженером и исследователем задачи. Так, например, блок «Graphic» может помочь визуализировать движение какой-либо точки, установив маркер выбранной формы в нужном месте.

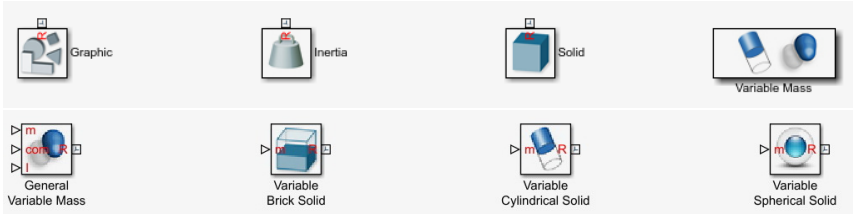


Рисунок 1.38 — Блоки, входящие библиотеку элементов тела Simscape Multibody

Четыре блока библиотеки дают возможность создавать тела, инерционные характеристики которых меняются во времени. Это могут быть цилиндр, сфера, параллелепипед произвольный эллипсоид инерции. Все параметры данных тел могут задаваться и меняться во времени. Это позволяет исследовать поведение очень сложных объектов во время их взаимодействия или в процессе израсходования части массы тела.

В Simscape Multibody, работая с различного рода шарнирами, указать ограничения в перемещение относительно какого-либо примитива до определённых версий было невозможно. Поэтому в некоторых случаях можно наблюдать, как тела деталей экспортированных из 3D моделей пересекаются. Для создания подобного и другого рода ограничений имеется библиотека ограничений *«Constraints Library»* (рис. 1.39).

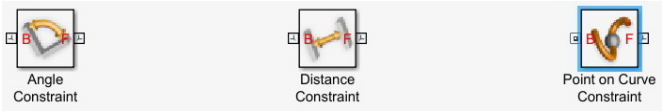


Рисунок 1.39 — Блоки, входящие библиотеку ограничений Simscape Multibody

Блоки библиотеки позволяют делать как простые ограничения, по углу поворота или по перемещению, так и сложные по пространственным кривым  . Это позволяет проектировать и исследовать динамику таких механизмов как, например, кулачковый (рис. 1.40) или с перемещением детали по направляющему пазу.

Траектория перемещения точки может быть задана с помощью блока  *«Spline»*. Этот блок позволяет задавать двух- и трёхмерные кривые, сглаженные сплайном.

Рисунок 1.40 — Пример применения ограничения движение точки по кривой

В какой-то степени к блокам ограничения можно отнести зубчатые зацепления (рис. 1.41).

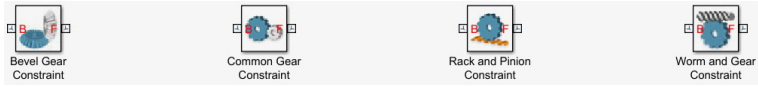


Рисунок 1.41 — Блоки библиотеки зубчатых зацеплений

Эти блоки ограничивают взаимное перемещение деталей в соответствии с параметрами зацепления. Наилучшего эффекта можно добиться при совместном применении 3D моделей деталей зубчатого зацепления, спроектированного с использованием систем геометрического 3D моделирования и модели в Simscape Multibody.

Модель простейшей зубчатой передачи может выглядеть так, как показано на рисунке 1.42. Шестерня и колесо установлены на корпусе и вращаются под действием приводного момента и момента сопротивления.

Точность совмещения зубьев шестерни и колеса для визуализации может быть обеспечена за счёт правильной ориентации систем координат или подбором начального положения в шарнирах.

В приведённой модели, в отличие от предыдущих, приводной момент и момент сопротивления приложены не в шарнирах, а непосредственно к телу детали с помощью блока  *«External Force and Torque»*. В случае с зубчатым зацеплением место приложения момента совпадает с портом выхода на шарнир. Поэтому блок внешних сил и моментов подсоединён в сеть в точке соединения зубчатого колеса и корпуса.

Библиотека *«Forces and Torques Library»* (рис. 1.43) позволяет дополнительно вводить в модель силу гравитации, отличную от глобальной, внутренние силы и моменты, упругие и демпфирующие элементы, а также

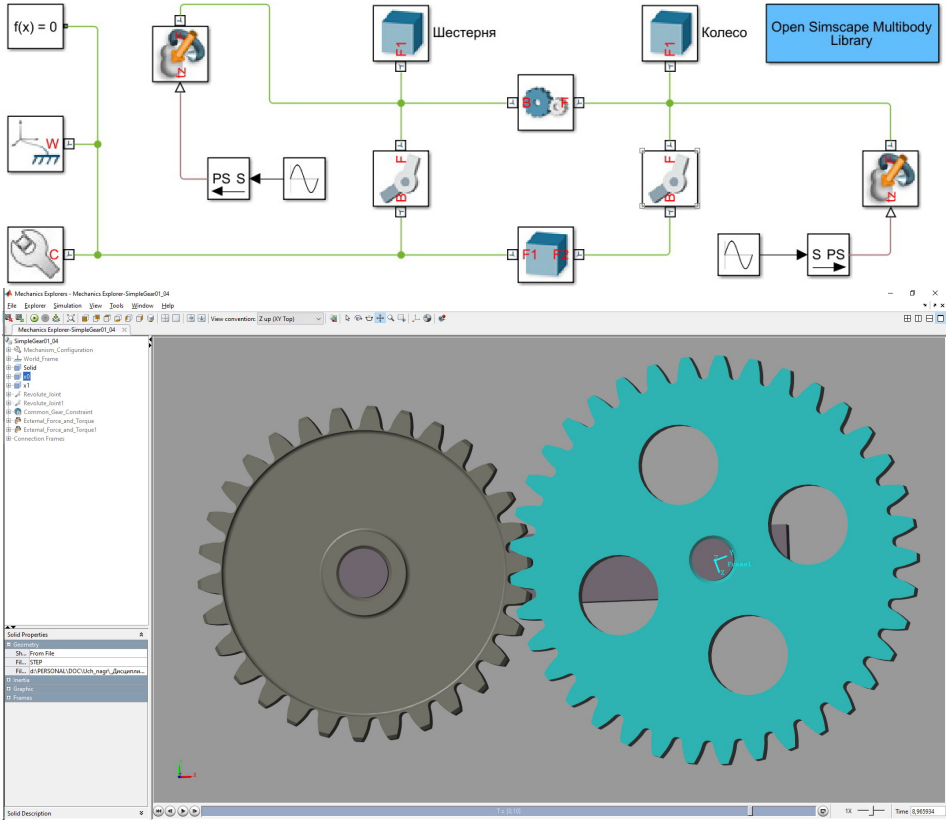


Рисунок 1.42 — Модель цилиндрического зубчатого зацепления

внешние усилия, которые могут быть приложены не только в шарнирах, а в любом месте механизма.

Блок *«Inverse Square Law Force»* моделирует силовое взаимодействие, зависящие от квадрата расстояния. То есть позволяет моделировать силу всемирного тяготения или взаимодействие электрических зарядов.

В заключении, говоря о модуле твердотельного моделирования Simscape Multibody, следует отметить, что он является логическим продолжением базовых библиотек Simscape, но при этом представляет собой систему моделирования более высокого уровня. Он существенно снижает затраты времени инженера и исследователя на подготовку и анализ модели. Если базовые фундаментальные библиотеки (Simscape Foundation library) позволяют в мак-

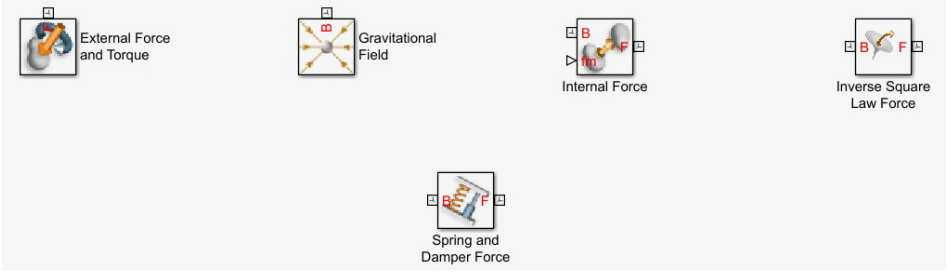


Рисунок 1.43 — Библиотека сил и крутящих моментов Simscape Multibody

симально детализировать модель, то Simscape Multibody позволяет получить тот же результат за счёт настроек параметров блоков, при этом использование анимации с возможностью использовать геометрию 3D объектов, подготовленных в специальных графических редакторах САПР, даёт возможность быстрее и проще выявить ошибки в разработке модели.

Данный продукт является весьма перспективным ещё и потому, что позволяет создавать и исследовать сложные модели механизмов, создавать корпоративную базу моделей, повысить скорость разработки и интеграции в более крупные модели для исследования конечного продукта. Например Simscape Multibody позволяет разработать модели отдельных агрегатов автомобиля, а в дальнейшем состыковать их в общую модель автомобиля и исследовать его поведение в определённых условиях. В отличии от Simscape Drivline или Vehicle Dynamics Blockset, как и других подобных модулей более высокого уровня, Simscape Multibody имеет оптимальное соотношение детализации и укрупнения при моделировании. Это позволяет получить реальную модель, у которой учтены все конструктивные особенности механизма. В более высокоуровневых моделях есть необходимость вводить ряд обобщённых параметров. В отдельных случаях эти параметры не известны, в других же их можно задать только приблизительно если отсутствуют более детальные модели. В любом случае высокоуровневые модели ведут к определённым упрощениям.

В конечном итоге выбор модулей для моделирования и их соотношение в конечной модели зависит от поставленных целей и задач.

ПРИМЕНЕНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ МОДЕЛИРОВАНИЯ ПРОЦЕССОВ

Широкое развитие компьютерной техники позволило в последние годы прийти к практическому применению нейронных сетей, как одного из математических методов, основанных на статистической природе нейронных сетей. Этому способствовало решение задач обучения, нейронных сетей, реализованное в ряде прикладных программных пакетов. Достаточно широкие возможности при относительной простоте задач по работе с нейронными сетями даёт Neural Network Toolbox из пакета Matlab.

Рассмотрим некоторые возможные варианты применения нейронных сетей для замены классических статистических методов и применения в математическом моделировании процессов, например в технических системах.

Применение нейронных сетей в настоящее время в первую очередь относится к задачам:

- классификации (распознавание образов, звуков и так далее);
- аппроксимации;
- прогнозирования временных рядов.

2.1 Использование нейронной сети для аппроксимации

Matlab имеет хорошие возможности для быстрого поиска и подбора аппроксимирующих зависимостей. Однако эти возможности далеко не всегда удовлетворяют потребностям исследователя. Зачастую подбор функции является нерешаемой задачей или же функция является очень сложной, иногда кусочно-заданной, и работать с ней в дальнейшем затруднительно. В таких случаях может быть целесообразно использование нейронной сети.

Рассмотрим использование нейронной сети на примере аппроксимации различных функций.

2.1.1 Применение нейронной сети для аппроксимации двумерных данных

Пусть имеется некоторое количество данных, фактически описываемое экспоненциальным законом и полученное с большим количеством шума (рис. 2.1).

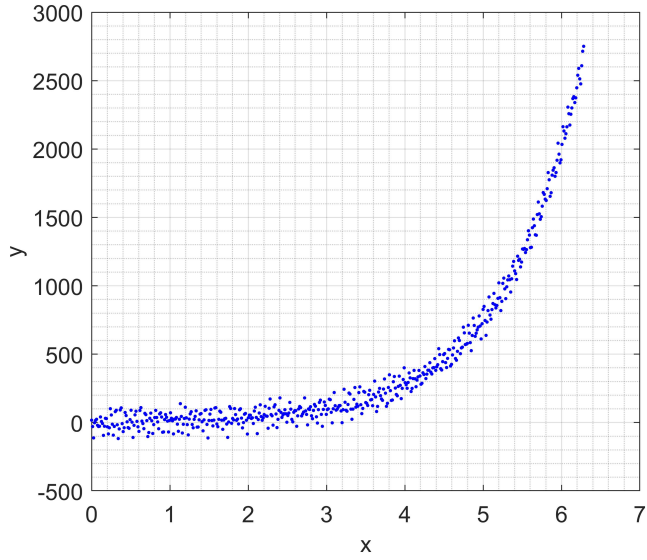


Рисунок 2.1 — Исходные данные для аппроксимации

Показанные на рисунке данные получены из зависимости

$$y = 5e^{x}, \tag{2.1}$$

на которую был нанесён шум случайным образом с помощью гармонических функций. В результате получено поле точек, распределённое случайным образом около заданной кривой. Таким образом могут располагаться, например, результаты проведённых экспериментов или данные, получаемые с датчика.

Применение Curve Fitting Application помогает подобрать экспоненциальную зависимость (рис. 2.2) с коэффициентом детерминации $R^2 = 0{,}9919$

$$y = 5{,}114e^{0{,}9967x}. \tag{2.2}$$

Команда `nnstart` позволяет запускать приложение для создания простейших нейронных сетей и их обучения. При выполнении этой команды, появляется окно, как показано на рисунке 2.3. Здесь мы можем выбрать тип нейронной сети, которую решили создать. В данном случае нас интересует сглаживающая нейронная сеть, позволяющая аппроксимировать данные. Со-

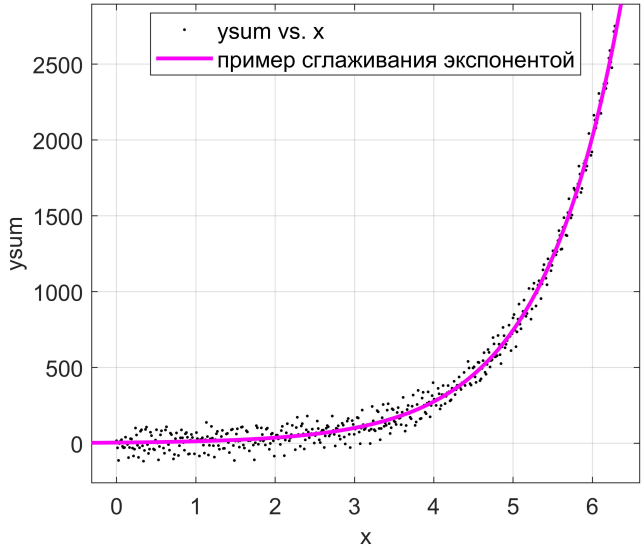


Рисунок 2.2 — Результат аппроксимации с помощью Curve Fitting Application

ответственно нажмём на кнопку *Fitting app*. Аналогичный результат можно было бы получить, выполнив команду nftool.

Инструмент *Neural Fitting app* позволяет создавать нейронные сети с количеством входов, соответствующих количеству аргументов, количество выходов, соответствующих количеству целей, и скрытым слоем содержащем нейроны с сигмоидной функцией активации. Количество нейронов сети может быть задано в зависимости от задачи по желанию пользователя. Слой нейронов на выходе содержит линейную функцию активации и количество нейронов по количеству целей.

Работая в диалоговом режиме, *Neural Fitting app* попросит вести данные для создания нейронной сети. Выберем на вход матрицу, соответствующую переменной x (рис. 2.4), а на выход в качестве целей значение y с шумами.

Следует помнить, что каждая строка матрицы входных данных является отдельным аргументом. То есть если матрица x имеет 10 строк, то аппроксимация будет проводиться по 10 факторам. Количество строк целей является количеством выходных параметров. Фактически аппроксимация с

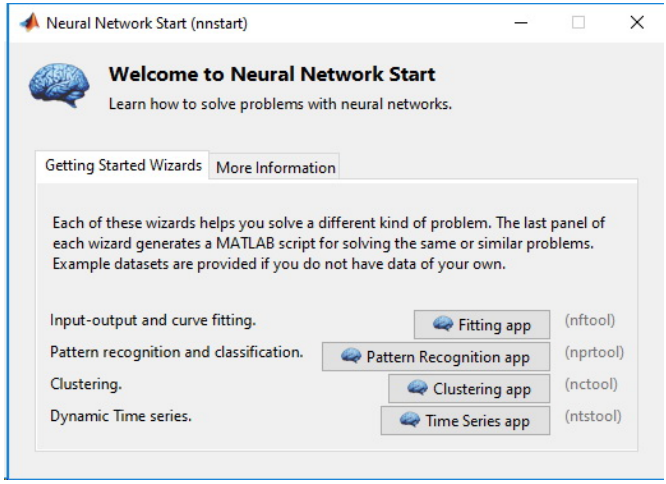


Рисунок 2.3 — Стартовое окно для создания нейронных сетей

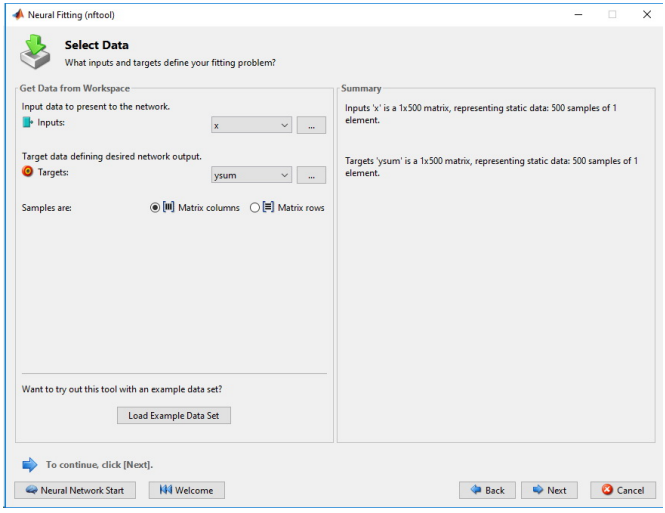


Рисунок 2.4 — Ввод данных для создания и обучения нейронной сети

помощью нейронной сети не ограничивает нас в размерности матрицы аппроксимируемых данных. Но все данные входов и данные целей должны быть объединены в отдельные матрицы.

В следующем диалоговом окне необходимо определить процентное соотношение данных, которые пойдут на обучение, проверку и тестирование нейронной сети. В процессе обучения, в зависимости от выбранных методов, проверка может не осуществляться. Она необходима для предотвращения переобучения сети. Процесс обучения будет прекращён, если сеть перестанет улучшаться. Это относится не ко всем методам. Тестовые данные будут использованы для тестирования сети по окончанию обучения.

Важным этапом при задании параметров сети является выбор количества нейронов в скрытом слое (рис. 2.5). Здесь необходимо иметь в виду, что

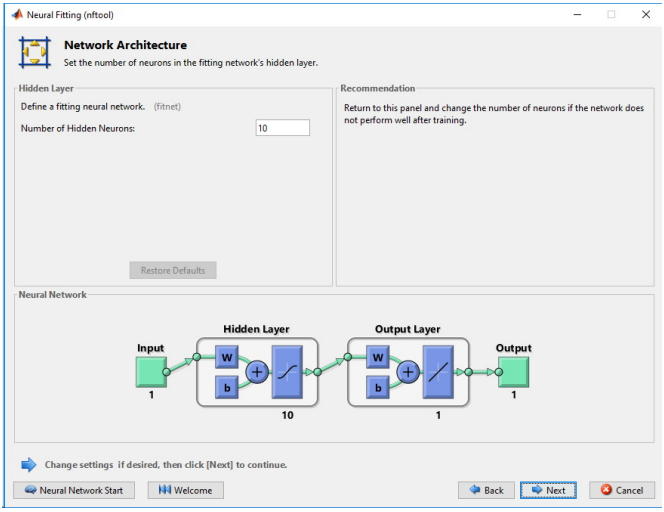


Рисунок 2.5 — Выбор количества нейронов в скрытом слое

малое количество нейронов приводит к хорошему сглаживанию данных, а их чрезмерное увеличение — к появлению шума.

Правильный подбор количества нейронов позволяет описать практически любые зависимости с хорошей определённостью и приемлемой гладкости кривых. Количество нейронов, необходимое для получения качественного результата, как правило, подбирается экспериментальным путём. В случае двухмерной аппроксимации, что в отдельных задачах вполне оправдано, для описания плавного участка кривой достаточно от 2 до 4 нейронов. Такое же количество подходит для каждого резкого перехода между участками.

Проведём обучение сети с 2, 4-мя и 20-ю нейронами. Сравним полученные результаты.

В следующем диалоговом окне (рис. 2.6) необходимо выбрать алгоритм обучения нажать кнопку *«Train»*.

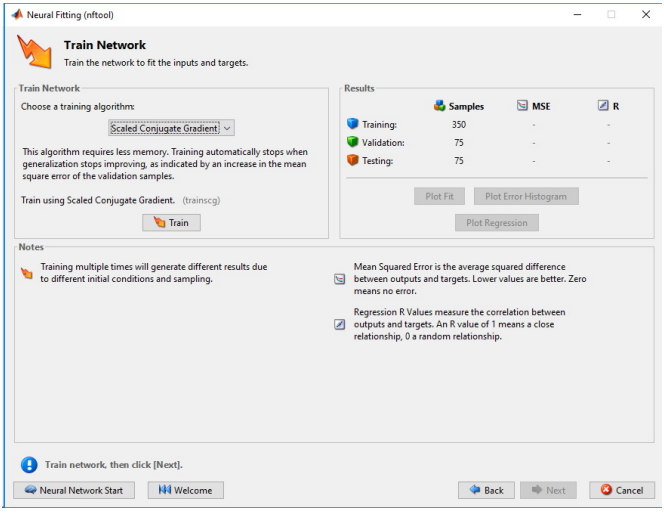


Рисунок 2.6 — Запуск обучения нейронной сети

Сглаживающие аппроксимирующие нейронные сети обучаются методом обратного распространения ошибки. Для минимизации ошибки используется различные алгоритмы оптимизация такие как: Левенберга–Марквардта, Байесовской регуляризации, масштабируемых сопряжённых градиентов. В разных задачах могут быть с разным успехом использованы различные методы обучения. Хотя в целом они показывают достаточно близкие результаты. Особенно методы Левенберга–Марквардта и Байесовской регуляризации. Зачастую алгоритм Левенберга–Марквардта позволяет провести обучение при меньшем количестве эпох, а алгоритм Байесовской регуляризации позволяет более точно аппроксимировать данные.

Под эпохой понимают один этап обучения, в котором в сеть проходит обучение по всей совокупности, имеющихся для этого данных. Так как изначально веса нейронов сети задаются случайным образом, количество эпох обучения для получения наилучшего результата может быть различным.

Выберем метод Левенберга–Марквардта. В процессе обучения открывается диалоговое окно, позволяющее увидеть: схему сети, изменение параметров ее обучения и графики, характеризующая эти параметры.

Как видно из рисунка 2.7, процесс обучения закончился за 92 приближения или эпохи, практически мгновенно.

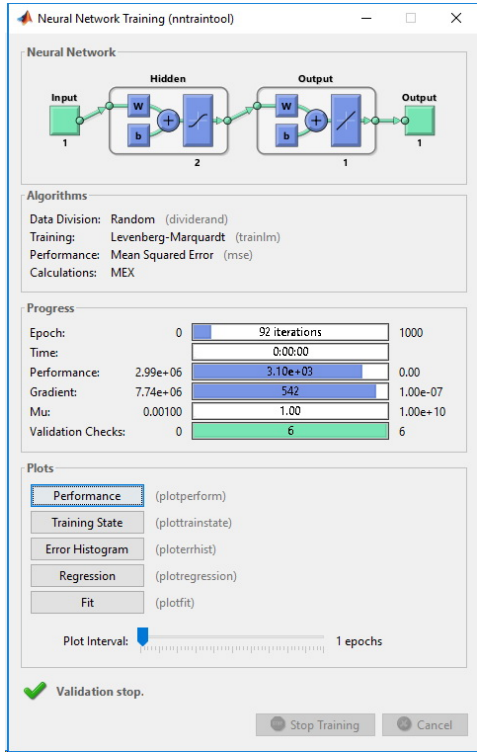


Рисунок 2.7 — Процесс обучения нейронной сети

Для оценки качества обучения нейронной сети в интерфейсе выводятся значения: среднеквадратичной ошибки, градиента функции, параметра настройки алгоритма μ и количество ошибок при контроле обучения.

Имеется возможность посмотреть как проходил процесс обучения, получив график (рис. 2.8) нажатием кнопки *«Training State»*.

По графику видно что основной прогресс был достигнут в течение первых эпох обучения. Это же подтверждает и график (рис. 2.9) среднеквадратичной ошибки. Минимальное значение было достигнуто на 86 эпохе, однако уже после 5–6 эпох значение изменялась несущественно.

Качество аппроксимации легко определить визуально по графику полученной функции сглаживания на рис. 2.10.

Наличие большого количества данных, на приведённом графике, не позволяет чётко увидеть линию, построенную нейронной сетью. Рассмотрим

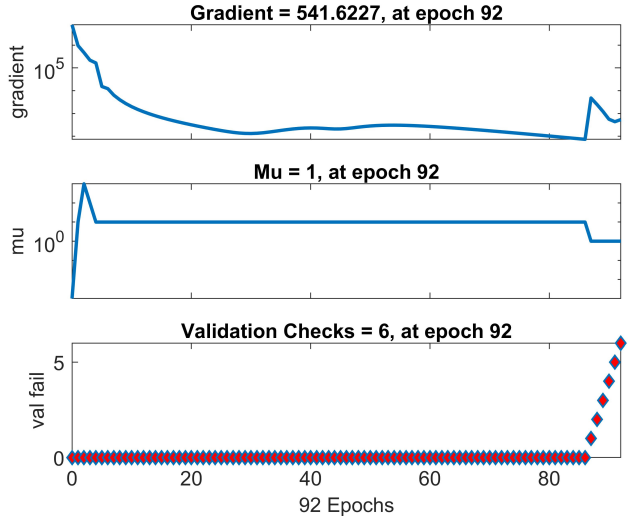


Рисунок 2.8 — Изменение параметров в процессе обучения нейронной сети

эту линию подробнее в дальнейшем, в сравнении с другими способами аппроксимации и теоретической кривой, на основе которой строились данные.

Очевидно одним из наиболее важных графиков, дающих оценку качеству обучения, является регрессионная зависимость между выходами и целями сети. Такая регрессионная связь оценивается по данным отобранным для обучения, для проверки и для тестирования отдельно (рис. 2.11). Но в конечном итоге интересует регрессионная связь для всего массива данных. Как видно из графиков, коэффициент корреляции составляет $R = 0{,}99592$, а $R^2 = 0{,}9919$. То есть нейронная сеть с двумя нейронами имеет такой же коэффициент детерминации как и уравнение (2.2), полученное традиционными статистическими методами.

Если результаты обучения нейронной сети нас не удовлетворяют, обучение может быть осуществлено повторно и неоднократно. Возможно возвращение и изменение количества нейронов в сети. Если результаты обучения удовлетворительны, нейронную сеть следует сохранить. *Naural Fitting app* позволяет сохранить нейронную сеть (рис. 2.12) в виде функции Matlab или Simulink диаграммы. Полученные результаты в виде функции или Simulink диаграммы в дальнейшем позволяют проводить все действия над нейронной сетью, как над обычный функцией.

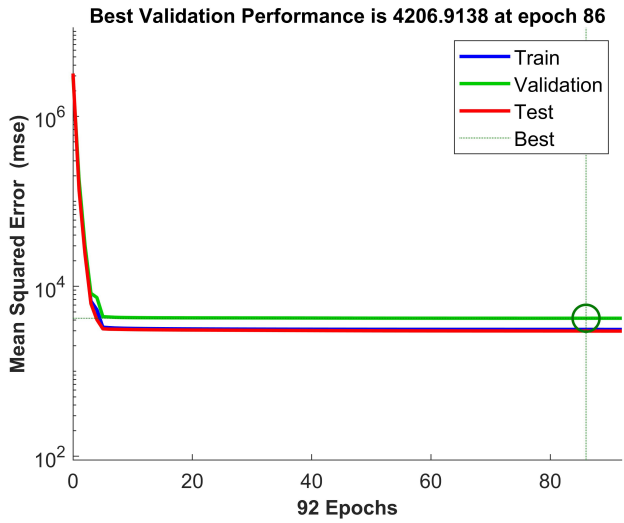


Рисунок 2.9 — Изменение среднеквадратичной ошибки в процессе обучения нейронной сети

Все выполненные процедуры могут быть автоматизированный с помощью скрипта Matlab. Naural Network Toolbox имеет полный набор команд позволяющий создавать и тонко настраивать нейронные сети и процесс обучения. *Naural Network app* даёт возможность автоматически сгенерировать скрипт для создания и обучения сети по фактически произведённым в приложении действиям. Пример такого скрипта для нейронной сети с двумя нейронами показан ниже.

```
x = x;
t = ysum;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
hiddenLayerSize = 2;
net = fitnet(hiddenLayerSize,trainFcn);
```

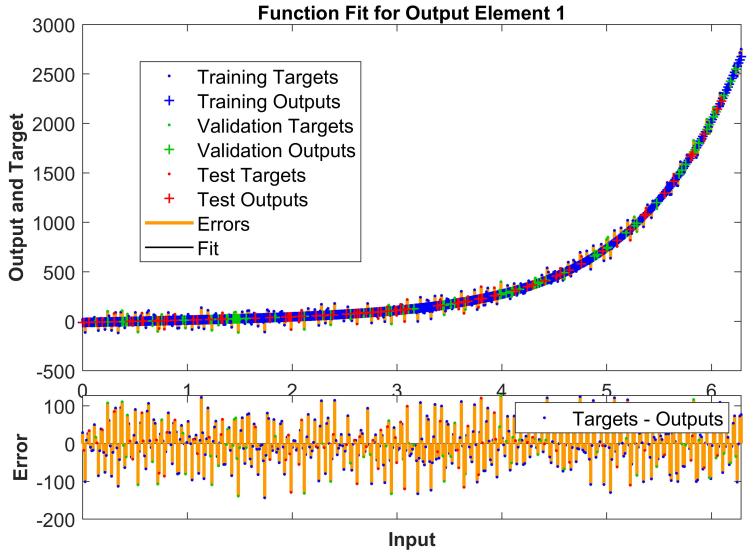


Рисунок 2.10 — Функция сглаживания нейронной сети и ошибки аппроксимации

```
23
24  % Setup Division of Data for Training, Validation, Testing
25  net.divideParam.trainRatio = 70/100;
26  net.divideParam.valRatio = 15/100;
27  net.divideParam.testRatio = 15/100;
28
29  % Train the Network
30  [net,tr] = train(net,x,t);
31
32  % Test the Network
33  y = net(x);
34  e = gsubtract(t,y);
35  performance = perform(net,t,y)
36
37  % View the Network
38  view(net)
39
40  % Plots
```

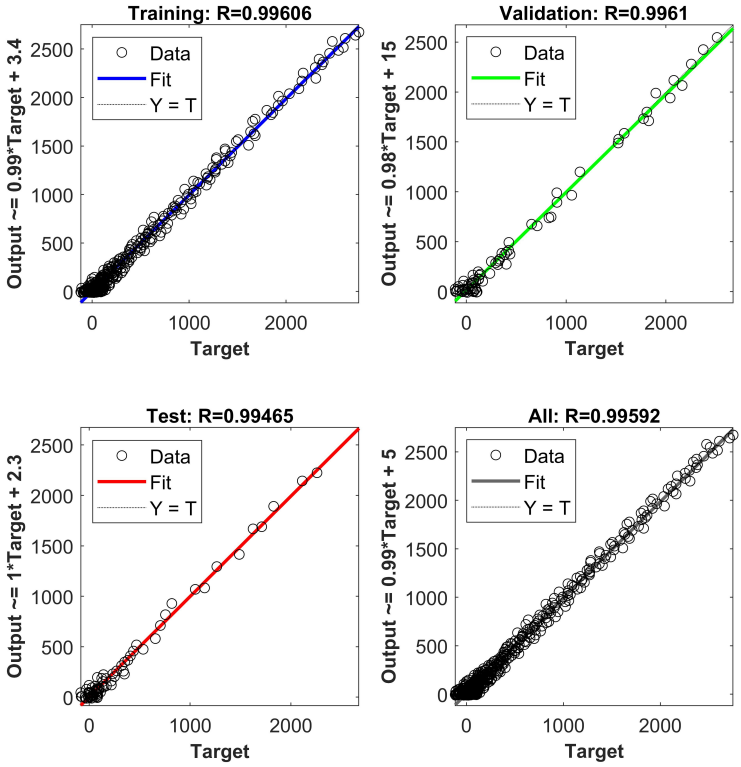


Рисунок 2.11 — Регрессионная связь между выходами и целями нейронной сети

```
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, ploterrhist(e)
%figure, plotregression(t,y)
%figure, plotfit(net,x,t)
```

Рассмотрим результаты аппроксимации данных различными способами (рис. 2.13).

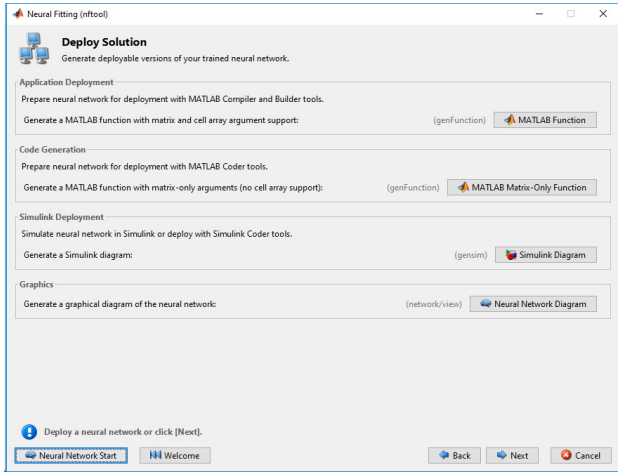


Рисунок 2.12 — Сохранение нейронной сети

Как видим нейронные сети справились с задачей и достаточно хорошо описали исходные данные. Некоторое неудовлетворение вызывает график построенный с помощью нейронной сети из 20 нейронов. Он имеет колебания и пытается описать те погрешности, которые были заданы специально, наложением шумов. Сглаживающая функция проявляется недостаточно. Это надо учитывать при использовании нейронных сетей выбирать их параметры в зависимости от целей и задач. В ряде случаев эти свойства нейронных сетей являются положительными и позволяет описать всплески, что сложно или невозможно сделать аналитическими зависимостями.

Более подробный взгляд на графики (рис. 2.14) показывает, что на различных участках степень приближения разных кривых к теоретической отличается.

В начале рассматриваемого участка по оси абсцисс нейронные сети показывают некоторую погрешность, однако в дальнейшем графики, построены нейронными сетями с двумя и четырьмя нейронами, находятся гораздо ближе к теоретической кривой чем график аппроксимирующей экспоненциальной функции. На отдельных участках графики фактически совпадают.

Нейронные сети с малым количеством нейронов могут фактически заменять аппроксимацию простейшими функциями. Основные же их преимущества проявляются в более сложных случаях.

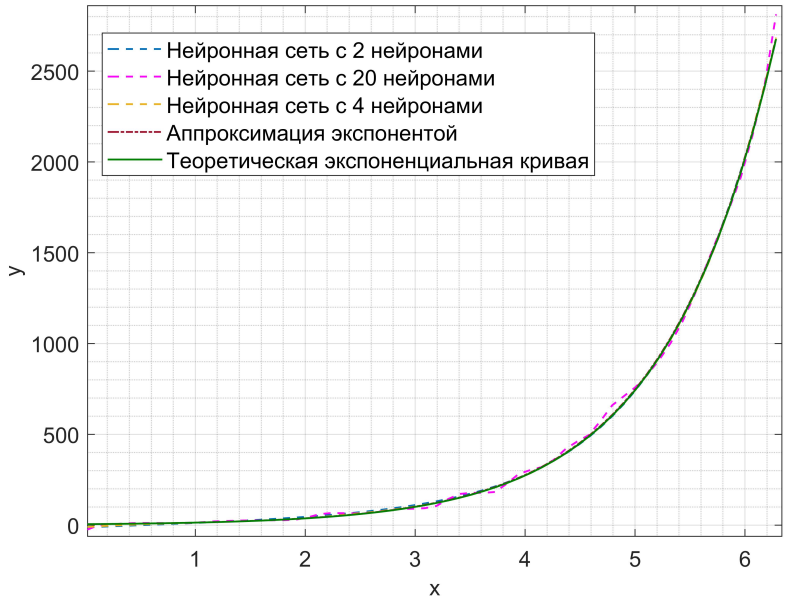


Рисунок 2.13 — Результаты аппроксимации данных различными способами

2.1.2 Применение нейронной сети для аппроксимации кусочно-заданной функции

Очень часто на практике встречаются графики, построенные по экспериментальным данным, которые невозможно описать одной зависимостью. В лучшем случае такие графики могут описываться кусочно-заданными функциями, то есть аппроксимировать данные можно только на отдельных участках. Проблема может заключаться и в том, что эти отдельные аналитические зависимости необходимо сгладить в местах переходов.

Практическим примером может являться график изменения крутящего момента двигателя (рис. 2.15) по внешней скоростной характеристике для двигателя имеющего наддув впускного воздуха.

Данный график имеет 3 характерных участка и один достаточно резкий переход от участка нарастания эффективного крутящего момента на участок его ограничения. Создадим нейронную сеть для аппроксимации этой внешней скоростной характеристики. В скрытом слое разместим 12 нейронов. Внешняя скоростная характеристика построена по 30 точкам. В связи с малым

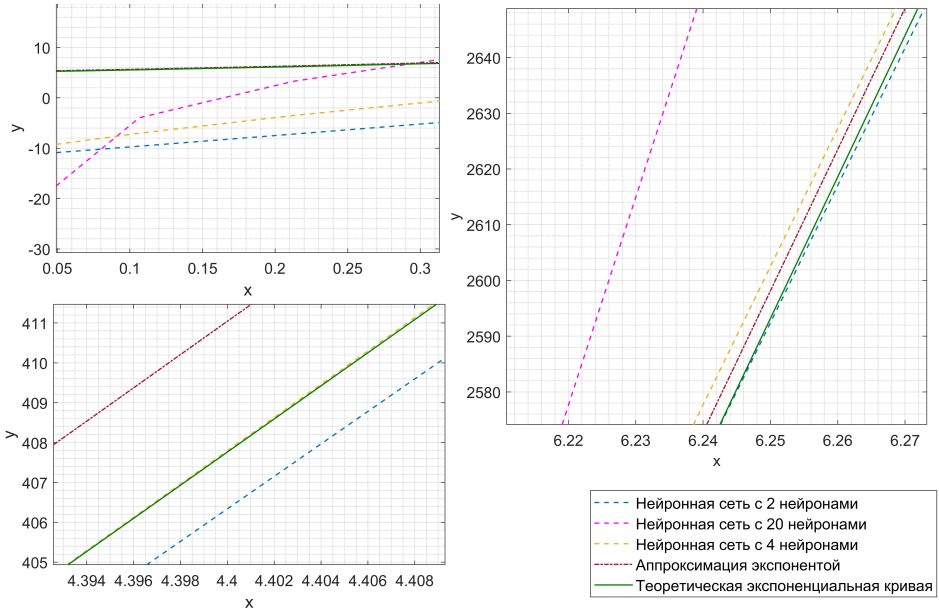


Рисунок 2.14 — Детализация графиков аппроксимации данных различными способами

количеством данных на проверку и тестирование сети отведём всего лишь по 5 %.

Количество точек, используемое для обучения сети в данном случае критически малое. Чтобы сократить отбор точек на проверку, для обучения использовался метод Байесовской регуляризации. По сравнению с алгоритмом Левенберга–Марквардта при малом количестве точек Байесовская регуляризация даёт лучший результат. Это видно как на графиках регрессии (рис. 2.16) при обучении, так и при сопоставлении внешних скоростных характеристик. Коэффициент детерминации полученной нейронной сети составляет $R^2 = 0{,}99986$. Это достаточно высокий результат, а отличие его от 1 говорит о сглаживании некоторых данных. Это хорошо заметно на графике (рис. 2.17). В случае если необходимо, чтобы график прошёл через все точки, можно увеличить количество нейронов в сети. Однако если точки экспериментальные, в этом нет необходимости.

В подобных случаях применение нейронных сетей выглядит наиболее целесообразным.

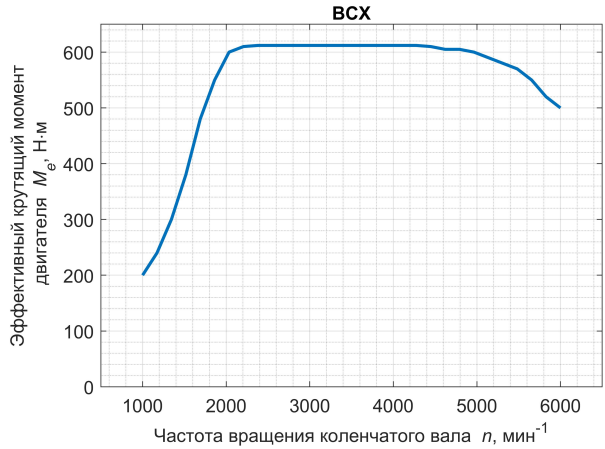


Рисунок 2.15 — Пример внешней скоростной характеристики двигателя с наддувом

2.1.3 Аппроксимация поверхности с помощью нейронных сетей

Аппроксимация многомерных данных, например, поверхностей имеет своим особенности. Выделим следующие из них:

— определение рационального количества нейронов для аппроксимации поверхности;

— более тщательная подготовка данных для обучения нейронной сети;

— выбор оптимального метода обучения и более тщательная настройка параметров обучения.

Наличие дополнительных факторов, влияющих на конечный результат, может приводить к появлению дополнительных изгибов поверхности. Решение вопроса заключается в правильном выборе количества нейронов в сети. Малое количество приведёт к существенному сглаживанию, а излишне может привести к необоснованным колебаниям поверхности.

Естественно форма поверхности является определяющей в этом вопросе. На рисунке 2.18 показаны данные серийной прошивки 22ус041s контроллера автомобиля Нива-Шевроле Bosch M7.9.7 с двигателем 21230-1411020-40, рассчитанного на выполнение норм по токсичности, известных как ЕВРО 4.

Поверхность, описывающая изменение угла опережения зажигания в зависимости от нагрузки и частоты вращения коленчатого вала, является достаточно плавной и, как будет показано дальше, не требует большого

количества нейронов для описания с помощью нейронной сети. Изменение коэффициента коррекции пульсаций потока имеет большое количество пиков и впадин. Сглаживающая нейронная сеть может дать результат, существенно отличающийся от необходимого. Для описания подобный поверхности необходимо иметь большое количество данных, равномерно распределенных по всей поверхности.

Калибровки двигателя обычно содержит информацию в виде массивов. Учитывая ограниченность в ресурсе и необходимость получить заданное быстродействие контроллера массивы, содержащие данные по показанным на рисунке калибровкам, имеют размерность 12×16 и 14×14. Такое количе-

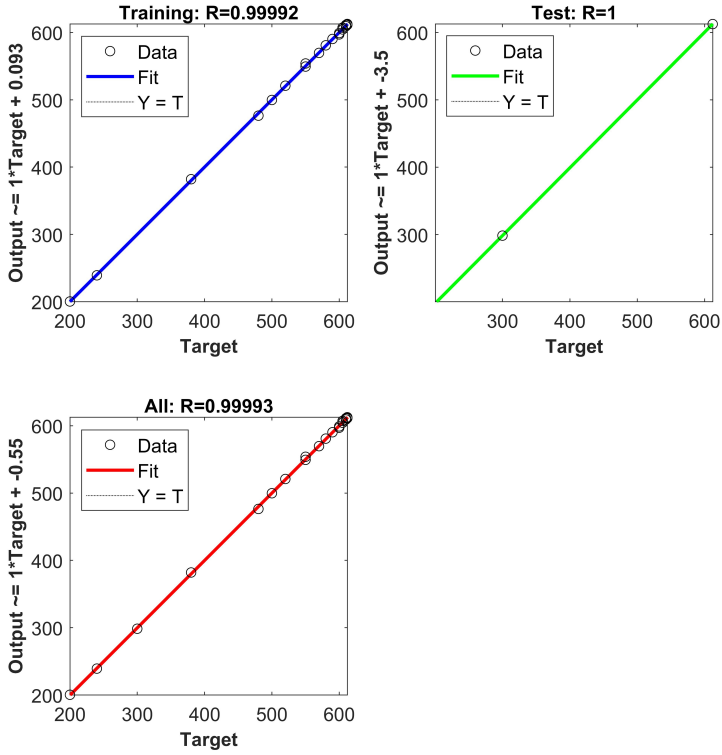


Рисунок 2.16 — Регрессионная связь между выходами и целями при обучении нейронной сети для аппроксимации внешней скоростной характеристики

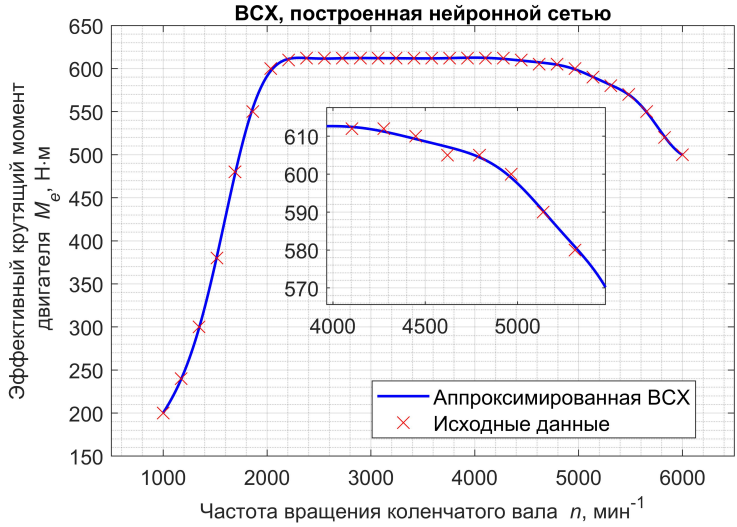
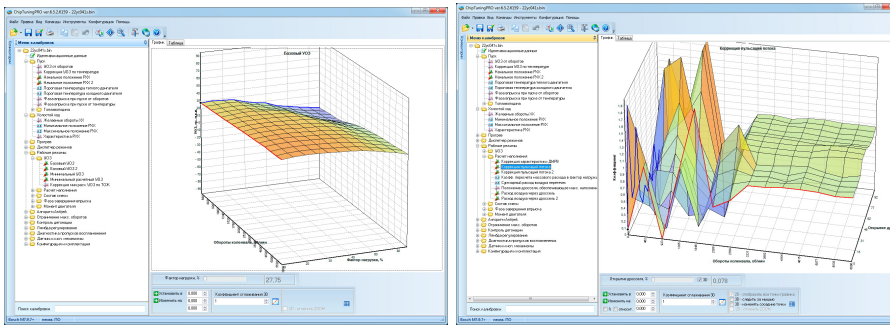


Рисунок 2.17 — Результаты аппроксимации внешней скоростной характеристики с помощью нейронной сети с 12 нейронами в скрытом слое



а) б)

а) базовый угол опережения зажигания на рабочих режимах двигателя;
б) коэффициент коррекции пульсаций потока при расчёте наполнения двигателя

Рисунок 2.18 — Данные калибровки двигателя 21230-1411020-40 из серийной прошивки 22ус041s контроллера Bosch M7.9.7

ство точек может быть достаточным для обучения сети, описывающей только плавную поверхность.

Отдельным вопросом является подготовка данных для обучения нейронной сети. Даже на примере простейших сетей, рассматриваемых в данном пособии, можно увидеть, что в зависимости от качества подготовки данных получаются принципиально различные результаты.

Во-первых, данные должны быть структурированы.

Входные данные должны представлять собой матрицу, каждая строка которой представляет собой значение отдельного фактора. Матрица целей представляет собой значение параметров на выходе из нейронной сети. Обе матрицы должны быть согласованы между собой.

Во-вторых, данные в матрицах должны быть перемешаны случайным образом. Если обучение проходит по структурированным матрицам по поддающимся логическому описанию законам, например имеется большое количество данных одна строка которых сортирована по возрастанию фактора, нейронная сеть может не поддаваться обучению. Это требование является особенно актуальным, если нейронная сеть обучается на большом количестве данных. В этом случае за одну эпоху она получает существенное обучение и весовые коэффициенты нейронов в дальнейшем сложно изменить. При обучении на тех же данных, но перемешанных случайным образом, результат обучения оказывается положительным.

При обучении нейронной сети, прежде чем делать окончательные выводы и принимать решения, следует попробовать обучение различными методами. Опыт работы с аппроксимирующими нейронными сетями показывает, что алгоритмы обратного распространения с оптимизацией по методам Левенберга–Марквардта и Байесовской регуляризации являются вполне конкурентными. Обучение с использованием алгоритма Левенберга–Марквардта является более быстрым. Он может дать весьма хороший результат на плавных кривых или поверхностях при достаточном количестве точек. Байесовская регуляризация позволяет лучше обучить нейронную сеть на плавных кривых с малым количеством точек и на сложных поверхностях при наличии достаточного количества данных, например как на рисунке 2.18, б. Как правило использование Байесовской регуляризации требует большего количества эпох обучения. При большом массиве данных параметров заложенных в команду `train` по умолчанию может быть недостаточно. В этом случае максимальное количество эпох необходимо увеличить командой `net.trainParam.epochs`.

При необходимости могут быть откорректированы и другие параметры, заданные по умолчанию. Более подробную информацию по настройке обучения можно найти в справочной системе Matlab или на сайте компании Mathworks [2].

Рассмотрим подготовку и обучение нейронной сети, описывающей зависимость угла опережения зажигания от нагрузки и частоты вращения коленчатого вала двигателя.

Из упоминаемой ранее прошивки контроллера с помощью программы ChipTuningPRO [1] была экспортирована таблица, соответствующая поверхности на рисунке 2.18, а. Указанная таблица приведена ниже (табл. 2.1).

Таблица 2.1 — Базовый угол опережения зажигания

Нагрузка, %	Угол опережения зажигания °п.к.в. при частоте вращения коленчатого вала, мин$^{-1}$							
	680	800	920	1000	1240	1520	1760	2000
10	27,75	28,5	29,25	30	30,75	31,5	32,25	33
15	28,5	29,25	30	30,75	32,25	33	33,75	34,5
20	27,75	30	30,75	31,5	33	33,75	34,5	35,25
25	27	29,25	30	31,5	33	33,75	34,5	35,25
30	22,5	24,75	27	28,5	31,5	33	33,75	34,5
35	19,5	21,75	24	26,25	30,75	32,25	33	33,75
40	17,25	18,75	20,25	22,5	28,5	31,5	32,25	33
50	12	14,25	16,5	18	22,5	26,25	29,25	31,5
60	9	10,5	12	13,5	16,5	20,25	24,75	27,75
70	6,75	7,5	9	9,75	12	15,75	20,25	22,5
80	4,5	5,25	6	6,75	8,25	12	15,75	18
100	2,25	3	3,75	4,5	6	9	12,75	15,75
Нагрузка, %	Угол опережения зажигания °п.к.в. при частоте вращения коленчатого вала, мин$^{-1}$							
	2520	3000	3520	4000	4520	5000	5520	6000
10	33,75	34,5	35,25	36	36,75	36,75	36,75	37,5
15	35,25	36	36,75	37,5	38,25	38,25	38,25	39
20	36	36,75	37,5	38,25	39	39	39	39,75
25	36	36,75	37,5	38,25	39	39	39	39,75
30	35,25	36	36,75	37,5	38,25	38,25	38,25	39
35	34,5	35,25	36	36,75	37,5	37,5	37,5	38,25
40	33,75	34,5	35,25	36	36,75	36,75	36,75	37,5
50	32,25	33	33,75	34,5	35,25	35,25	35,25	36
60	30	30,75	31,5	32,25	33	33	33	33,75
70	24,75	26,25	27	27,75	28,5	29,25	30	30,75
80	20,25	22,5	24	24,75	25,5	26,25	27	27,75
100	17,25	18,75	20,25	21	21,75	22,5	23,25	24

Из указанной таблицы можем выделить столбец и строку содержащие значение факторов влияющих на угол опережения зажигания. Их, соответ-

ственно, 12 и 16. Количество углов опережения зажигания $12 \cdot 16 = 192$. Мы имеем явное несоответствие количества данных. Создадим сетку соответствия значений факторов. Для этого воспользуемся командой meshgrid:

```
[X,Y]=meshgrid(UOZnkv,UOZnagr)
```

`UOZnkv` и `UOZnagr` переменные, содержащее значение оборотов коленчатого вала и фактора нагрузки на двигатель, записанные в головке и боковике таблицы 2.1.

В результате получаем 2 матрицы размером 12×16 со значениями факторов. Теперь полученные массивы можно превратить в строки, то есть изменить размер на 1×192:

```
X=reshape(X,[1,192]);
Y=reshape(Y,[1,192]);
Z=reshape(UOZdate,[1,192]);
```

На следующем шаге подготовки необходимо случайным образом перемешать данные и при этом не потерять их взаимосвязь. Для перемешивания в случайном порядке можно воспользоваться генератором случайных чисел `rand`. Это команда генерирует матрицу со случайными числами в диапазоне от нуля до единицы. Естественно, что размер этой матрицы должен быть 1×192. Полученный массив случайных чисел разместим в первой строке обобщённого массива данных. Во второй и третьей строке поместим строки аргументов, а в четвёртой строке данные, соответствующие углу опережения зажигания. После этого достаточно отсортировать массив по первой строке с помощью команды `sortrows` и извлечь данные для обучения нейронной сети. То есть вторую и третью строку, как входы в нейронную сеть, и четвёртую, в качестве целей. Описанный алгоритм приведён ниже в программном коде:

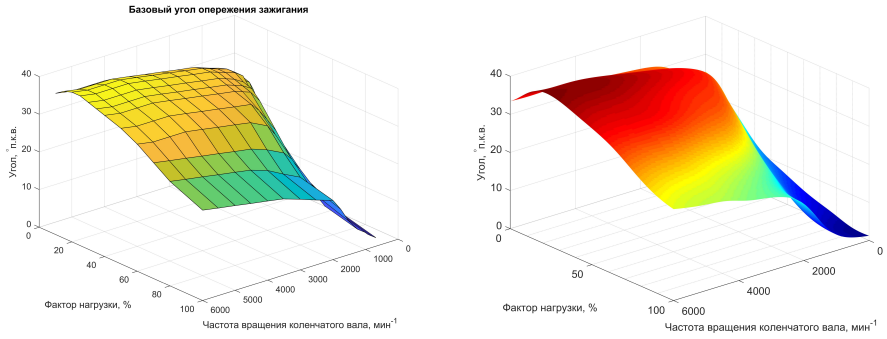
```
allrand=[rand(1,192);X;Y;Z] % Объединение данных в массив
randallsort=sortrows(allrandall.').' % сортировка по первой строке,
        с предварительным и последующим транспонированием
NNfaktor=randallsort(2:3,:) % выделение входов нейронной сети
NNtarget=randallsort(4,:) % выделение целей нейронной сети
```

Полученные переменные `NNfaktor` и `NNtarget`, содержат необходимые данные для обучения и находятся в рабочем пространстве Matlab. Процесс обучения может быть проведён, в соответствии с описанием из предыдущих подразделов.

После некоторого повторения процесса обучения, полученная нейронная сеть обеспечила на выходе данные с коэффициентом определённости выше $R^2 = 0,998$. Окончательный вывод адекватности данных, получаемых с помощью нейронной сети, можно сделать только сравнением исходного и конечного графиков поверхности. Следует обращать внимание на то, чтобы точки выдаваемые нейронной сетью не только совпадали с исходными

данными обучения, но и адекватно описывали поведение поверхности в промежутках.

Как видно из графиков на рис. 2.19, построенная с помощью нейронной сети поверхность очень хорошо согласуется с исходной.



а) б)

а) данные калибровки; б) аппроксимация нейронной сетью

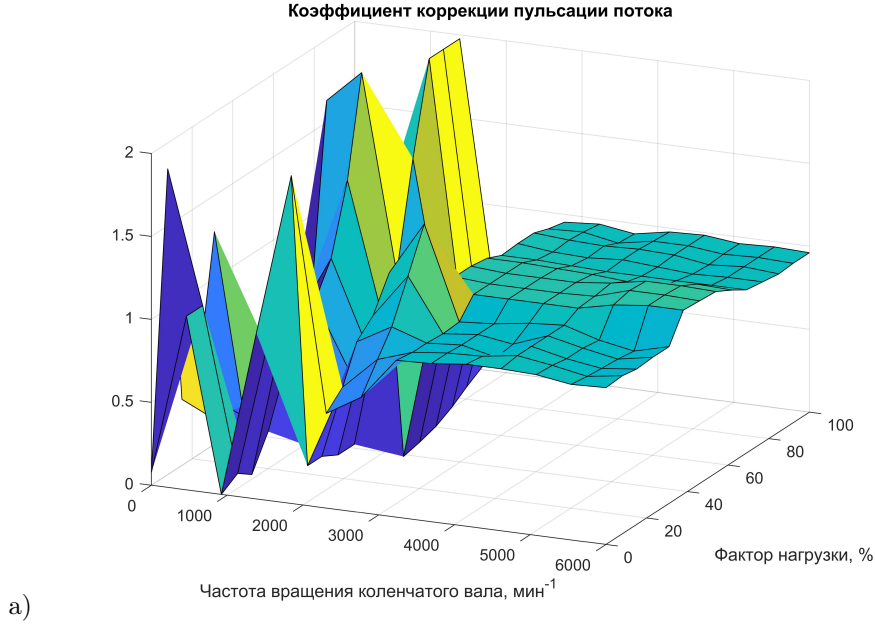
Рисунок 2.19 — Базовый угол опережения зажигания на рабочих режимах двигателя

Для описания и аппроксимации данной поверхности потребовалась нейронная сеть, содержащая 10 нейронов в скрытом слое. Количество нейронов может быть ещё сокращено с приемлемым результатом.

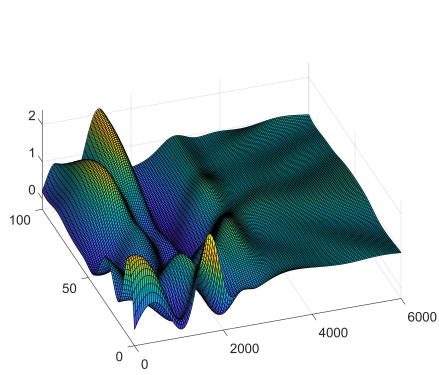
Несколько иначе дело обстоит с аппроксимацией поверхности, показанной на рисунке 2.18, б. Обучение нейронных сетей на небольшом количестве данных даёт похожие поверхности. Но это не адекватный результат, так как при малом количестве нейронов сглаживаются пики и впадины на поверхности (рис. 2.20, б), а при достаточном количестве нейронов и коэффициенте корреляции практически равном единице получается поверхность, которая проходит через все точки, но между ними изгибается случайным образом (рис. 2.20, в).

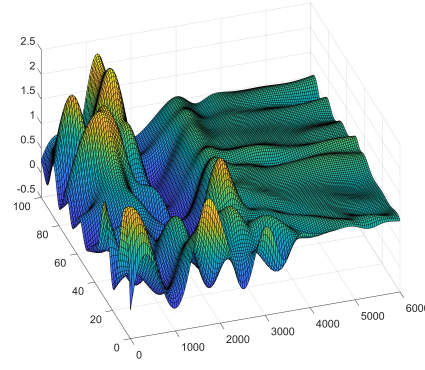
Если стоит задача максимально повторить при аппроксимации данные аналогичные показанным на рисунке 2.20, а, необходимо при подготовке данных для обучения увеличить их количество. Например за счёт применения линейной интерполяции. Это можно выполнить с помощью команды `interp2`. Пример ее применения может выглядеть следующим образом:

```
ZvL = interp2(X,Y,date,Xvg,Yvg,'linear')
```



а)





б) в)

а) данные калибровки; б) аппроксимация нейронной сетью с 25 нейронами; в) аппроксимация нейронной сетью с 50 нейронами;

Рисунок 2.20 — Результаты применения нейронных сетей для аппроксимации поверхность с большим количеством пиков и впадин при малом количестве данных

Здесь `X,Y,date` известные точки на поверхности. А `Xvg,Yvg` аргументы, для которых необходимо определить значение функции `ZvL`. Параметр `'linear'` определяет вид интерполяции. В данном случае она линейная. Полученные массивы `Xvg, Yvg, ZvL` объединяются и перемешиваются случайным образом аналогично описанному выше.

В приведённом примере количество точек было увеличено в 1500 раз. Это фактически сняло ограничение на количество нейронов в сети. Увеличение количества данных привело к увеличению времени обучение сети. На обучение сети из 50 нейронов потребовалось более 2500 эпох обучения по методу Байесовской регуляризации.

Полученная нейронная сеть позволяет построить поверхность (рис. 2.21) достаточно близкую к начальным данным (рис. 2.20, а). Однако это не предел и при необходимости результат может быть существенно улучшен.

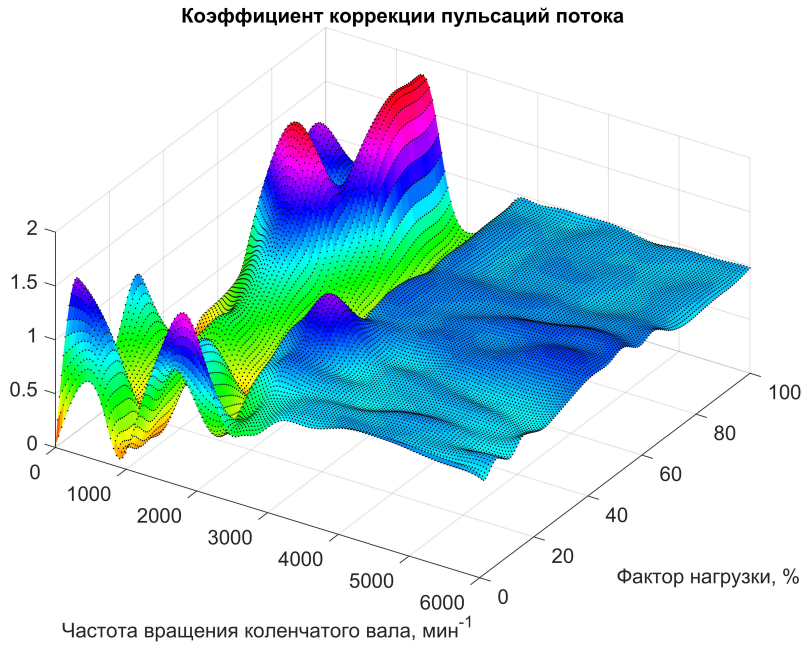


Рисунок 2.21 — Результат аппроксимации коэффициента коррекции пульсаций потока с помощью нейронной сети с 50 нейронами в скрытом слое

Полученные нейронные сети могут быть сохранены в качестве Simulink диаграммы, Matlab функции или исполняемого файла. При необходимости возможна генерация C/C++ кода для последующего использования нейронных сетей на других устройствах.

Процесс создания и обучения нейронной сети может быть полностью автоматизирован с помощью команд Matlab. Ниже приведена часть скрипта Matlab с помощью, которого получена и обучена нейронная сеть, позволившая построить поверхность на рис. 2.21.

```
% Создаем аппроксимирующую нейросеть
hiddenLayerSize = 50; % количество нейронов в уровне
CF3dkorrektL= fitnet(hiddenLayerSize); % сеть создана
view(CF3dkorrektL) % Просмотр структуры сети
% Распределение имеющихся данных для обучения сети и проверок
CF3dkorrektL.divideParam.trainRatio = 90/100; % количество данных,
используемых в качестве целей
CF3dkorrektL.divideParam.valRatio = 5/100; %  количество данных,
используемых для проверки
CF3dkorrektL.divideParam.testRatio = 5/100; % количество данных,
используемых для последующего тестирования сети
Определение обучающего алгоритма и параметров
%  net.trainFcn = 'trainlm';
CF3dkorrektLПостроить.trainFcn = 'trainbr'; Байесовская регуляризация
% net.trainFcn = 'trainscg';
CF3dkorrektL.trainParam.epochs = 3000 % максимальное количество эпох
обучения
CF3dkorrektL.trainParam.showWindow = true % true, false - показывать
или нет окно обучения
CF3dkorrektL.trainParam.min_grad =1e-9 % минимальный градиент
производительности
CF3dkorrektL.trainParam.mu_max = 1e14 % максимум mu
% Обучение
[CF3dkorrektL, tr]= train(CF3dkorrektL,NNfaktorvLg,NNtargetvLg)
% Тест сети
zafnnets = CF3dkorrektL(NNfaktorvLg) % результаты выходов после обучения
errors = gsubtract(NNtargetvLg, zafnnets);
performance = perform(CF3dkorrektL, NNtargetvLg, zafnnets)
tInd = tr.testInd;
tstOutputs = CF3dkorrektL(NNfaktorvLg(:, tInd));
tstPerform = perform(CF3dkorrektL, NNtargetvLg(tInd), tstOutputs)
% Построение графиков теста
plotregression(NNtargetvLg,zafnnets,'Perpeссия z_{af}');
plottrainstate(tr);
plotperform(tr);
```

```
Генерация MATLAB функции для дальнейшей работы с сетью
genFunction(CF3dkorrektL,'CF3dkorrekt50L'); % 'CF3dkorrekt50L' - имя
файла в рабочем каталоге
```

2.2 Применение нейронной сети для прогнозирования временных рядов

Рекуррентные нейронные сети (RNN)[1] имеют более сложную структуру. Их особенностью является влияние выходных значений сети на входы определённых нейронов и слоев. Такие сети как бы обладают памятью. То есть свои последующие значения они вычисляются с учетом результатов вычисления предыдущих значений. Это расширяет возможность применения нейронных сетей. То есть подобная сеть не только аппроксимирует данные текущего времени, но и позволяет прогнозировать изменения в дальнейшем.

Одной из особенностей рекуррентных сетей является сложность в их обучение. Данные для сети должны быть структурированы в последовательные ряды.

Рекуррентные нейронные сети могут иметь структуры различной сложности. Одними из широко известных на данный момент стали LSTM сети[2]. Сети с, так называемой, длинной краткосрочной памятью. Они позволили добиться существенных результатов в работе с текстами. Фактически текстовые данные являются последовательностью некоторых данных. А рекуррентные сети как раз и работают с последовательностями. Как и большинство рекуррентных нейронных сетей, LSTM-сеть является достаточно универсальной. При достаточном числе элементов сети она теоретически может выполнить любое вычисление, на которое способен обычный компьютер, для чего необходима соответствующая матрица весов, которая может рассматриваться как программа.

Освоение данных сетей является не только интересной, но и достаточно важной задачей, которая в дальнейшем позволяет решать инженеру задачи прогнозирования совершенно различного характера. Это могут быть

[1] Рекуррентные нейронные сети (англ. Recurrent neural network; RNN) — вид нейронных сетей, где связи между элементами образуют направленную последовательность. Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки

[2] Долгая краткосрочная память (англ. Long short-term memory; LSTM) — разновидность архитектуры рекуррентных нейронных сетей, предложенная в 1997 году Сеппом Хохрайтером и Юргеном Шмидхубером.

задачи экономического и управленческого характера, как например, прогнозирование объемов продаж, объемов поставок запчастей, а также задачи по разработке системы автоматизированного управления. Хотя наибольшую известность подобные сети получили по результатам обработки текстов в задачах распознавания и перевода, они позволяют по обработке видео прогнозировать изменение сцены и тому подобное.

Для инженера, не имеющего глубоких знаний в программировании создание, обучение и применение различного рода рекуррентных нейронных сетей может быть облегчено при использовании Matlab Natural Toolbox. Рассмотрим применение рекуррентных сетей на основе простейших нелинейных авторегрессионных сетей NAR и NARX, представленных в Matlab.

Приведём примеры создания и применения нелинейных авторегрессионных нейронных сетей для прогнозирования временных рядов, как рекурсивных зависимостей[3], различного характера в функции от времени.

В данном подразделе будем рассматривать создание нейронных сетей и их обучение с помощью команд Matlab. Такой подход позволяет автоматизировать обучение нейронной сети и осуществлять более тонкую настройку процесса обучения.

2.2.1 Прогнозирование значений рекурсивных последовательностей с помощью нелинейных авторегрессионных нейронных сетей (NAR)

К рекурсивным числовым последовательностям относится такие известные в математике числовые ряды как числа Фибоначчи или ряд чисел Люка. Они описаны соответствующими математическими формулами. Несмотря на то, что подобные ряды достаточно просты, обучение нейронной сети может оказаться весьма непростой задачей. В качестве примера возьмем числа Люка.

Числа Люка задаются рекуррентной формулой

$$L_n = L_{n-1} + L_{n-2}$$

[3]Рекурсивная функция (от лат. recursio — возвращение) — это числовая функция $f(n)$ числового аргумента n, которая определяет каждое своё последующее значение на основе предыдущих $f(n-1), f(n-2), \ldots$.

с начальными значениями $L_0 = 2$ и $L_1 = 1$. Фактически каждое число может быть определено в зависимости от своего номера n в ряду по формуле

$$L_n = \left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n. \tag{2.3}$$

Подготовим вектора с номерами точек и с числами Люка:

```
clear; clc;
nLukmax=30 % количество точек последовательности для обучения
i=[1:nLukmax] % вектор номеров точек
Luk=((1+5^(1/2))/2).^i+((1-5^(1/2))/2).^i
```

Создаем авторегрессионную нейронную сеть «net» с помощью команды narnet:

```
net = narnet(1:5,200);
```

Команда имеет следующий синтаксис

```
narnet(feedbackDelays,hiddenSizes,trainFcn)
```

где feedbackDelays — количество значений, определяющих задержку обратной связи;

hiddenSizes — Количество нейронов в скрытом слое или слоях;

trainFcn — функция обучения, которая может быть задана по умолчанию или изменяться в дальнейшем.

Проведем подготовку данных временного ряда для обучения сети с помощью команды preparets. Эта команда имеет следующий синтаксис:

```
[Xs,Xi,Ai,Ts,EWs,shift] = preparets(net,Xnf,Tnf,Tf,EW)
```

Данная команда осуществляет подготовку вектора данных для обучения и применения временного ряда. Происходит переформатирование аргументов и целей обучения временных рядов. Подготавливаются данные по слоям задержки.

Аргументы команды:

net — имя нейронной сети;

Xnf — входы без обратной связи;

Tnf — цели без обратной связи;

Tf — цели обратной связи;

EW — ошибка весов (по умолчанию = 1).

Все аргументы должны иметь тип данных cell. Для преобразования можно использовать команду num2cell. Такой же тип данных будут иметь и переменные, возвращаемые функцией:

Xs — смещенные входы;

Xi — начальные состояния задержки ввода;

Ai — первоначальные задержки слоев;

Ts — смещенные цели;

EWs — смещенная ошибка весов;

shift — количество временных шагов, отсеченных с передней стороны X и T, чтобы правильно заполнить Xi и Ai.

Аргументы функции должны быть заполнены последовательно. В случае отсутствия для данного типа нейронной сети какого-либо аргумента вводим пустую ячейку {}. Отсутствующие аргументы в конце вектора могут не вводится.

Тот же принцип будем использовать и для вектора вывода функции. В результате получаем следующий программный код:

```
Lukcell=num2cell(Luk);
[Xs,Xi,Ai,Ts] = preparets(net,{},{},Lukcell);
net.trainFcn = 'trainbr';
net.trainParam.epochs=4000;
net = train(net,Xs,Ts,Xi,Ai);
```

В нем осуществляется подготовка данных к обучению сети, задается алгоритм обучения, в данном случае алгоритм Байесовской регуляризации, и количество эпох обучения, установлено 4000. Команда train запускает обучение нейронной сети.

Команда train в позволяет обучать различные сети за исключением глубоких, например сверхточных, нейронных сетей. Если предварительно командой net.trainFcn задан алгоритм обучения сети, а командой net.trainParam заданы параметры обучения, то наиболее общий синтаксис будет выглядеть следующим образом:

```
[net,tr] = train(net,X,T,Xi,Ai,EW)
```

Здесь как и обычно net — имя сети, X — выходы сети, T — цели сети. Следует обратить внимание на то, что команда train универсальная и позволяет обучать сети различной конфигурации. В связи с этим входы и цели сети могут иметь различный формат. Для статических задач это могут быть матрицы, для задачи динамических — это массивы ячеек. Размерность массива определяется количеством входов и выходов нейронной сети. Более подробную информацию можно найти в справочной системе Matlab. Так как мы обучаем динамическую авторегрессионную сеть, на вход должны подаваться массивы ячеек. Именно в этом формате и выдаёт необходимые исходные данные команда preparets.

В зависимости от типа сети не все исходные параметры являются обязательными, например, первоначальное задержки ввода Xi и вывода Ai необходимы только для авторегрессионные сетей, а наличие цели T требуется только в случае обучения с учителем для сетей, которые требует таких целей.

После начала процесса обучения по умолчанию появляется уже известное нам окошко (рис. 2.7), в котором мы имеем возможность наблюдать

процесс обучения, а при необходимости и графики отдельных параметров по заданному количеству эпох обучения. Обучение нейронной сети может быть достаточно длинное для динамических задач. Оно существенно зависит от размерности сети и количества заданных эпох обучения.

Обучение простой динамической авторегрессионной сети, на примере рекуррентных рядов типа Люка или Фибоначчи, является достаточно сложной задачей. Это связано в первую очередь с тем, что значение чисел в этих рядах существенно возрастает. Поэтому количество точек для обучения не может быть очень большим. При большом количестве значения в начале и в конце ряда будут отличаться на десятки порядков. Это означает, что значения в начале ряда могут оказаться в пределах погрешности обучения нейронной сети. Поэтому было принято незначительное количество точек ряда равное 30.

Автоматизированные подготовка данных для обучения и обучение в Matlab предполагают, что из множества точек будет отобрано примерно 30 % (такое значение используется по умолчанию) для проведения тестирования и проверки переобучения. Точки отбираются случайным образом. То есть могут быть отобраны 2 точки находящиеся рядом. Естественно это прервет природную закономерность ряда чисел Люка. Нейронная сеть может не заметить этой закономерности и даже наиболее вероятно ошибется. Решить эту проблему можно или специальная подготовка данные для обучения, или многократным повторением автоматизированного процесса до получения приемлемого результата. В данном случае пойдём по второму пути.

После некоторого количества экспериментов количество нейронов в скрытом слое сети была увеличено до 200, а количество задержек было принято равным 5.

В результате обучения получена открытая динамическая авторегрессионная сеть (рис. 2.22). По своей сути в таком виде она не может быть

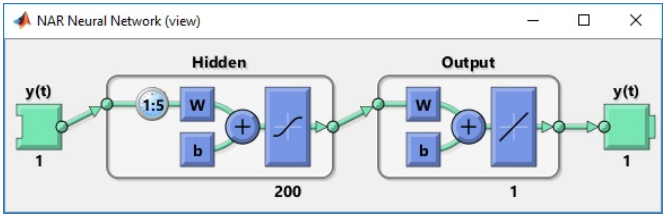


Рисунок 2.22 — Схема открытой динамической авторегрессионной сети

использована, так как на вход должны подаваться выходы сети на предыдущем шаге. Для получения прогнозных значений эту сеть необходимо

зациклить. Преобразование сети с открытой обратной связью в замкнутый цикл можно осуществить с помощью команды closeloop. В ее синтаксисе имеется следующий вариант:

```
net = closeloop(net)
[netc,xic,aic] = closeloop(net,xi,ai)
```

Здесь netc,xic,aic параметры аналогичные net,xi,ai для варианта замкнутой нейронной сети. В случае использования более простого синтаксиса может понадобиться использование команды preparets.

То есть помощью программного кода

```
[Y,Xf,Af] = net(Xs,Xi,Ai);
[netc,Xic,Aic] = closeloop(net,Xf,Af);
view(netc)
```

Получаем замкнутую в цикл динамическую авторегрессионную сеть (рис. 2.23).

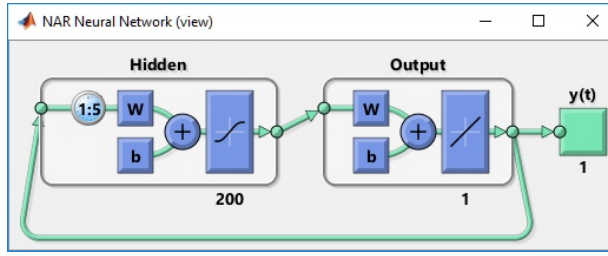


Рисунок 2.23 — Схема замкнутой динамической авторегрессионной сети

Для прогноза следующих nprog=7 точек числового ряда можем воспользоваться полученной сетью.

```
Lukcellnet = netc(cell(0,nprog),Xic,Aic)
Luknetnprog=cell2mat(Lukcellnet)
```

Сравнение прогноза нейронной сети с теоретическими данными для заданных точек в последовательности чисел Люка хорошо видно на следующем графике (рис. 2.24).

Первые три точки прогноза практически полностью совпадают и лишь на 4-й точке временного ряда появляется незначительная погрешность. В дальнейшем графики расходятся.

Простейшая динамическая авторегрессионная сеть позволила нам сделать достаточно точный прогноз на 4 временных шага. Однако числа Люка и Фибоначчи возрастают достаточно монотонно и не являются характерными динамически изменяющимися величинами.

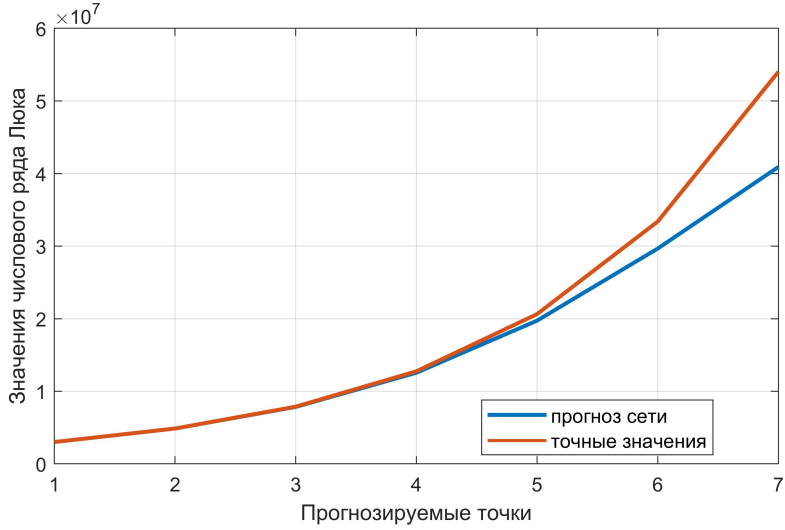


Рисунок 2.24 — Сравнение прогноза нейронной сети с теоретическим рядом

2.2.2 Прогнозирование значений рекурсивных последовательностей с помощью нелинейных авторегрессионных нейронных сетей с внешним входом (NARX)

Рассмотрим создание динамической авторегрессионной сети с внешним входом (NARX). Такие задачи могут стоять и использоваться более широко, чем рассмотренные в п. 2.2.1.

Оценка качества прогноза динамической авторегрессионной нейронной сетью возможна только в случае если известны последующие значения, то есть обучение проводится на части временного ряда, а его окончание используется для оценки качества прогноза. Авторегрессионные нейронные сети с внешним входом позволяют вводить некоторые внешние факторы, оказывающие влияние на результаты прогноза. Поиск подобных данных является исследовательской задачей, с которой наверняка часто встречаются исследователи разного рода. По сути мы имеем объект в виде чёрного ящика, у которого на вход подаются факторы, меняющиеся с течением времени.

Для возможности получение навыка в создании и обучении, анализе поведения авторегрессионной нейронной сети в качестве исходных данных зададим входы и цели меняющиеся по гармоническим законом. При этом обучение будем проводить за целое количество периодов изменения функции.

Пусть входы X меняются по закону синуса

$$X_i = \sin(i), \tag{2.4}$$

где i — номер временного шага, соответствующий углу в градусах.

Пусть цели нейронной сети являются функцией от будущего значения входа, предыдущего значения выхода и временного шага:

$$y_{i-1} = X_i\, y_{i-2} + 0{,}5\cos(7(i-1)), \tag{2.5}$$

Подготовка исходных данных для обучения нейронной сети приведена в следующем программном коде

```
clear; clc;
max=720;
i=linspace(0,max-1,max);
X=sin(deg2rad(i));
y1(1)=0;
y1(2)=0;
for ii=3:max
y1(ii-1)=X(ii)*y1(ii-2)+0.5*cos(7*deg2rad(ii-1));
end
y1(ii)=X(ii)*y1(ii-2)+0.5*cos(7*deg2rad(ii-1));
figure
plot(X);
hold on;
plot(y1,'g');
save('input_date_sin04.mat','X','y1');
figure
plot(X,y1);
```

В результате подготовки получены графики входов и целей динамической авторегрессионной сети, показанные на рисунке 2.25.

Получаем достаточно большое количество точек, соответствующих сложным образом изменяющемуся гармоническому временному ряду.

Процесс подготовки, создания и обучения нейронной сети абсолютно аналогичен описанному ранее. Поэтому приведём программный код без дополнительных пояснений.

```
X=num2cell(X);
y=num2cell(y1);
nT=2
net=narxnet(1:nT,1:nT,15);
[Xs,Xi,Ai,Ts]=preparets(net,X,{},y);
[net,tr]=train(net,Xs,Ts,Xi,Ai);
```

Рисунок 2.25 — График изменения входов и целей по временной шкале

```
view(net);
Y=net(Xs,Xi,Ai);
Y=cell2mat(Y)
netc = closeloop(net);
view(netc)
[Xsc,Xic,Aic] = preparets(netc,X,{},y);
```

15 нейронов в скрытом слое в данном случае достаточно для получения качественного результата. Среднеквадратичная ошибка процесса обучения составила $2{,}5 \cdot 10^{-7}$. Об этом свидетельствует и график ошибок полученных в результате обучения (рис. 2.26).

Полученная динамическая нейронная авторегрессионная сеть (рис. 2.27) может предсказывать поведение данного ряда во времени в зависимости от входов сети.

Сравнение исходного графика целей сети и результатов прогноза на два периода в зависимости от переменной, изменяющейся по тому же гармоническому закону (2.4) показано на рисунке 2.28.

Как видно из рисунка результаты прогноза и цели сети, использованные для обучения, практически совпадают. Отличия достаточно незначительны. Сеть выдаёт такую же гармоническую функцию, какая и была задана при обучении. Даже в случае прогноза на периоды времени многократно

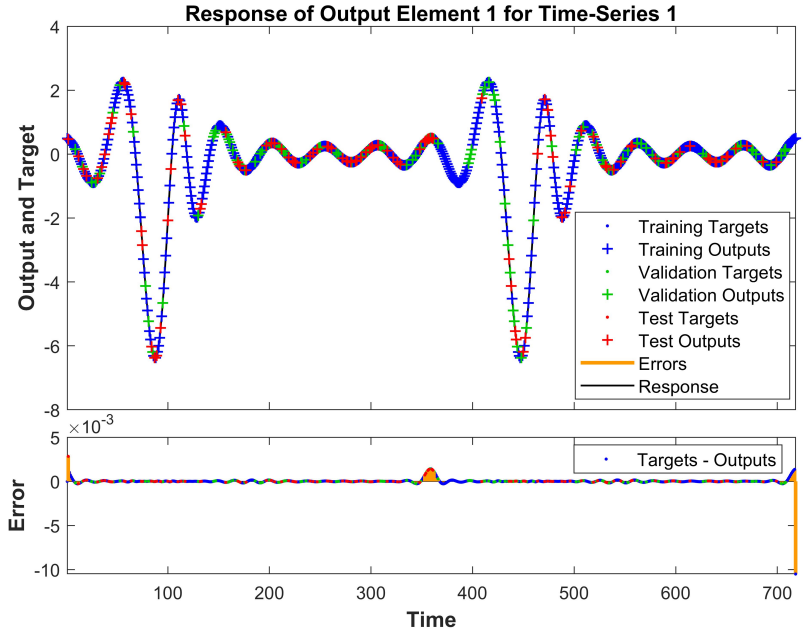


Рисунок 2.26 — График ответов сети и отклонения от целей по временной шкале

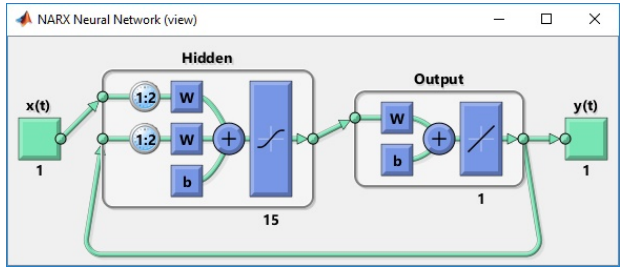


Рисунок 2.27 — Схема динамической нейронной авторегрессионной сети с внешним входом

превышающие период обучения. Следовательно при качественном обуче-

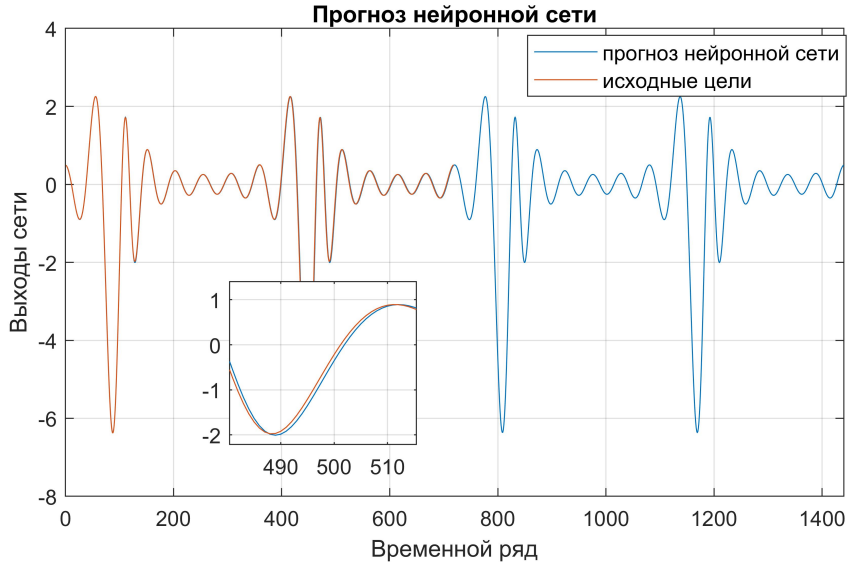


Рисунок 2.28 — Сравнение целей и выходов сети в результате прогноза на два периода

нии динамическая авторегрессионная нейронная сеть фактически повторяет функциональную зависимость между входами и выходами.

Следует отметить, что это не чисто математическое повторение, хотя и в основе искусственных нейронных сетей лежит математика. Для того чтобы разобраться в этом вопросе подадим на вход обученной сети значения отличающиеся от тех, на которых проходило обучение, но также изменяющаяся по гармоническому закону

$$X_i = \cos(i). \tag{2.6}$$

Ожидаемо, что выходы сети будут отличаться. Однако синус и косинус фактически являются одной и той же функцией, смещённой по углу, а в нашем случае по времени. Это и подтверждает результат прогнозирования с помощью нейронной сети. Как видно из рисунка 2.29 в начальный промежуток времени выходы существенно отличаются от множества, на котором сеть проходила обучение. В дальнейшем же график выравнивается и превращается в известную нам периодическую функцию. Хотя она и смещена на 90

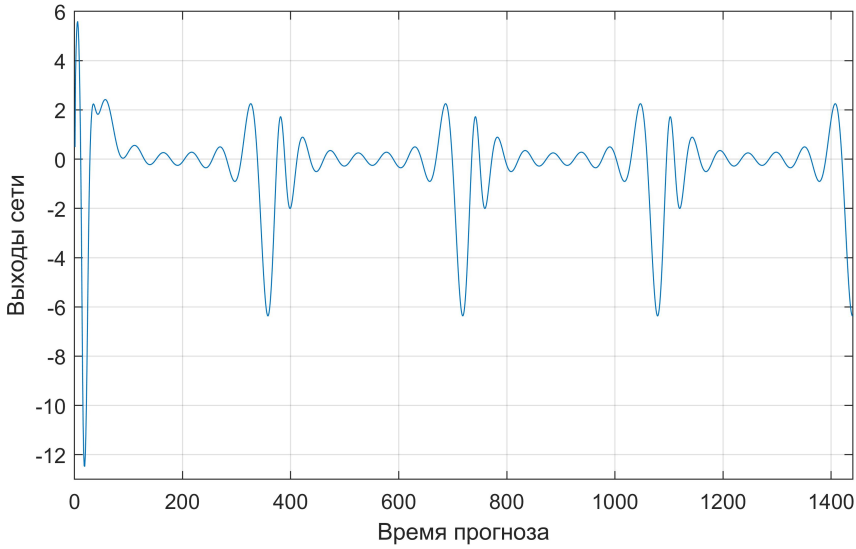


Рисунок 2.29 — Выходы сети при синусоидальном изменении значения на входе

секунд. То есть нейронная сеть распознала во входных данных синусоидальный закон их изменения и соответственно начала выдавать ожидаемый в этом случае результат. Но учитывая влияние предыдущих значений мы получили переходной процесс примерно в первые 90 секунд прогнозирования.

Можно утверждать, что нейронная сеть подкорректировала взаимосвязь между входами, выходами и временным интервалом. Фактически мы получили график, соответствующий зависимости (2.7)

$$y_{i-1} = X_i\, y_{i-2} + 0{,}5 \sin(7(i-1)), \tag{2.7}$$

что отличается от начально заданной (2.5). Это хорошо видно по графикам приведенным на рисунке 2.30. Синей пунктирной линией показан график, который был бы получен в случае, если бы сеть запомнила и реализовала те взаимосвязи, которые были заложены в данные для обучения с математической точностью. Красной сплошной линией показана зависимость, которую фактически изобразила обученная нейронная сеть.

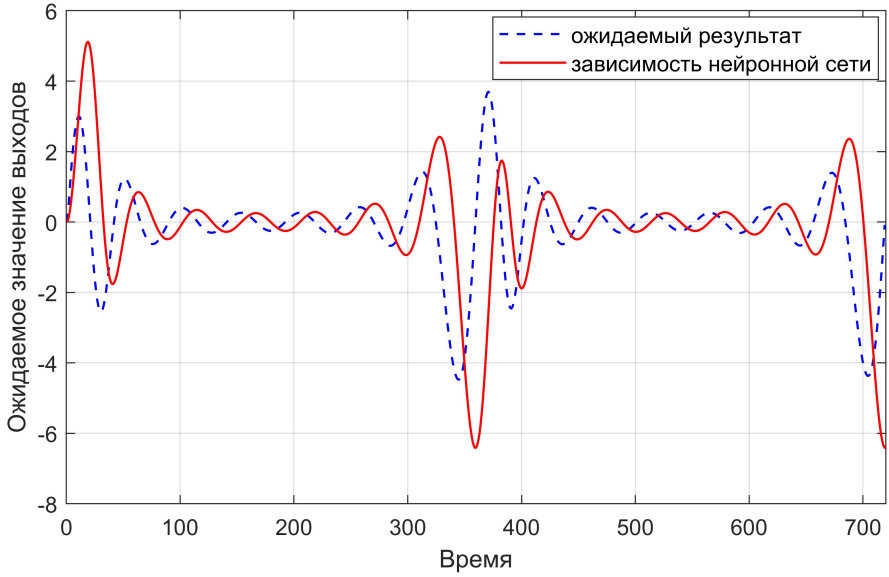


Рисунок 2.30 — Сравнение изменений, проведённых нейронной сетью

Как часто утверждается, нейронные сети обладают свойством обобщения. В данном случае мы получили аналогичное обобщение сетью. То есть нейронная сеть не заметила разницы между синусом и косинусом и обобщила эти две функции, рассматривая как одну с некоторым смещением во времени.

По характеру зависимости выдаваемой сетью можно судить и о несоответствии характера входных данных на отдельном промежутке времени. Временные изменения входного сигнала приводят к изменению выхода сети, но возвращение к начальному порядку следования входных данных, после некоторого переходного процесса, позволяет сети очень быстро перейти к начальным зависимостям (рис. 2.31). Подобные свойства позволяют при соответствующем обучении создавать системы автоматизированного регулирования на основе нейронных сетей рассмотренного типа.

Рассмотренные примеры нейронных сетей показывают, что данная технология позволяет в отдельных случаях заменить классические методы аппроксимации экспериментальных данных с целью как получения функциональных зависимостей на участках, для которых имеются результаты эксперимента, так и при необходимости получения трендов и прогнозирования. Прогнозирование может быть как на небольшие промежутки

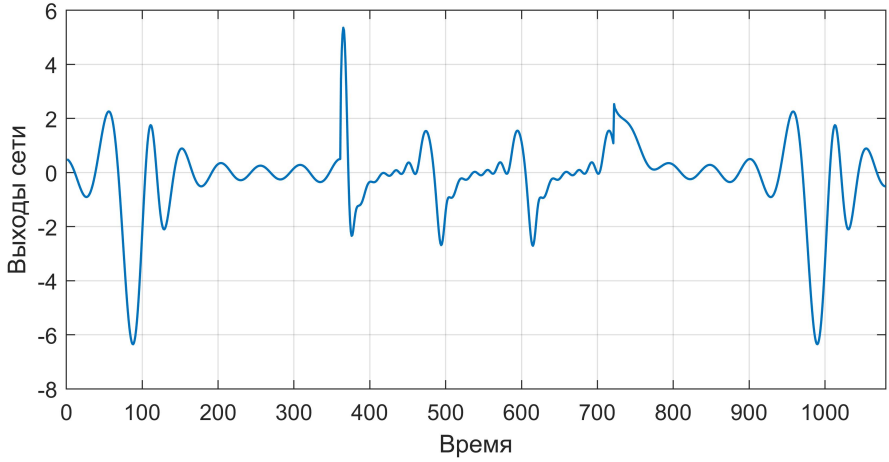


Рисунок 2.31 — Реакция сети на возмущение входного сигнала

времени, в случае если фактически нельзя иметь уверенность в какой-либо функциональной зависимости исследуемых параметров объекта, так и на значительные отрезки если подобные зависимости прослеживается. При отсутствии или сложности функциональных связей возможно прогнозирование на небольшие отрезки времени с последующим обучением в сети на новых экспериментальных данных.

Модели на основе нейронных сетей могут быть использованы в исследовательских целях или в рутинной инженерной деятельности.

СПИСОК ЛИТЕРАТУРЫ

1. ChipTuningPRO. SMS-Soft. — 12.07.2018. — URL: http://sms-software.ru.
2. MathWorks: Documentation R2018a. — 07.06.2018. — URL: https://www.mathworks.com/help/index.html (дата обр. 15.02.2017).
3. *Radionova L.*, *Chernyshev A.* Mathematical Model of the Vehicle in MATLAB Simulink // Procedia Engineering. — 2015. — Т. 129. — С. 825—831. — DOI: 10.1016/j.proeng.2015.12.114. — URL: http://www.sciencedirect.com/science/article/pii/S1877705815039983?via%3Dihub.
4. Автомобильные двигатели / В. М. Архангельский [и др.] ; под ред. М. С. Ховаха. — М. : Машиностроение, 1977. — 591 с.
5. *Аюпов В. В.* Математическое моделирование технических систем : учебное пособие. — Пермь : ИПЦ «Прокростъ», 2017. — 242 с. — ISBN 978-5-94279-337-1. — М-во с.-х. РФ, федеральное гос. бюджетное образов. учреждение высшего образования «Пермская гос. с.-х. акад. им. акад. Д.Н. Прянишникова».
6. *Большаков В. П.* Твердотельное моделирование деталей в CAD-системах: AutoCAD, КОМПАС-3D, SolidWorks, Inventor, Creo : Учебный курс (рекомендовано УМО). — М. : Питер, 2014. — 304 с. — ISBN 978-5-496-01179-2.
7. *Васильев В. В.*, *Симак Л. А.*, *Рыбникова А. М.* Математическое и компьютерное моделирование процессов и систем в среде MATLAB/SIMULINK : Учебное пособие для студентов и аспирантов. — Киев : НАН Украины, 2008. — 91 с. — ISBN 978-966-02-4389-7.
8. Двигатели внутреннего сгорания : Теория поршневых и комбинированных двигателей. / Д. Н. Вырубов [и др.] ; под ред. А. С. Орлина, М. Г. Круглова. — 4-е изд., перераб. и доп. — М. : Машиностроение, 1983. — Гл. Учебник для втузов по специальности «Двигатели внутреннего сгорания». 372 с.
9. Двигатели внутреннего сгорания. В 3 т. Т. 2. Динамика и конструирование / В. Н. Луканин [и др.] ; под ред. В. Н. Луканина ; под ред. М. Г. Шатрова. — М. : Высшая школа, 2005. — 400 с.
10. *Духанов А. В.*, *Медведева О. Н.* Имитационное моделирование сложных систем : курс лекций. — Владимир : Изд-во Владим. гос. ун-та, 2010. — 107 с. — ISBN 978-5-9984-0037-7. — Владим. гос. ун-т.
11. *Дьячков Ю. А.* Моделирование технических систем. — 2011. — 239 с.

12. *Золотых Н. Ю.* Использование пакета Matlab в научной и учебной работе. — Нижний Новгород : ННГУ, 2006. — 165 с.

13. *Колчин А. И.*, *Демидов В. П.* Расчет автомобильных и тракторных двигателей : Учебное пособие для ВУЗов. — 4-е изд., перераб. и доп. — М. : Высшая школа, 2008. — 496 с.

14. *Королев А. Л.* Компьютерное моделирование технических систем : учебное пособие для студ. высш. учеб. заведений. — Челябинск : Изд-во Челяб. гос. пед. ун-та, 2009. — 170 с. — ISBN 978-5-85716-803-5.

15. *Коткин Г. Л.*, *Черкасский В. С.* Компьютерное моделирование физических процессов с использованием MATLAB : Учеб. пособие. — Новосибирск : Новосиб. ун-т, 2001. — 173 с.

16. *Лазарев Ю. Ф.* Моделирование процессов и систем в Matlab. Учебный курс. — Киев : Издательская группа BHV, 2005. — 512 с.

17. Матлаб. Решение уравнений в частных производных / Российский химико-технологический университет им. Д.И. Менделеева. — URL: `https://studfiles.net/preview/1804443/` (дата обр. 23.01.2019).

18. *Мезенцев К. Н.* Моделирование систем : Основы системотехники и исследования систем : курс лекций : в 2 т. Ч. 1 / под ред. д-ра техн. наук, проф. А. Б. Николаева. — Москва : МАДИ, 2017. — 84 с. — URL: `http://lib.madi.ru/fel/fel1/fel18E454.pdf`.

19. *Тарасик В. П.* Математическое моделирование технических систем. — Минск : Дизайн-ПРО, 2004. — 640 с.

20. *Химченко А. В.*, *Мищенко Н. И.* Имитационное моделирование работы механизма отключения цилиндра в двигателе с кривошипно-кулисным механизмом // 8-Е Луканинские чтения. Проблемы и перспективы развития автотранспортного комплекса. Москва, 31 января 2019 г. — Московский автомобильно-дорожный государственный технический университет (МАДИ), 2019. — С. 383—396. — URL: `https://elibrary.ru/download/elibrary_37540777_61835915.pdf` (дата обр. 12.12.2019).

21. *Черных И. В.* Simulink: среда создания инженерных приложений. — М. : ДИАЛОГ-МИФИ, 2003. — 496 с.

22. *Черных И. В.* Simulink: Инструмент моделирования динамических систем. — URL: `http://matlab.exponenta.ru/simulink/book1/index.php` (дата обр. 03.03.2017).

ЭЛЕКТРОННОЕ УЧЕБНОЕ ИЗДАНИЕ

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ТЕХНИЧЕСКИХ СИСТЕМ

УЧЕБНОЕ ПОСОБИЕ

Химченко Аркадий Васильевич
Мищенко Николай Иванович

Компьютерная вёрстка в LaTeX Химченко А. В.

Подписано к изданию 16.05.2018 г. Гарнитура Computer Modern.
Объем издания 27,9 Мбайт, авт. л. 5,7.

Государственное образовательное учреждение
высшего профессионального образования
«Донецкий национальный технический университет»
Автомобильно-дорожный институт
284646, ДНР г. Горловка, ул. Кирова, 51
E-mail: hiav@adidonntu.ru

Свидетельство о государственной регистрации ДНР
серия АА03 № 029192 от 7 апреля 2016 г.