Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем
Направление подготовки / специальность 09.04.04 Программная инженерия
Направленность (профиль) / специализация Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____2024 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Разработка приложения «ПЗК манипулятора с произвольным числом звеньев»

Исполнитель
студент группы 2105-ом _____ (подпись, дата) Д.Ф. Вернер

Руководитель
доцент, канд. техн. наук _____ (подпись, дата) Т.А. Галаган

Руководитель научного
содержания программы
магистратуры
профессор, доктор техн. наук _____ (подпись, дата) И.Е. Еремин

Нормоконтроль
доцент, канд. техн. наук _____ (подпись, дата) Т.А. Галаган

Рецензент
доцент, канд. техн. наук _____ (подпись, дата) Т.В. Труфанова

Благовещенск, 2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____2024 г.

З А Д А Н И Е

К магистерской диссертации студента группы 2105-ом Вернера Д. Ф.

1. Тема магистерской диссертации: Разработка приложения «ПЗК манипулятора с произвольным числом звеньев»

(Утверждено приказом от 06.03.2024 № 632-уч)

2. Срок сдачи студентом законченной работы (проекта): 10.06.2024

3. Исходные данные к магистерской диссертации: пояснительная записка к курсовой работе, техническое задание

4. Содержание магистерской диссертации (перечень подлежащих разработке вопросов): Обзор предметной области, проектирование алгоритма работы программы, разработка и тестирование программного продукта

5. Перечень материалов приложения (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.): техническое задание, код разработанного программного продукта

6. Рецензент магистерской диссертации: Труфанова Т.В. доцент, канд. техн наук

7. Дата выдачи задания 29.01.2024

Руководитель выпускной квалификационной работы: Галаган Т. А., канд. техн. наук, доцент

(фамилия, имя, отчество, должность, уч.степень, уч.звание)

Заявление принял к исполнению _____

РЕФЕРАТ

Магистерская диссертация содержит 72 с., 35 рисунков, 4 таблицы, 50 источников, 5 приложений.

МАНИПУЛЯТОР, ЗВЕНЬЯ МАНИПУЛЯТОРА, ПРЯМАЯ ЗАДАЧА КИНЕМАТИКИ, РАЗРАБОТКА ПРИЛОЖЕНИЯ, ПОИСК РЕШЕНИЯ.

Объект исследования – процесс управления промышленными манипуляторами.

Предмет исследования – разработка приложения для автоматизации части процесса управления.

Цель работы – реализация программного обеспечения для поиска решения прямой задачи кинематики для модели манипулятора, собираемой пользователем.

В процессе исследования проводилось изучение, анализ, тестирование и сравнение различных существующих методов поиска решения, а также проверка работоспособности программы на реальных данных.

В результате исследования были реализованы следующие функции системы: функция моделирования манипулятора, функция поиска решения, функция вывода полученного решения.

Область применения: разработанное приложение может быть использовано при разработке и моделировании промышленных манипуляторов.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ДХ – Денавит-Хартенберг;

ОЗК – обратная задача кинематики;

ПЗК – прямая задача кинематики.

ВВЕДЕНИЕ

Автоматизация технологических процессов – это совокупность различных методов и средств, направленных на осуществление технологического процесса без непосредственного участия человека, либо оставляя за ним право принятия ключевых решений.

Применение автоматизации технологических процессов позволяет сократить численность обслуживающего персонала, увеличить объем выпускаемой продукции, повысить эффективность производственного процесса, улучшить качество продукции, повысить безопасность, экологичность и экономичность производства, что делает данное исследование практически значимым и актуальным.

Целью магистерской работы была разработка приложения для поиска решения прямой задачи кинематики промышленного манипулятора в аналитическом виде.

Для достижения поставленной цели необходимо было решить следующие задачи:

ознакомиться с существующим алгоритмом поиска решения прямой задачи кинематики;

рассмотреть представленные на рынке приложения для решения подобных задач;

подобрать программное обеспечение, необходимое для реализации желаемых функций разрабатываемой программы;

непосредственно разработать программу.

Новизна работы состоит в разработке приложения, обеспечивающего поиск решения прямой задачи кинематики в аналитическом виде, т.е. в виде уравнений, позволяющих определить положение рабочего органа манипулятора в зависимости от его обобщённых координат в любой момент времени без необ-

ходимости поиска информации о его положении в предыдущие моменты времени.

Результаты работы апробированы на конференциях:

XXIV региональная научно-практическая конференция «Молодежь XXI века: шаг в будущее» (г. Благовещенск, 2023 г.).

Международная научно-практическая конференция «Кооперация науки и общества – путь к модернизации и инновационному развитию» (г. Уфа, 2024 г).

По результатам конференций опубликована статья в журнале «Вестник Амурского государственного университета» (выпуск 105, серия естественные и экономические науки)

1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРЕДМЕТНОЙ ОБЛАСТИ И ОБЪЕКТА ИССЛЕДОВАНИЯ

С развитием технического прогресса всё чаще встаёт вопрос автоматизации выполнения задач производственного процесса. С автоматизированными системами управления производственными процессами можно быстрее принимать решения и реагировать на изменения рынка.

Особенно часто для автоматизации используются манипуляторы. Перед введением их в употребление необходимо проводить некоторые расчёты, в частности, рассчитывать прямую и обратную задачи кинематики. Их решение достаточно хорошо изучено, и осуществляется по известным алгоритмам.

Однако процесс поиска решения в первый раз необходимо проводить вручную. Для облегчения работы с манипуляторами может помочь программа, которая по их виду будет решать прямую задачу кинематики.

Использование манипулятора начинается с его разработки, которая, в свою очередь, начинается с этапа моделирования. Сокращение времени, потраченного на поиск успешной модели управления, позволит сократить общее время исследования, что позволит повысить количество рассмотренных моделей за тот же срок разработки.

Приложения для выполнения подобной задачи достаточно специфичны, а потому не распространены на рынке. Тем не менее, автоматизация процесса поиска решения позволит инженерам-робототехникам быстрее находить уравнения управления, а также подбирать наиболее успешные комбинации звеньев для выполнения тех или иных работ производственного процесса.

1.1 Предметная область и объект проводимого исследования

1.1.1 Степень подвижности манипулятора

Промышленный робот – предназначенный для выполнения двигательных и управляющих функций в производственном процессе манипуляционный робот, то есть автоматическое устройство, состоящее из манипулятора (манипу-

ляционной системы) и перепрограммируемого устройства управления, которое формирует управляющие воздействия, задающие требуемые движения исполнительных органов манипулятора. Применяется для перемещения предметов производства и выполнения различных технологических операций.

Рисунок 1.1 – Пример промышленного манипулятора

Манипуляционная система представляет собой совокупность связанных между собой звеньев, образующих пространственный механизм с разомкнутой кинематической цепью, причем первое звено является основанием манипулятора, а последнее соединено с рабочим органом, непосредственно взаимодействующим с объектом манипулирования. Под звеном понимается деталь или совокупность деталей, которые в процессе перемещения остаются жесткими, т. е. не изменяют размеров и формы.

Первое неподвижное звено манипуляционной системы называется стойкой, или основанием.

Соединение двух соприкасающихся звеньев, допускающее их относительное движение, называют кинематической парой или просто парой.

Отдельное звено, как абсолютно твердое тело, имеет 6 степеней свободы:

три независимых поступательных движения в направлении осей X, Y, Z и три вращательных относительно их движения, определяемых углами Эйлера γ, θ, φ. Совокупность линейных и полярных координат x, y, z, γ, θ, φ называется обобщенными координатами.

В свою очередь минимальное количество обобщенных координат, полностью определяющих положение и возможные направления движения тела, называют числом его степеней свободы, или степеней подвижности.

Несвязанное звено имеет максимальное число степеней свободы, равное шести. Каждая связь уменьшает это число. Следовательно, если таких связей L, то число степеней свободы связанного звена определяется соотношением $W = 6 - L$, где W – число степеней свободы звена, а L – число условий связи, или ограничений.

В зависимости от числа ограничений $L \in \{1, 2, 3, 4, 5\}$ различают кинематические пары первого, второго, третьего, четвертого и пятого классов. Причем пары 1 и 2-го классов, имеющие соответственно 5 и 4 степеней свободы, очень сложны, и в манипуляционных системах не используются.

Соединяя отдельные звенья рассмотренными способами, можно получить манипулятор требуемой конструкции. Структурная схема одного из таких манипуляторов показана на рис. 1.2. Каждое звено имеет свое наименование. Так, с неподвижным основанием связана колонна, с колонной связана каретка, с кареткой – рука, с рукой – кисть, с кистью – захват. Эти звенья образуют друг с другом кинематические пары 5-го класса, имеющие по одной степени подвижности, однако в целом количество степеней подвижности у манипулятора гораздо больше.

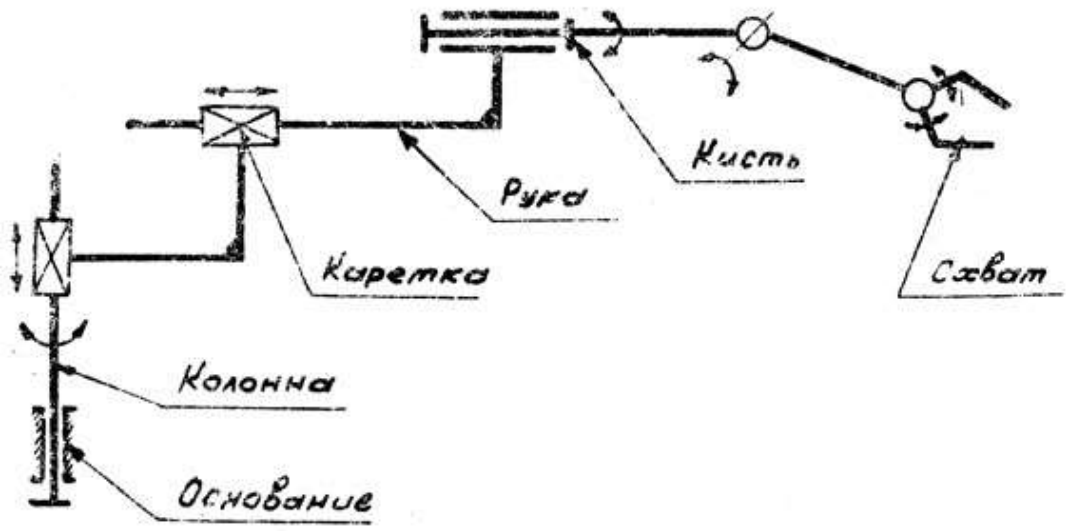

Рисунок 1.2 – Типовая схема манипулятора

Под степенями подвижности манипулятора (степенями свободы) понимают обобщенные координаты, определяющие в пространстве положения его звеньев. Число степеней подвижности манипулятора n определяется по формуле

$$n = 6k - \sum_{i=1}^{5} iP_i, \qquad (1.1)$$

где k – число подвижных звеньев;

P_i – число кинематических пар i-го класса.

Например, в рассматриваемом случае манипулятор, изображенный на рисунке 1.2, содержит 5 подвижных звеньев (исключая неподвижное основание и внутреннее подвижное звено в схвате), образующих 5 кинематических пар 5-го класса. Поэтому, согласно формуле (1.1): $n = 6 \cdot 5 - 5 \cdot 5 = 5$.

Различают следующие виды степеней подвижности: координатные – обеспечивающие выведение манипулятора в зону манипулирования; переносные (рабочие) – определяющие выведение захвата в заданные места рабочей зоны; ориентирующие (локальные) – обеспечивающие требуемые ориентации захвата в заданном месте рабочей зоны.

1.1.2 Прямая и обратная задачи кинематики

Кинематика в физике – раздел механики, изучающий математическое

описание движения идеализированных тел без рассмотрения причин движения.

Прямая задача кинематики – процесс определения параметров связанных подвижных объектов для достижения необходимой позиции, ориентации и расположения этих объектов. Говоря о манипуляторах, прямая задача кинематики состоит в определении положения рабочего органа при известных общих координатах манипулятора (рисунок 1.3)

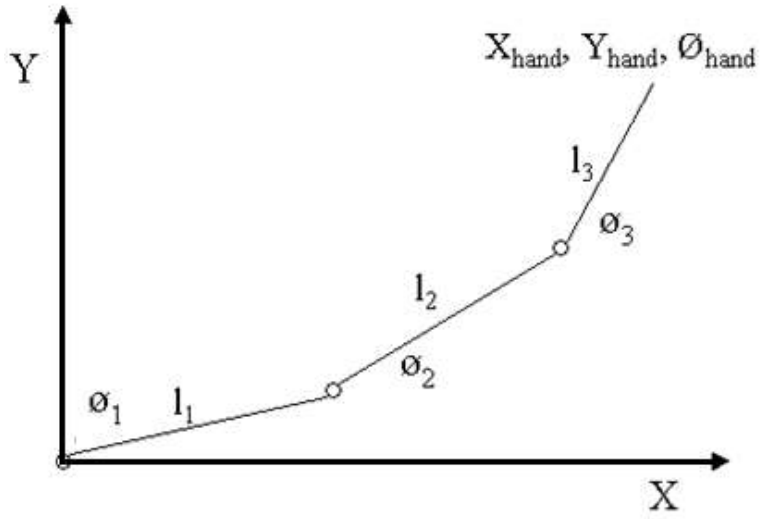

Рисунок 1.3 – Кинематическая схема манипулятора

Противоположной ей является инверсная или обратная задача кинематики. Она заключается в определении желаемых общих координат манипулятора для помещения его рабочего органа в заданное положение.

Прямая и обратная задачи кинематики отличаются по своей сложности. Прямая задача кинематики допускает только одно решение, в то время как при решении обратной задачи кинематики могут быть получены несколько значений обобщенных координат, для которых рабочий орган манипулятора оказывается в желаемой точке (рисунок 1.4). Это явление называется кинематической избыточностью и является одной из основных проблем при попытке автоматизации поиска решения обратной задачи кинематики.

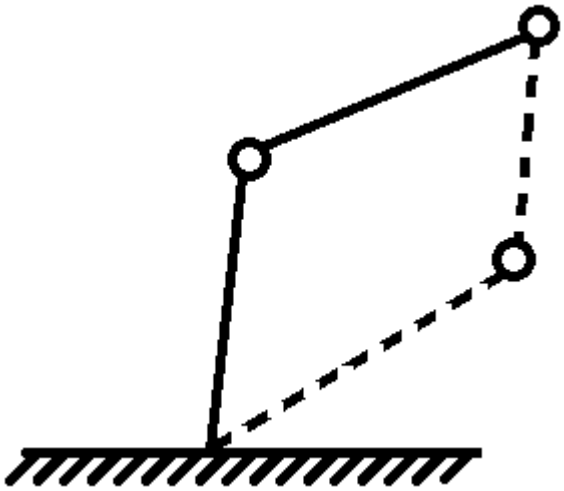

Рисунок 1.4 – Кинематическая избыточность

После получения решения прямой и обратной задачи кинематики для конкретного манипулятора их можно применить в процессе управления. Типовая модель манипулятора в программе MATLAB Simulink показана на рисунке 1.5.

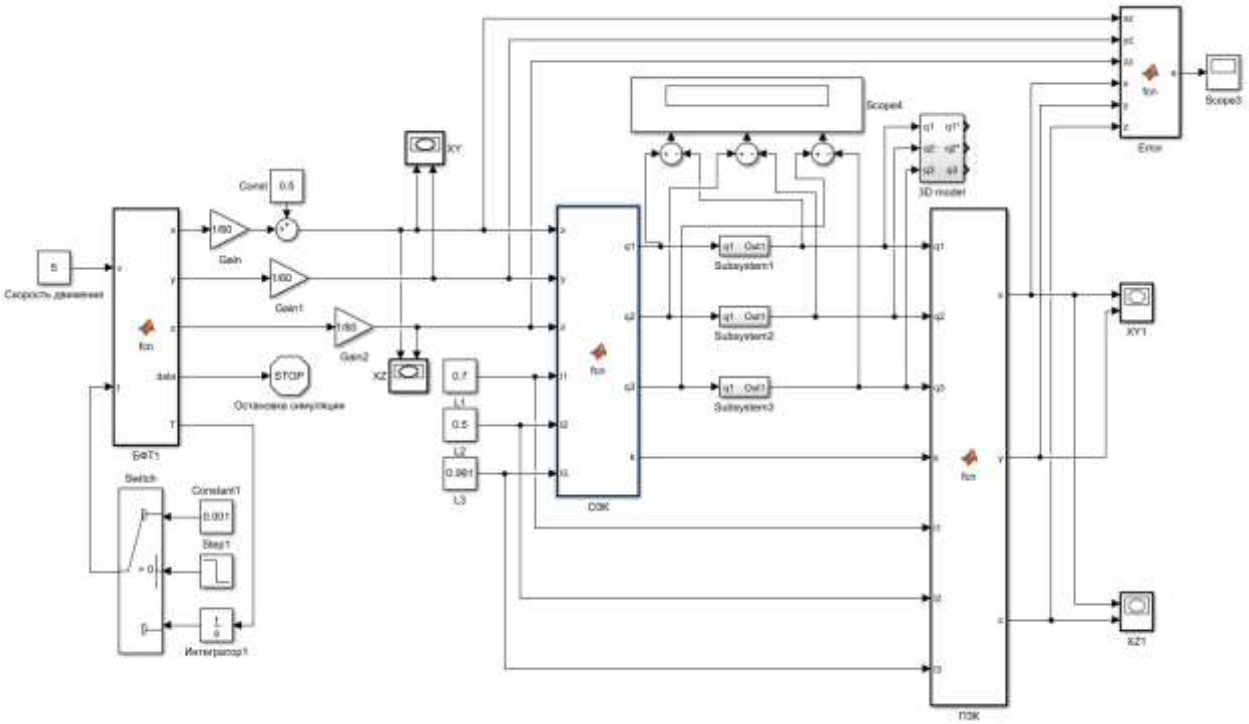

Рисунок 1.5 – Модель процесса управления манипулятором

В процессе управления желаемые координаты положения рабочего орга-

на подаются на блок решения обратной задачи кинематики манипулятора. В результате вычислений на выход блока подаются значения обобщённых координат. Эти значения поступают на блок решения прямой задачи кинематики, который, в свою очередь вычисляет положения рабочего органа. В реальной системе значения координат рабочего органа и обобщённых координат манипулятора не могут передаваться без изменений – на них влияют физические параметры в виде веса звеньев, сопротивления воздуха, погрешности в движениях двигателей каждого из звеньев. Поэтому часто промышленные манипуляторы оборудуют датчиками, которые могут считывать значения углов поворота шарниров и величину раскрытия телескопических соединений.

Поэтому важным является поиск аналитического решения прямой задачи кинематики – с использованием точного значения обобщённых координат, может быть получено точное положение рабочего органа, что позволяет минимизировать ошибку управления.

1.2 Обзор существующих методов решения рассматриваемой задачи

1.2.1 Метод Денавита-Хартенберга

Для решения ПЗК используется представление Денавита-Хартенберга, позволяющее определить положение рабочего органа манипулятора через положение его основания.

Для описания вращательных и поступательных связей между соседними звеньями Денавит и Хартенберг предложили матричный метод последовательного построения систем координат, связанных с каждым звеном кинематической цепи. Смысл представления Денавита–Хартенберга (ДХ-представление) состоит в формировании однородной матрицы преобразования, имеющей размерность 4×4 и описывающей положение системы координат каждого звена относительно системы координат предыдущего звена. Это дает возможность последовательно преобразовать координаты схвата манипулятора из системы отсчета, связанной с последним звеном, в базовую систему отсчета, являющейся инерциальной системой координат для рассматриваемой динамической системы.

Каждая система координат формируется на основе следующих трех правил:

ось Z_{i-1}направлена вдоль оси i–го сочленения;

ось X_i перпендикулярна оси Z_{i-1} и направлена от нее;

ось Y_i дополняет оси X_i, Z_i до правой декартовой системы координат.

ДХ-представление твердых звеньев зависит от четырех геометрических параметров, соответствующих каждому звену. Эти четыре параметра полностью описывают любое вращательное или поступательное движение и определяются в соответствии с рисунком 1.6 следующим образом:

θ_i – присоединенный угол, на который надо повернуть ось X_{i-1} вокруг оси Z_{i-1}, чтобы она стала сонаправлена с осью X_i (знак определяется в соответствии с правилом правой руки);

d_i – расстояние между пересечением оси Z_{i-1} с осью X_i и началом $(i-1)$-й системы координат, отсчитываемое вдоль оси Z_{i-1};

a_i – линейное смещение – расстояние между пересечением оси Z_{i-1} с осью X_i и началом i-й системы координат, отсчитываемое вдоль оси X_i, т. е. кратчайшее расстояние между осями Z_{i-1} и Z_i;

α_i – угловое смещение – угол, на который надо повернуть ось Z_{i-1} вокруг оси X_i, чтобы она стала сонаправленной с осью Z_i (знак определяется в соответствии с правилом правой руки).

Для вращательных сочленений параметры d_i, a_i и α_i являются характеристиками сочленения, постоянными для данного типа робота. В то же время θ_i является переменной величиной, изменяющейся при движении (вращении) i-го звена относительно $(i-1)$-го.

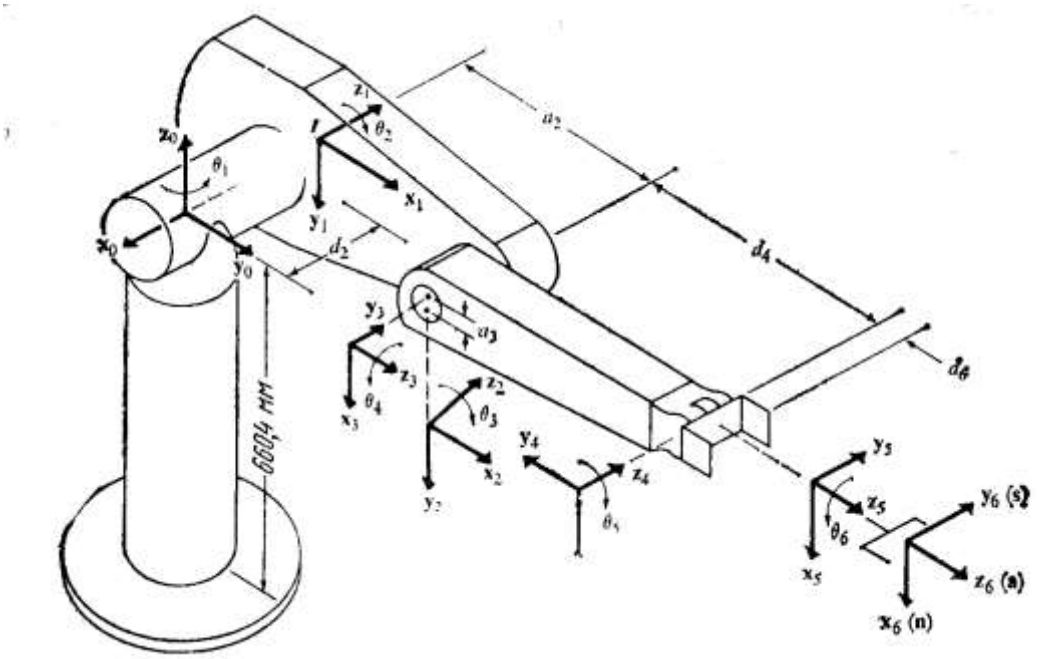

Рисунок 1.6 – Пример расположения параметров ДХ для манипулятора PUMA

1.2.2 Применение решения

В результате поиска представления ДХ получается матрица вида (1.2) размерами 4×4, которую можно условно поделить на 4 части:

$$T = \begin{bmatrix} R_{3\times3} & p_{3\times1} \\ f_{1\times3} & 1\times1 \end{bmatrix} = \begin{bmatrix} \text{Поворот} & \text{Сдвиг} \\ \text{Перспектива} & \text{Масштабирование} \end{bmatrix}, \qquad (1.2)$$

где верхняя левая подматрица размерностью 3x3 представляет матрицу поворота – матрицу преобразования трехмерного вектора положения в евклидовом пространстве, переводящую его координаты из повернутой системы координат в абсолютную систему координат; верхняя правая подматрица-столбец 3×1 представляет собой вектор положения начала координат повернутой СК относительно абсолютной; нижняя левая подматрица-строка 1×3 задает преобразование перспективы (в робототехнике состоит из нулей); нижняя правая подматрица-элемент является глобальным масштабирующим множителем (в робототехнике всегда равен 1).

Эта матрица позволяет определить положение рабочего органа манипулятора относительно его основания и для k-звенного манипулятора имеет вид:

$$ {}^{0}_{k}T = \begin{bmatrix} x_k & y_k & z_k & p_k \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (1.3) $$

где n – вектор нормали к охвату;

s – касательный вектор схвата;

a – вектор подхода схвата;

p – вектор положения схвата.

В случае плоскопараллельного движения пальцев охвата вектор нормали перпендикулярен пальцам манипулятора. Касательный вектор схвата лежит в плоскости движения пальцев схвата и указывает направление движения пальцев во время открытия и закрытия схвата. Вектор подхода направлен по нормали к ладони схвата (т. е. перпендикулярен плоскости крепления инструмента в схвате). Вектор положения направлен из начала базовой системы координат к началу системы координат схвата, которое, как правило, расположено в точке, являющейся геометрическим центром полностью сжатых пальцев (рисунок 1.7)

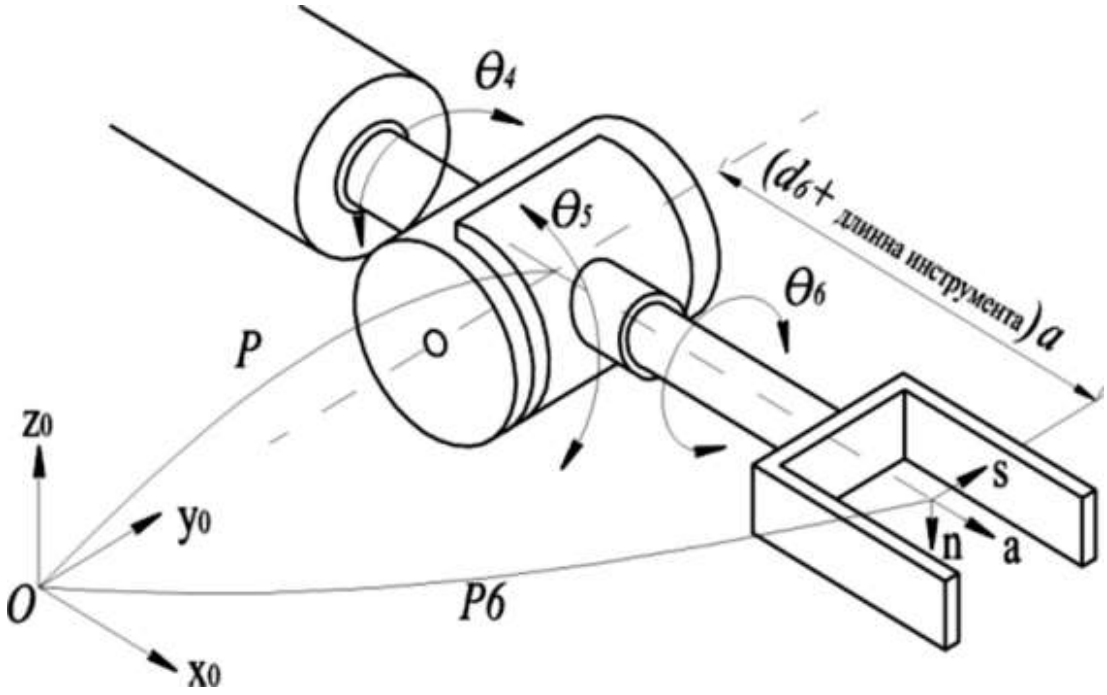

Рисунок 1.7 – Схема схвата манипулятора

Таким образом, положение рабочего органа инструмента относительно

абсолютной системы координат дается формулой (1.4)

$$^{\text{абс. СК}}_{\text{инстр.РО}}T = B \cdot {}^{0}_{k}T \cdot H, \tag{1.4}$$

где матрица B – определяет положение манипулятора в абсолютной системе координат;

H – матрица, определяющая положение закрепленного в манипуляторе инструмента в системе координат схвата.

Исходя из исследования предметной области и имеющегося метода решения, для разрабатываемого приложения были поставлены следующие функциональные требования:

Сборка манипулятора – для проведения вычислений и поиска решения необходимо сообщить программе о модели и звеньях манипулятора, что удобнее всего сделать с помощью конструктора, доступного для работы пользователю.

Наличие библиотеки компонентов – исходящее из предыдущего требование. Для удобства пользователя и ограничения возможных вариантов манипуляторов программа должна обладать встроенной библиотекой компонентов, содержащей типовые звенья манипулятора: жесткое звено, поворотные шарниры, жесткие сцепления, телескопические звенья.

Поиск решения – основной целью является поиск решения прямой задачи кинематики, что подразумевает, что программа должна иметь функционал для поиска этого решения в аналитическом виде, то есть в виде формул, связывающих координаты положения нулевого звена манипулятора и его рабочего органа с помощью его обобщённых координат.

На основании функциональных требований было составлено техническое задание, приведенное в приложении А.

2 АЛГОРИТМИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕШЕНИЯ ЗАДАЧИ

Проектирование программного продукта является ключевым этапом в жизненном цикле разработки, определяя основные принципы его построения и функционирования. Он включает в себя разработку общей архитектуры системы, определение основных компонентов и их взаимосвязей. Это позволяет создать логическую структуру продукта, которая будет легко масштабироваться, адаптироваться к изменениям и поддерживать интеграцию с другими системами. Разрабатываемый программный продукт – приложение для поиска решения ПЗК – является средой для моделирования манипулятора и поиска его решения с помощью заданного алгоритма. Специфика приложения позволяет провести анализ функций, необходимых для достижения желаемых целей.

При выборе программного обеспечения важно учитывать несколько факторов, таких как функциональность, удобство использования, совместимость с другими программами, доступность технической поддержки и цена. После оценки этих факторов и сравнения различных программных решений можно сделать обоснованный выбор сред разработки, которые наилучшим образом удовлетворит потребностям и помогут реализовать функциональные требования разрабатываемого программного обеспечения.

2.1 Обзор возможностей профильного программного обеспечения

Решение ПЗК и ОЗК при проектировании манипуляторов производится вручную и зависит от числа и вида звеньев, входящих в него. Проблеме управления посвящено много работ.

Среди них стоит отметить работы, посвященные управлению роботом-манипулятором и поиску решения с помощью нейросетевых технологий и персептронов (рисунок 2.1). Однако они делают больший акцент на поиске решения ОЗК и методах управления.

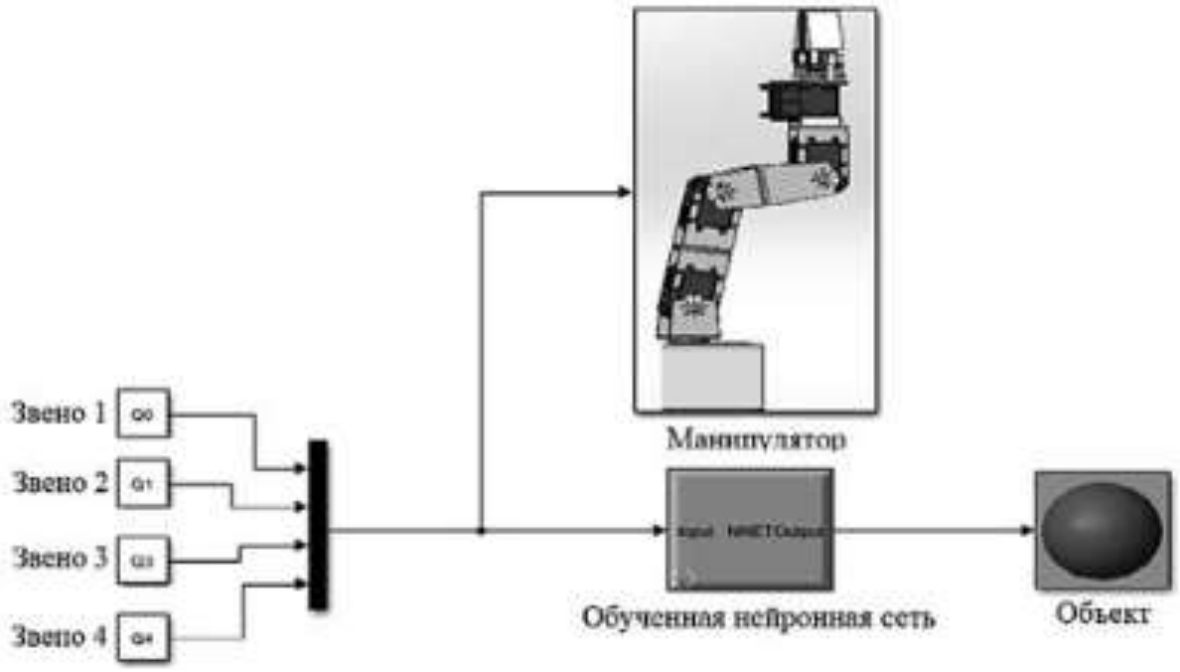

Рисунок 2.1 – Использование нейронных сетей для решения задач кинематики

Также существуют исследования, направленные на решение ПЗК для специфичных видов роботов-платформ и роботов-манипуляторов, имеющих в структуре нестандартные элементы.

Помимо этого, была проведена разработка программы для вычисления ПЗК и ОЗК манипулятора в числовом виде в реальном времени.

ПЗК и ОЗК применяются не только в робототехнике. Решение этих задач необходимо для корректного отображения связей кинематических цепей в 3D моделировании. Поэтому алгоритмы поиска решения встроены в большинство современных программ для 3D моделирования и анимации (рисунки 2.2 – 2.3).

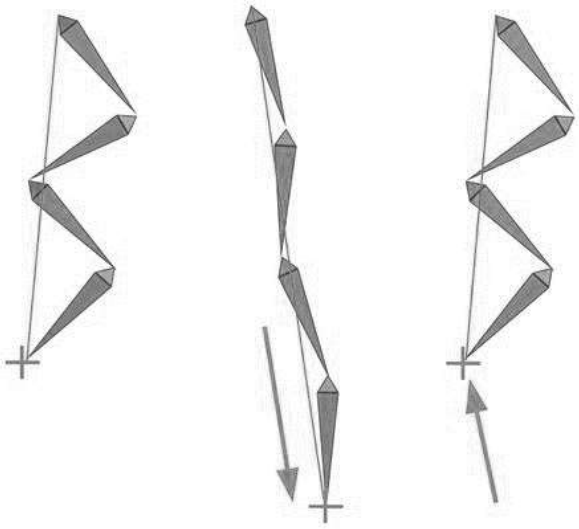

Рисунок 2.2 – Движение скелета в программе Flash с учётом задач кинематики.

Главный недостаток этих программ состоит в том, что решение ПЗК и ОЗК является лишь промежуточным этапом, инструментом, применяемым для решения задачи анимации. То есть в том случае, когда необходимо получить матричное представление искомых параметров, эти программы не могут быть использованы.

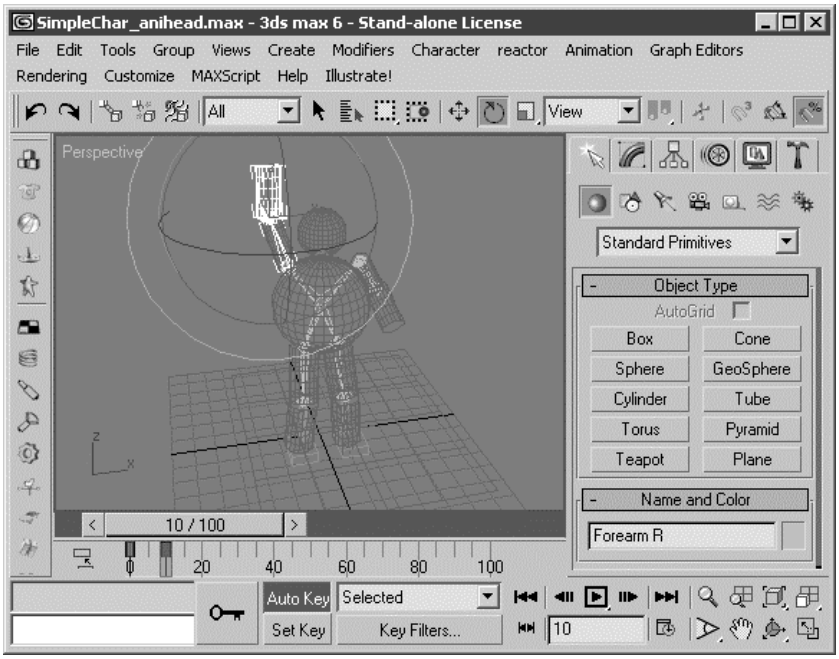

Рисунок 2.3 – Использование кинематики в программе 3ds MAX 6

Такой же проблемой обладает специальный инструмент для вычисления прямой и обратной кинематики для программы 3ds MAX «FK Solver» и «IK Solver» (рисунок 2.4), однако описанные в них алгоритмы могут быть использованы для решения задачи данной работы.

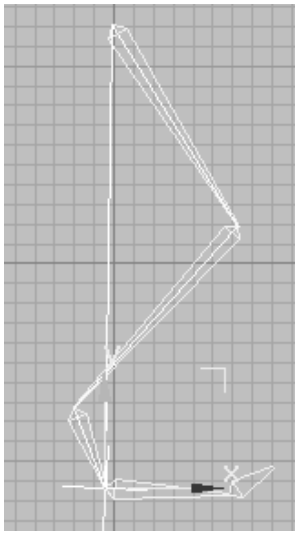

Рисунок 2.4 – Применение решателя обратной задачи кинематики

Помимо этого, в MATLAB существуют модули, позволяющие провести моделирование и анимацию модели (рисунок 2.5).

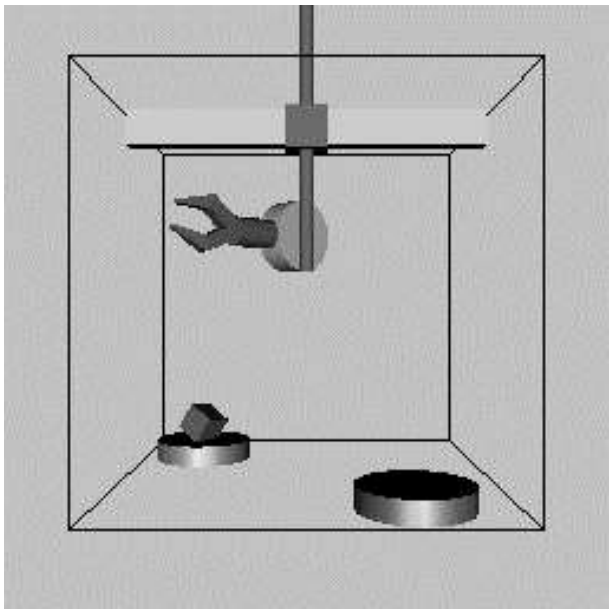

Рисунок 2.5 – Анимационный модуль MATLAB.

Этот модуль также сосредоточен на решении задачи отображения движения без явного вычисления решения для ПЗК и ОЗК.

Таким образом можно сделать вывод о том, что существуют программы, использующие в своей работе решение ПЗК, однако ни одна из них не ставит поиск его матричного вида своей целью.

На основании проведенных исследований можно составить сравнительную таблицу имеющихся решений (таблица 2.1).

Таблица 2.1 – Сравнительный анализ существующих решений

Параметр	Программа для решения ПЗК и ОЗК	Решение с применением нейросетей	Программы для 3D анимации	Предлагаемое решение
Модульная система	+	+	+	+
Численное решение	+	+	-	+
Аналитическое решение	-	-	-	+
Решение ОЗК	+	+	+	-
Вычисление числового решения в реальном времени	+	+	+	-
Визуализация 3D модели	-	-	+	+

Таким образом, реализуемая программа отличается от аналогов наглядностью представления изучаемого манипулятора, а также поиском решения в аналитической форме, что позволит использовать его при управлении манипулятором.

Можно выделить две основные задачи проекта – моделирование манипулятора и решение прямой задачи кинематики для него.

Среди решений первой задачи можно выделить следующие подходы: игры-конструкторы и специальные программы. Их сравнение приведено в таблице 2.2.

Таблица 2.2 – Сравнительный анализ существующих подходов к моделированию

Тип	Достоинства	Недостатки
Игры-конструкторы	Имеется библиотека компонентов;	Творческое направление дает избыточную свободу действий;

Продолжение таблицы 2.2

	Некоторые из игр имеют систему «точек скрепления». Соединение нескольких деталей возможно только в них;	Излишнее графическое оформление;
Специальные программы	Детали имеют простой вид; Малое занимаемое дисковое пространство;	Много требований к указанию параметров, которые не используются в решении.

Как видно из таблицы, оптимальным вариантом метода моделирования манипулятора внутри программы будет являться библиотека компонентов, содержащая готовые простые модели типовых звеньев манипулятора, что позволит упростить процесс разработки, позволяя при этом конечному пользователю собрать модель большинства применяемых в процессах автоматизации манипуляторов.

Сравнение преимуществ и недостатков способов поиска решения прямой задачи кинематики среди рассмотренных программных продуктов, а именно: нейросетевой поиск, приближенное вычисление положения с использованием информации о начальном положении и программная реализация полученного вручную решения, показаны в таблице 2.3.

Таблица 2.3 – Сравнительный анализ подходов к поиску решения

Подход	Преимущества	Недостатки
Поиск решения с помощью нейросетей	Высокая точность, относительная простота	Необходимо иметь достаточно материала для обучения, которое также занимает время
Приближенное вычисление положения	Высокая точность получаемых координат	Малая пригодность для применения в реальных условиях
Программная реализация ручного решения	Простота реализации	Необходимость начинать работу с нуля при внесении изменений в модель манипулятора

На основе таблицы 2.3 можно определить оптимальный вариант подхода к поиску решения для разрабатываемого программного продукта:

Использование модульной библиотеки позволит оперативно искать решение прямой задачи кинематики для проектируемого манипулятора и так же оперативно вносить изменения в его структуру, получая при этом новое реше-

ние в кратчайшие сроки;

Поиск решения в аналитическом виде позволит избежать накапливаемых во время работы манипулятора ошибок и вычислять положение рабочего органа даже в после восстановления работы в случае критического сбоя;

Рассмотрение взаимного влияния подвижных шарниров друг на друга позволяет сократить необходимый объём обучающего материала в виде готовых примеров, а также даёт возможность вручную проверить все варианты и аспекты их взаимодействия.

Рассмотренные в данном разделе подходы и методы решения исследуемой проблемы, а также их преимущества и недостатки позволили определить основные шаги алгоритма разработки и функционирования результирующего программного продукта, а также сузить и конкретизировать требования к его итоговому функционалу.

Функциональная декомпозиция приложения представлена на рисунке 2.6.

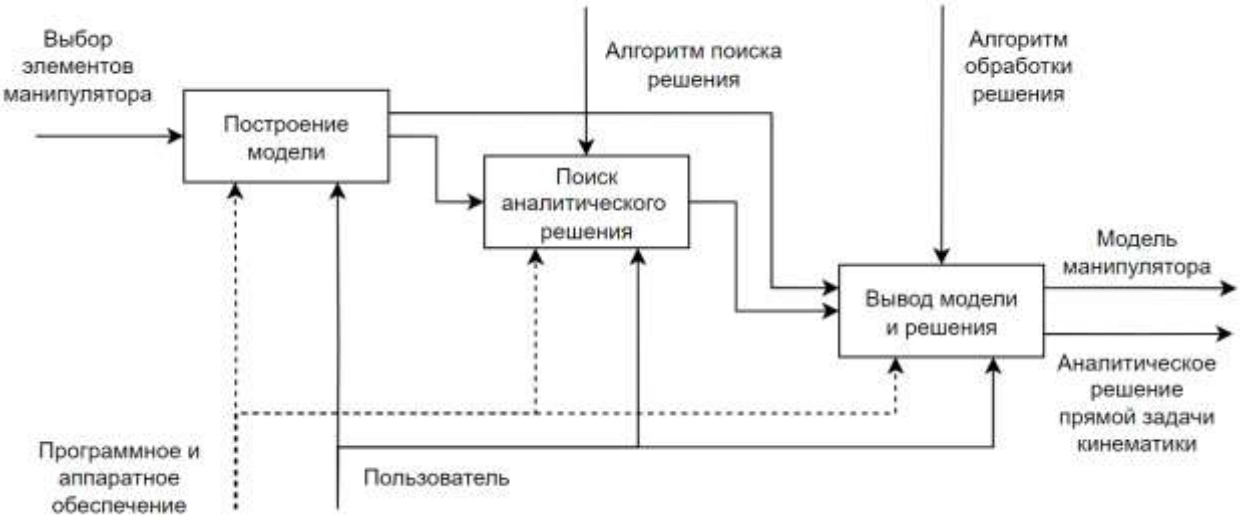

Рисунок 2.6 – Функциональная декомпозиция приложения

2.2 Предлагаемый алгоритм компьютеризированного решения задачи

Для решения ПЗК предлагается использовать алгоритм, основанный на представлении Денавита-Хартенберга, позволяющий определить положение

рабочего органа манипулятора через положение его основания.

Для описания вращательных и поступательных связей между соседними звеньями используется матричный метод последовательного построения систем координат, связанных с каждым звеном кинематической цепи. Смысл представления Денавита–Хартенберга состоит в формировании однородной матрицы преобразования, описывающей положение системы координат каждого звена относительно системы координат предыдущего звена

$$A_{i-1}^{i} = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.1)$$

где i – номер системы координат, а параметры определяются следующим образом:

θ_i – присоединенный угол, на который надо повернуть ось X_{i-1} вокруг оси Z_{i-1}, чтобы она стала сонаправлена с осью X_i (знак определяется в соответствии с правилом правой руки);

d_i – расстояние между пересечением оси Z_{i-1} с осью X_i и началом $(i-1)$-й системы координат, отсчитываемое вдоль осиZ_{i-1};

a_i – линейное смещение – расстояние между пересечением оси Z_{i-1} с осью X_i и началом i-й системы координат, отсчитываемое вдоль оси X_i, т. е. кратчайшее расстояние между осями Z_{i-1} и Z_i;

α_i – угловое смещение – угол, на который надо повернуть ось Z_{i-1} вокруг оси X_i, чтобы она стала сонаправленной с осью Z_i (знак определяется в соответствии с правилом правой руки).

Углы поворота шарниров, а также величины раскрытия телескопических соединений называются обобщёнными координатами, и обозначаются q_i.

Это дает возможность последовательно преобразовать координаты схвата манипулятора из системы отсчета, связанной с последним звеном, в базовую систему отсчета, являющейся инерциальной системой координат для рассматри-

ваемой динамической системы.

Каждая система координат формируется на основе следующих трех правил:

ось Z_{i-1}направлена вдоль оси i–го сочленения;

ось X_i перпендикулярна оси Z_{i-1} и направлена от нее;

ось Yi дополняет оси X_i, Z_i до правой декартовой системы координат.

Оси координат и обобщенные координаты манипулятора в соответствии с указанными правилами расставлены на рисунке 2.7.

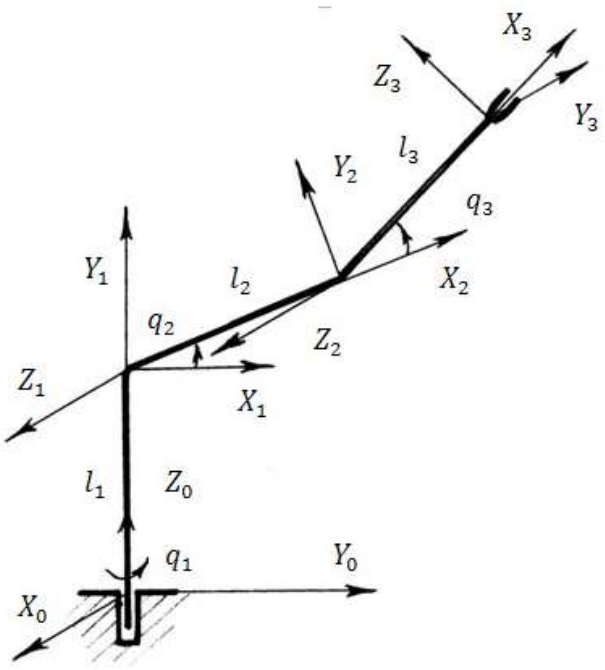

Рисунок 2.7 – Пример расположения параметров ДХ для манипулятора

Для вращательных сочленений параметры d_i, a_i и α_i являются характеристиками сочленения, постоянными для данного типа робота. В то же время θ_i является переменной величиной, изменяющейся при движении (вращении) i-го звена относительно $(i-1)$-го.

В большинстве применяемых манипуляторов используются три основных вида шарнира. Рассмотрев все их комбинации (рисунок 2.8) можно составить для них матрицы преобразования, которые затем возможно применять при вычислении решения в программе.

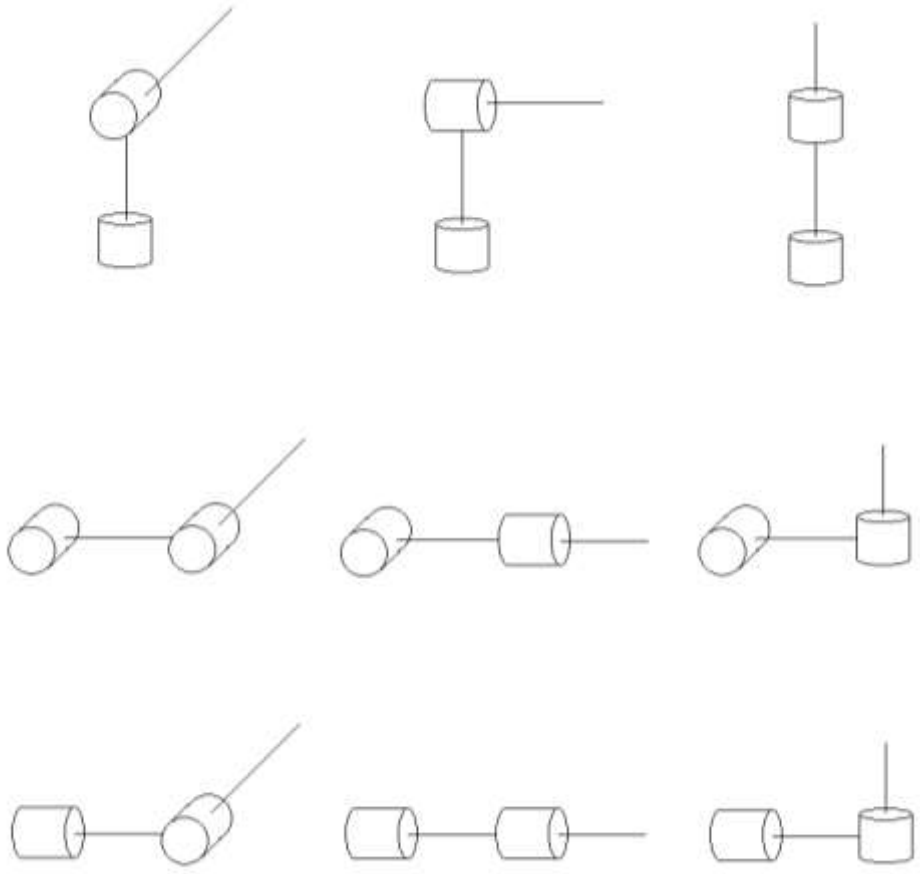

Рисунок 2.8 – Возможные комбинации шарниров

Решением прямой задачи кинематики будет являться матрица A_0^n, вычисляемая по формуле (2.1).

$$A_0^n = A_0^1 \cdot A_1^2 \cdot \ldots \cdot A_{n-1}^n, \tag{2.1}$$

где n – количество звеньев манипулятора.

Для программной реализации данного алгоритма будет применяться объектно-ориентированный подход. Для хранения информации об обобщённых координатах, длинах и типах звеньев будет использоваться класс Links.

Для хранения матриц преобразования будет использован класс TFMatrix, в котором будут описаны конструкторы для задания матриц исходя из геометрических параметров звена, а также методы, определяющие правила их пере-

множения, и совершающие форматирование имеющихся в матрице данных.

После формирования пользователем модели манипулятора, информация о звеньях будет собираться в один массив.

При нажатии кнопки вычисления, для каждого звена будет вызываться конструктор, параметры которого будут определяться комбинациями шарниров, рассмотренными ранее. После формирования и форматирования всех матриц преобразования они будут умножены для получения финального результата, который, в свою очередь, будет выводиться на экран и в текстовый файл.

2.3 Характеристика выбранного программно-технического обеспечения

Для разработки программы были выбраны следующие программные средства.

Для проверки работы алгоритма была выбрана среда разработки Visual Studio, язык разработки – C#.

C Sharp – это объектно-ориентированный язык программирования. Изначально он рассматривался как средство создания утилит для платформ Microsoft .NET Framework и .NET Core. Изначально этот язык программирования был придуман компанией Microsoft для собственных целей и служб. Он предусматривает следующие преимущества:

строгую типизацию;

сохранение концепций объектно-ориентированного программирования;

функциональность;

стабильную работу через Visual Studio;

компактный и легко читаемый код.

C# позволяет легко обрабатывать исключения и имеет встроенный сборщик мусора. Он создан таким образом, чтобы разработчикам было удобно писать и читать код.

Платформа пользовательского интерфейса для создания классических приложений для Windows – Windows Forms позволяет быстро собирать и вносить изменения в разработанный макет пользовательского интерфейса (рисунок

2.9), а встроенные средства разработки могут быть использованы для программирования событий, создания графики, вывода готовых решений не только на экран, но и в отдельный файл.

Рисунок 2.9 – Макет интерфейса в программе Visual Studio

Древовидная структура проекта позволяет с удобством отслеживать имеющиеся формы, файлы классов и библиотеки ресурсов, использующихся в программе (рисунок 2.10).

Рисунок 2.10 – Обозреватель решения и библиотека ресурсов

В Windows Forms форма – это визуальная поверхность, на которой выводится информация для пользователя. Обычно приложение Windows Forms строится путем добавления элементов управления в формы и создания кода для реагирования на действия пользователя, такие как щелчки мыши или нажатия

клавиш. Элемент управления – это отдельный элемент пользовательского интерфейса, предназначенный для отображения или ввода данных.

При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события с кодом и обрабатывает события при их возникновении.

В Windows Forms предусмотрено множество элементов управления, которые можно добавлять в формы. Например, элементы управления могут отображать текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы.

Ещё одним преимуществом разработки приложения в Visual Studio является поддержка интеграции со средой разработки Unity, которая также использует язык разработки C# и позволяет перевести процесс моделирования манипулятора в 3D пространство.

Для создания 3D моделей модулей – примитивных звеньев и шарниров, будет использована программа для создания трёхмерной компьютерной графики “Blender” (Рисунок 2.11).

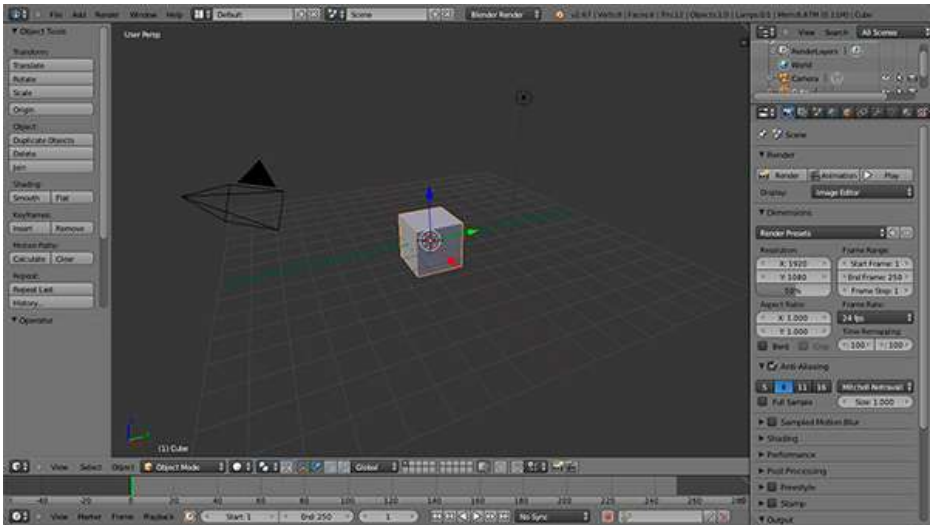

Рисунок 2.11 – Программа для 3D-моделирования Blender

Эта программа обладает своими преимуществами и недостатками:

Blender является бесплатным продуктом с открытым исходным кодом, что означает простоту использования и модификации;

это мощный инструмент, который можно использовать для решения множества задач. Он имеет широкий спектр функций и постоянно пополняется новыми;

Blender может быть использован для проектов самых разных масштабов – профессионалами в кино, телевидении и игровой индустрии, а также любителями и студентами;

Blender имеет большое и активное сообщество. Существует множество интернет-форумов и сообществ, на которых можно получить помощь других пользователей.

Unity – это среда для разработки, в которой объединены различные программные средства, используемые при создании ПО – текстовый редактор, компилятор, отладчик и т. д. Именно эта среда была выбрана для разработки приложения.

Движок Unity3D использует компонентно-ориентированный подход, в рамках которого разработчик создает объекты и добавляет к ним различные компоненты.

Unity поддерживает импорт моделей из всех популярных 3D редакторов путём конвертации: Max, Maya, Blender, Cinema4D, Modo, Lightwave и Cheetah3D, например, .MAX, .MB, .MA и др.

У этого процесса есть свои преимущества и недостатки:

Быстрый процесс повторного импорта обновленных моделей;

Простота самого процесса импорта;

На машинах, задействованных в работе над Unity проектом, должны быть установлены лицензионные копии программного обеспечения для моделирования;

Файлы, содержащие ненужные данные могут стать неоправданно большими;

Большие файлы могут замедлить процесс автосохранения;

Импортируемые файлы проходят меньше проверок, из-за чего становится труднее устранить ошибки.

Благодаря удобному Drag & Drop интерфейсу и функциональному графическому редактору, движок позволяет рисовать карты и расставлять объекты в реальном времени и сразу же тестировать получившийся результат.

Второе преимущество движка – наличие огромной библиотеки ассетов и плагинов, с помощью которых можно значительно ускорить процесс разработки.

Третья сильная сторона Unity 3D – поддержка огромного количества платформ, технологий, API. Созданные на движке приложения можно легко портировать между ОС Windows, Linux, OS X, Android, iOS, на консоли семейств PlayStation, Xbox, Nintendo, на VR- и AR-устройства.

Unity поддерживает DirectX и OpenGL, работает со всеми современными эффектами рендеринга, включая новейшую технологию трассировки лучей в реальном времени.

Из недостатков движка Unity можно выделить медлительность. Создание масштабных, сложных сцен с множеством компонентов может негативно повлиять на производительность, в результате чего разработчикам придется потратить дополнительное время и ресурсы на оптимизацию, а возможно – и удаление некоторых элементов из проекта.

Кроме того, приложения, созданные на Unity, довольно тяжеловесны: даже самая простая пиксельная игра может занимать несколько сотен мегабайт на ПК. Для жесткого диска компьютеров это небольшой объем, но, если проект разрабатывается и для мобильных платформ, оптимизация его размера станет дополнительной задачей в процессе разработки.

Схема взаимодействия используемых программных продуктов показана на рисунке 2.12.

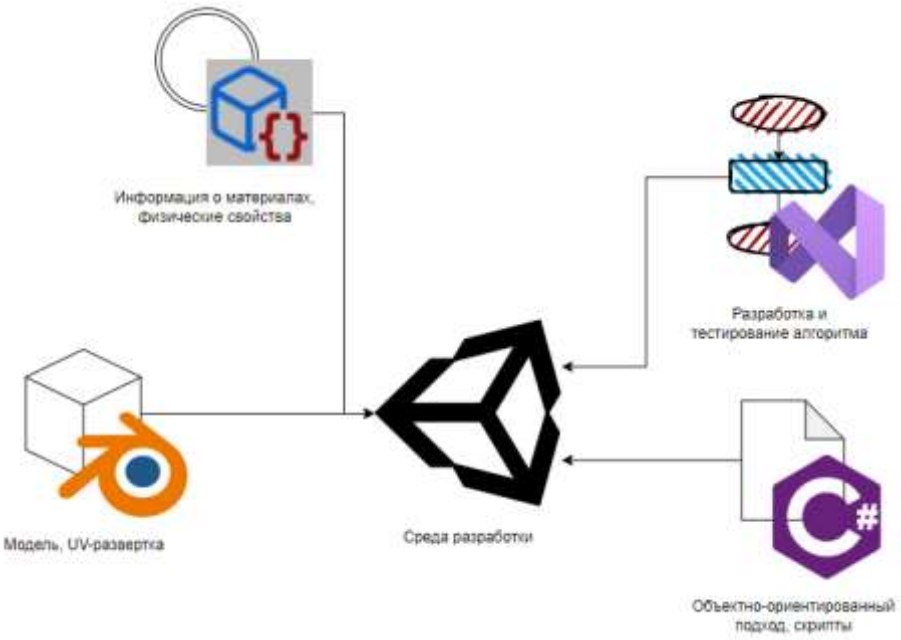

Рисунок 2.12 – Используемое программное обеспечение

Программное обеспечение, показанное на рисунке 2.12 выполняет следующие функции:

Blender используется для создания 3D модели звена, создания его UV развертки, нанесения на неё текстур, а также преобразования файла модели из стандартного разрешения .blend в поддерживаемый средой разработки Unity формат .fbx;

Среда разработки Microsoft Visual Studio позволяет создать простое приложение Windows Forms для тестирования работы алгоритма, а также может быть использовано для редактирования скриптов финальной версии приложения, благодаря поддержке языка программирования C# и встроенной функции подключения и интеграции со средой Unity;

Платформа Unity Hub с движком Unity применяется для разработки итогового программного продукта. Функции создания и назначения материалов для 3D объектов, импортированных из Blender, позволяют придать им желаемый внешний вид. Наличие объектов типа Scriptable Object дает возможность хранить наборы переменных для их применения при создании объектов сцены.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРЕДЛАГАЕМОГО АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ

Данный раздел содержит общие сведения о задаче, которую необходимо решить, и выбранном алгоритме для ее решения. Также в нем описаны шаги, предпринятые для компьютерной реализации этого алгоритма.

3.1 Основные этапы практической разработки программного продукта

3.1.1 Апробация предлагаемого метода компьютерной реализации алгоритма

Процесс решения задачи был условно разделен на три этапа:

Моделирование манипулятора с использованием библиотеки компонентов, содержащей ограниченное количество видов звеньев, в которые входят поворотные и телескопические шарниры, а также жесткие звенья;

Непосредственный поиск решения прямой задачи кинематики, включающий в себя определение типа шарниров и вычисление матриц соответствующих матриц преобразования с последующим их перемножением для получения итоговой матрицы;

Вывод решения в удобном для пользователя виде.

Перед разработкой полной версии приложения для более быстрой проверки работоспособности алгоритма решения задачи и устранения критических ошибок была проведена его апробация в упрощенном виде без использования сред разработки Unity и Blender. 3D моделей.

Блок схема алгоритма решения задачи представлена на рисунке 3.1.

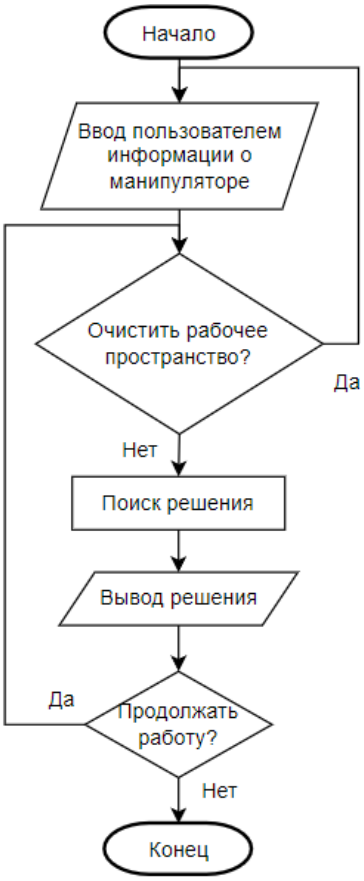

Рисунок 3.1 – Блок-схема алгоритма работы программы

Для компьютерной реализации была выбрана среда Visual Studio, язык программирования C#. С помощью платформы Windows Forms, для работы с программой был разработан учитывающий предложенный алгоритм пользовательский интерфейс, представленный на рисунке 3.2.

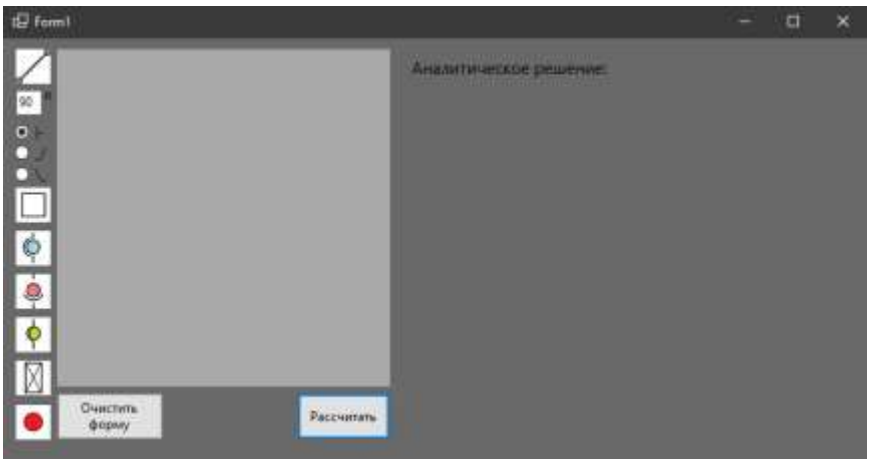

Рисунок 3.2 – Стартовый вид рабочей формы

Условно форма поделена на три области:

слева находится панель элементов, которая содержит доступные для использования при моделировании звенья. Их количество ограничено для возможности применения алгоритма к рассмотренным ранее сочетаниям звеньев;

центральная область является рабочей, выбранные звенья появляются здесь. Под рабочей областью располагаются кнопки для поиска решения и очистки формы;

справа находится область для вывода решения. В некоторых случаях решение может занимать достаточно много места, поэтому помимо области вывода запись решения также осуществляется в файл.

3.1.2 Моделирование манипулятора

Для хранения информации об используемых в манипуляторе элементов был создан класс Link, хранящий информацию о типе очередного звена, а также его обобщённую координату.

В случае, когда объект класса используется для хранения информации о жестком звене в качестве обобщенной координаты используется его длина.

В ходе работы ссылки на все объекты класса хранятся в общем массиве Links.

Нажатие на элемент панели инструментов создает очередной объект класса и задает его общую координату в зависимости от вида звена.

Элементы, размещённые на панели инструментов, представляют собой:

жёсткое звено, а также элемент для ввода угла, под которым это звено будет расположено на форме;

жёсткое соединение, которое используется для соединения двух жестких звеньев без возможности изменения угла между ними;

три поворотных шарнира в плоскостях YZ, XY и XZ;

телескопическое звено, которое не меняет угол поворота, но меняет длину.

представленные звенья являются основными и позволяют собрать модель большинства реально существующих манипуляторов.

3.1.3 Поиск и вывод решения

После создания модели манипулятора и нажатия на кнопку «Вычислить» запускается процесс поиска решения прямой задачи кинематики для собранной модели.

Для определения матриц преобразования был создан класс TFMatrix. Он содержит в себе конструктор для создания объекта-матрицы преобразования с заданными параметрами. Помимо этого, в нем содержатся правила для умножения матриц, а также для преобразования значений матрицы в пригодный для вывода вид.

Из-за потенциально большого размера искомых формул в качестве значений матриц преобразования используется текст.

Начиная со второго по порядку звена для каждого из них вызывается функция поиска матрицы преобразования на основе формул и значений, полученных при анализе возможных пар звеньев.

Подставленные значения обобщенных координат после подстановки в формулы должны быть преобразованы, так как конструктор может оставлять ошибки в виде повторяющихся знаков математических операций.

После получения матриц преобразования для всех подвижных звеньев происходит их перемножение, и полученный результат выводится на экран (рисунок 3.3).

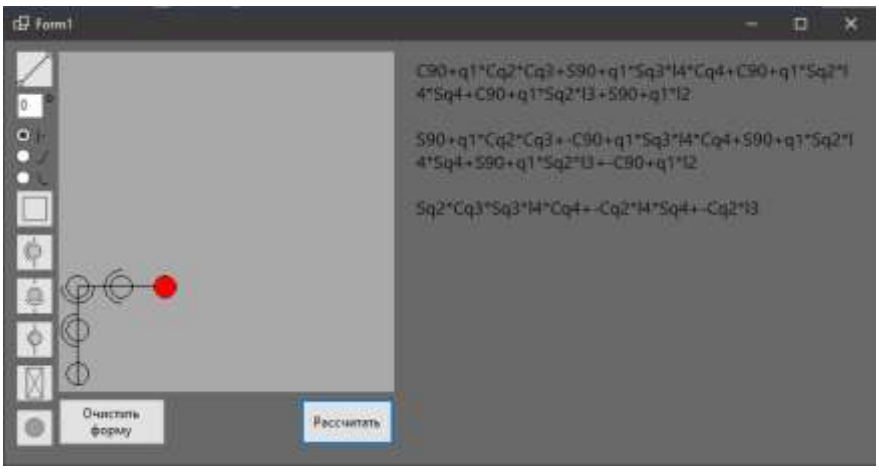

Рисунок 3.3 – Пример вывода решения на экран

Помимо этого, полученное решение выводится в отдельный текстовый файл. Полученные формулы могут быть при необходимости преобразованы по тригонометрическим формулам и применены в процессе управления.

3.2 Реализация алгоритма в среде Unity

После проверки работы алгоритма в среде Microsoft Visual Studio было до конца сформировано видение проекта.

Для визуализации взаимодействия объектов системы используются диаграммы последовательности. Они показывает временную последовательность обмена сообщениями и данными между объектами в процессе их работы. На основе работы с прототипом приложения была составлена диаграмма последовательностей, показанная на рисунке 3.4.

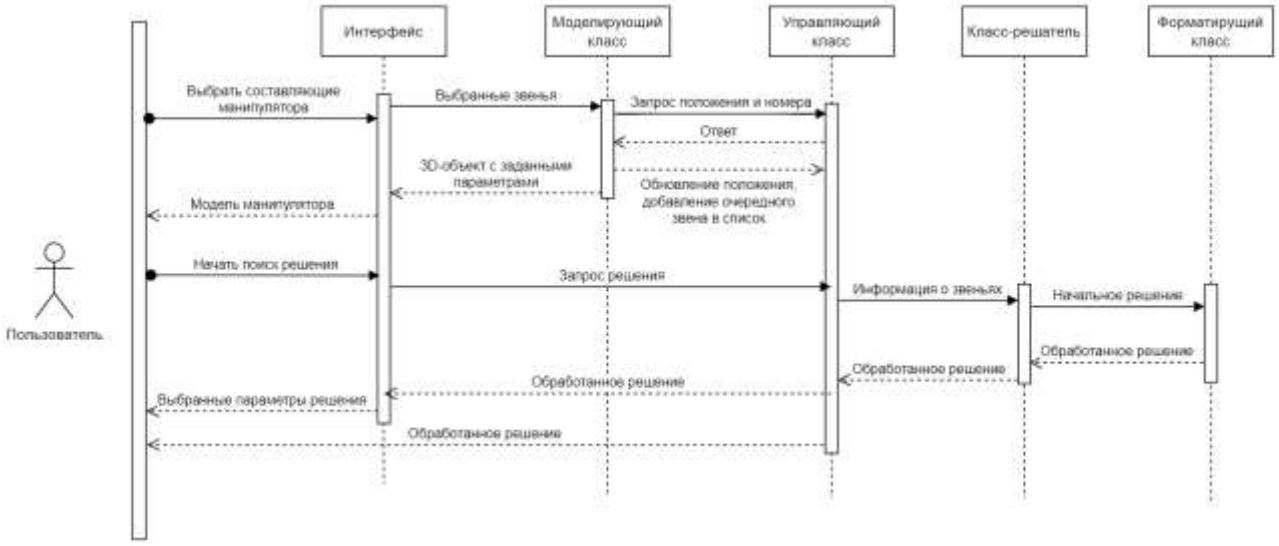

Рисунок 3.4 – Диаграмма последовательностей

На данной схеме виден процесс работы пользователя с приложением:

пользователь посылает с помощью графического интерфейса информацию о входящих в состав манипулятора звеньев. Программа фиксирует эти данные в общем списке звеньев, что необходимо для правильного размещения и последующего вычисления решения;

при добавлении очередного звена программа размещает его 3D-модель в рабочей области, а также обновляет список звеньев;

по завершении сборки модели манипулятора пользователь вызывает процедуру поиска решения с помощью кнопки на графическом интерфейсе;

класс для поиска решения получает информацию о звеньях и на её основе проводит поиск решения согласно заложенному в программе алгоритму;

после получения итогового решения программа сохраняет его в файле формата .txt, а также выводит формулы для поиска координат рабочего органа на экран.

Платформа Unity объектно-ориентирована, поэтому все операции и переменные хранятся внутри классов. Диаграмма классов, описывающая их взаимодействие внутри приложения, показана на рисунке 3.5.

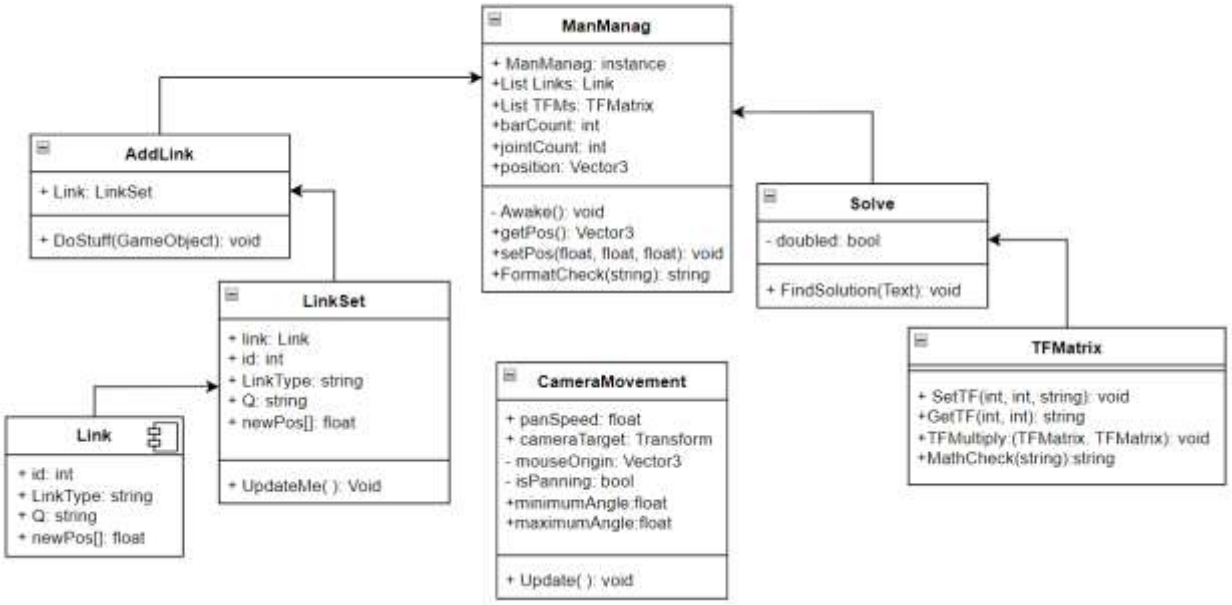

Рисунок 3.5 – Диаграмма классов

На представленной диаграмме классы Link, LinkSet, AddLink отвечают за хранение и присвоение параметров звеньям манипулятора. В них находится информация о типе звена, его длине или обобщённой координате, а также смещение, которое определяет положение следующего звена манипулятора.

Код класса AddLink приведен в приложении Б.

Классы TFMatrix и Solve осуществляют хранение информации о матрицах поворота, а также методы вычисления итоговой матрицы и приведения найден-

ного решения к форматированному виду.

Коды класса Solve приведен в приложении В.

Класс CameraMovement – технический и отвечает за движение камеры в рабочем пространстве. Его код приведен в приложении Г.

Класс ManManag – статический, в нем хранятся глобальные переменные, такие как координаты положения очередного звена, список звеньев и матриц преобразования. Код этого класса представлен в приложении Д.

С учётом составленной диаграммы классов была начата разработка на платформе Unity (рисунок 3.6).

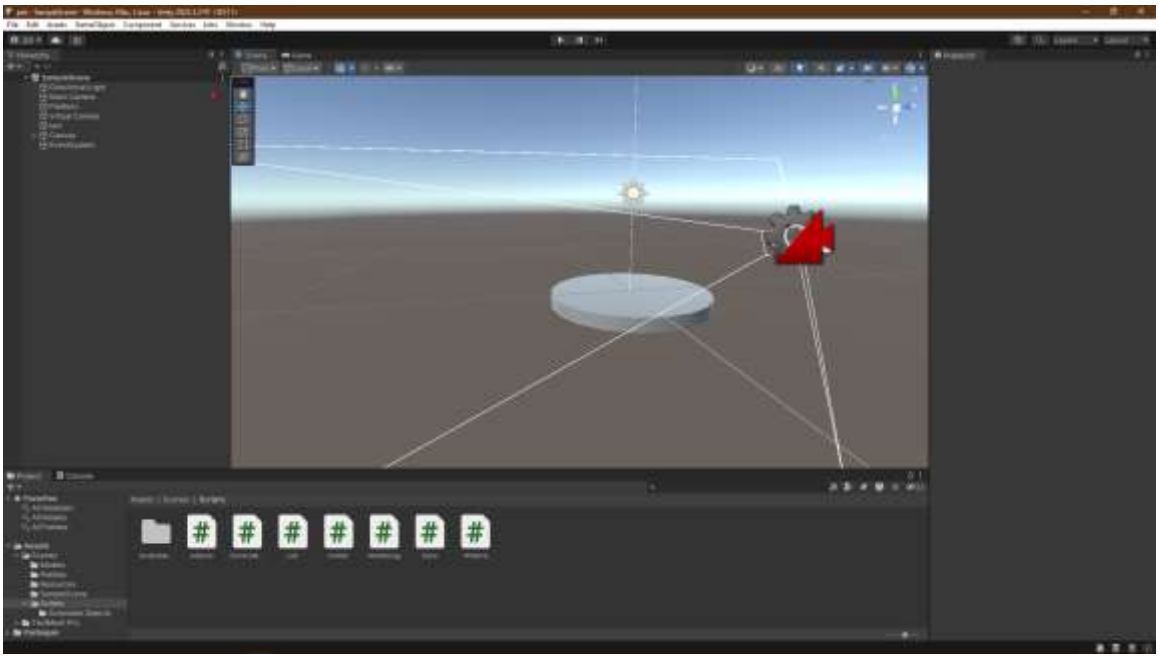

Рисунок 3.6 – Рабочее окно программы Unity

Рабочее пространство Unity называется сценой. На рисунке представлена сцена, в которой расположены начальная платформа, а также камера, из которой пользователь будет видеть рабочую область.

Для упрощения процесса разработки и при необходимости добавления на сцену большого числа одинаковых объектов в Unity имеется возможность создания префабов. Так как модули манипулятора не отличаются визуально, имеют одинаковые наборы параметров и могут появляться в манипуляторе по нескольку раз, для каждого из них был создан префаб (рисунок 3.7)

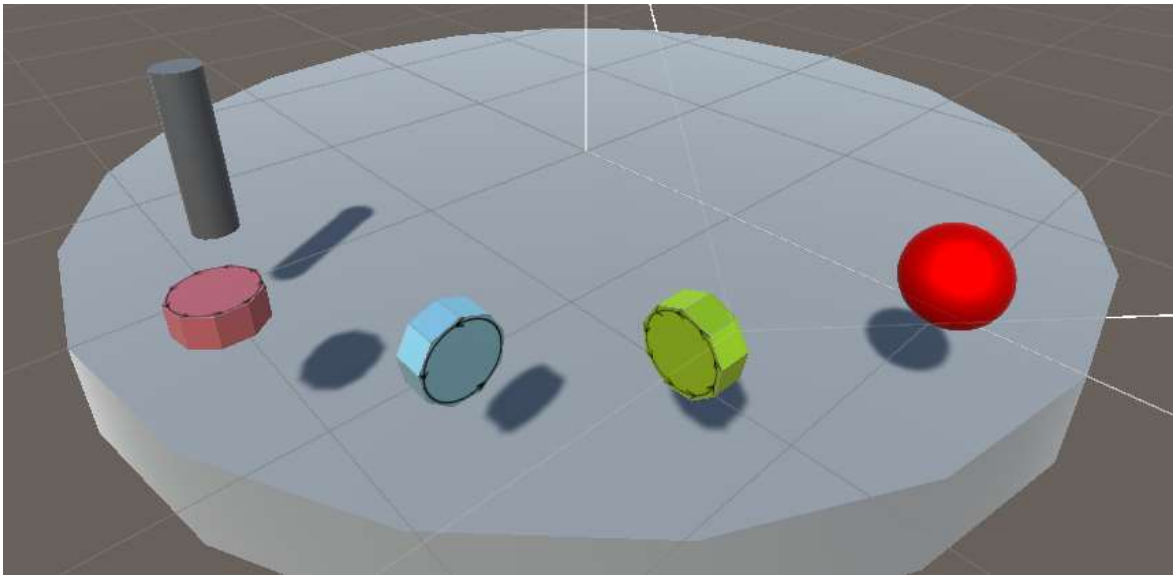

Рисунок 3.7 – Префабы модулей манипулятора

3.3 Примеры фактического тестирования программного продукта

Тестирование проводилось в два этапа. Первым тестом являлась проверка работоспособности каждого отдельного звена. Для этого были собраны простейшие модели манипуляторов, правильное решение прямой задачи кинематики которых было очевидно.

Для второго теста были выбраны уже готовые решения типовых манипуляторов с тремя и шестью звеньями.

На рисунке 3.8 представлен пример расстановки осей координат и обобщенных координат в соответствии с правилами выбранного алгоритма для трёхзвенного манипулятора.

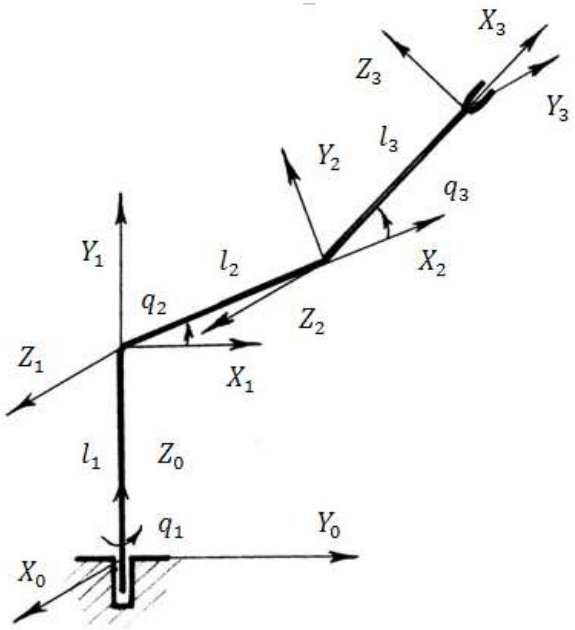

Рисунок 3.8 – Пример расположения параметров для манипулятора

Параметры матриц преобразования для манипулятора представлены в таблице 3.1.

Таблица 3.1 – Параметры для матриц преобразования манипулятора

Параметр	Системы координат		
	1	2	3
α	$\frac{\pi}{2}$	0	$-\pi$
θ	q_1	q_2	q_3
d	0	l_2	l_3
a	l_1	0	0

Матрицы преобразования вида (3.1) с учетом параметров из таблицы 3.1 примут вид:

$$A_1 = \begin{bmatrix} cosq_1 & 0 & sinq_1 & 0 \\ sinq_1 & 0 & -cosq_1 & 0 \\ 0 & 1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} cosq_2 & -sinq_2 & 0 & l_2cosq_2 \\ sinq_2 & cosq_2 & 0 & l_2sinq_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} cosq_3 & 0 & -sinq_3 & l_3cosq_3 \\ sinq_3 & 0 & cosq_3 & l_3sinq_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

где q_i – обобщённые координаты;

l_i – длина звена;

i – номер системы координат.

Искомым решением (3.2) будет являться произведение матриц из (3.1)

$$A_0^3 = \begin{bmatrix} cosq_1cos(q_2+q_3) & -sinq_1 & -cosq_1sin(q_2+q_3) & cosq_1(l_3cos(q_2+q_3)+l_2cosq_2) \\ sinq_1cos(q_2+q_3) & cosq_1 & -sinq_1sin(q_2+q_3) & sinq_1(l_3cos(q_2+q_3)+l_2cosq_2) \\ sin(q_2+q_3) & 0 & -\cos(q_2+q_3) & l_1+l_2sinq_2+l_3sin(q_2+q_3) \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3.2)$$

где q_i –обобщённые координаты;

l_i – длина звена;

i – номер системы координат.

Для проверки правильности программы был собран манипулятор по схеме, представленной на рисунке 3.9.

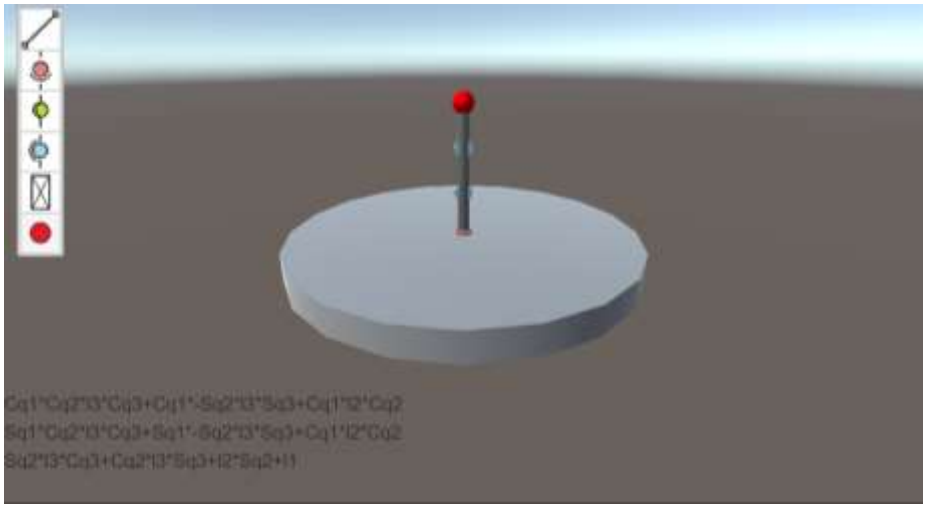

Рисунок 3.9 – Собранная модель

После нажатия кнопки «Вычислить» уравнения для поиска координат рабочего органа отобразились в текстовой области, а полное полученное решение было записано в текстовый файл (Рисунок 3.10).

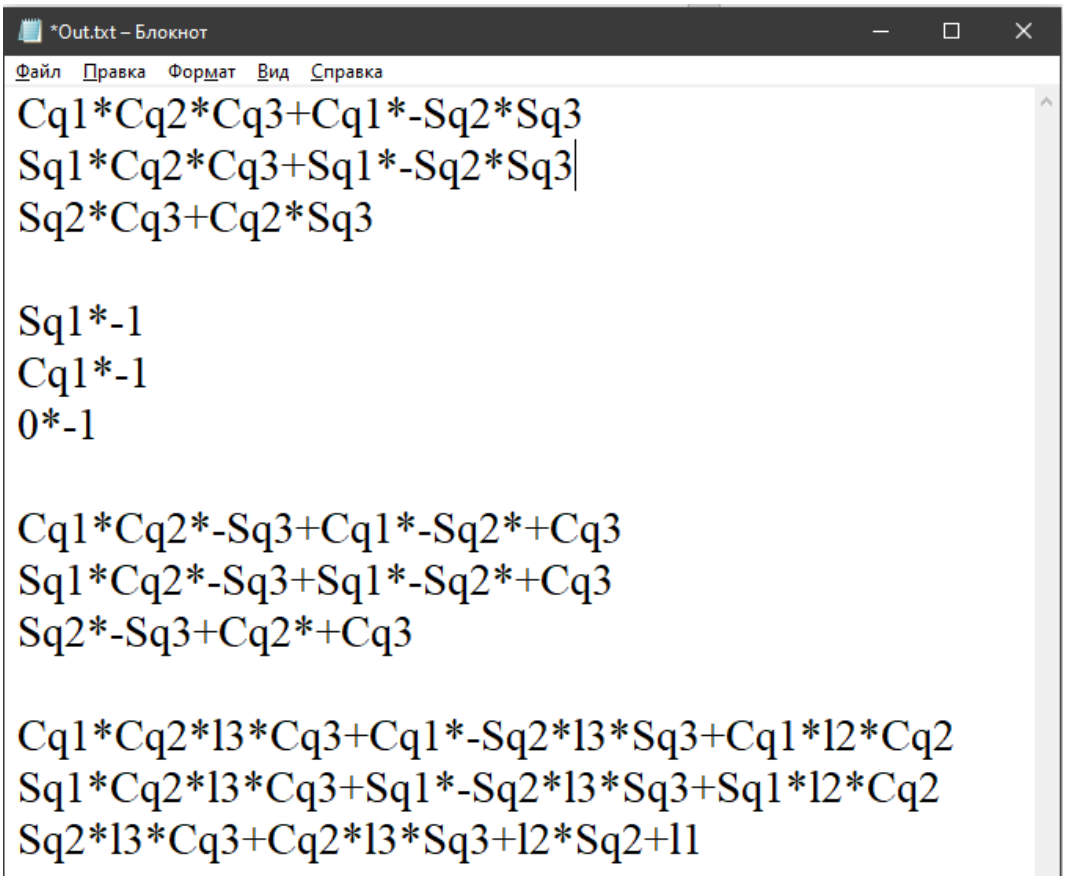

Рисунок 3.10 – Полученное решение

В выходном файле каждые три строки представляют собой первые три элемента каждого столбца. После некоторых математических преобразований было установлено, что полученное в результате применения программы решение равно искомому (3.3), что означает, что первый тест пройден.

Аналогично первому тесту, был проведен второй для более сложной шестизвенной модели манипулятора (рисунок 3.11).

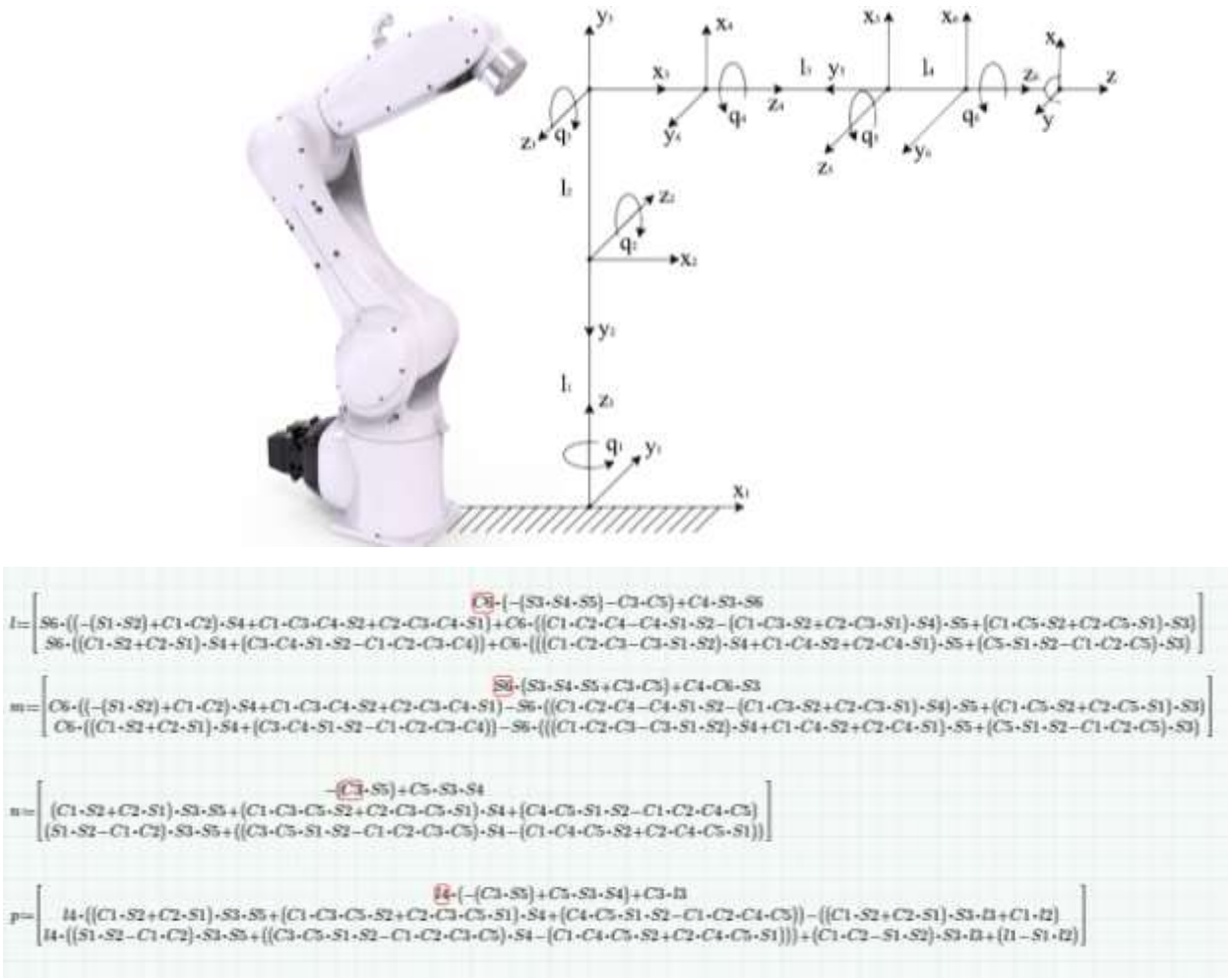

Рисунок 3.11 – Кинематическая схема и решение прямой задачи кинематики для шестизвенного манипулятора

После сборки модели и поиска с последующим преобразованием решения был получен верный результат, что означает, что программа работает правильно.

3.4 Анализ достоверности и практической значимости полученных результатов

Как было описано во второй главе, существующие манипуляторы использовались для тестирования и отладки программного обеспечения во время разработки. Поэтому можно говорить, что программа может быть использована для поиска решения прямой задачи кинематики для них.

Для проверки возможности применения программы для проектируемых манипуляторов, были вручную найдены несколько решений прямой задачи для

манипуляторов №1 и №2, показанных соответственно на рисунках 3.12 и 3.13.

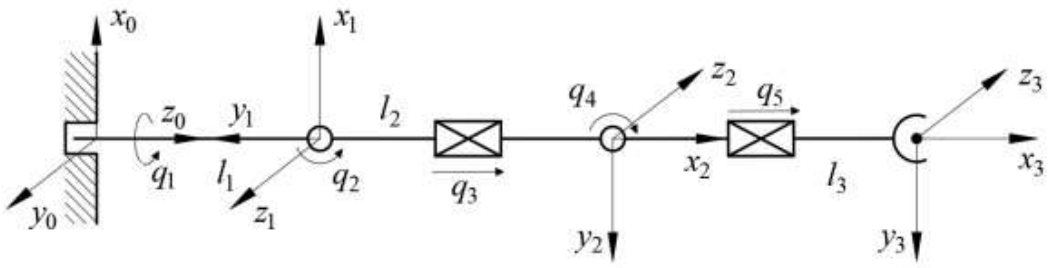

Рисунок 3.12 – Тестовый манипулятор №1

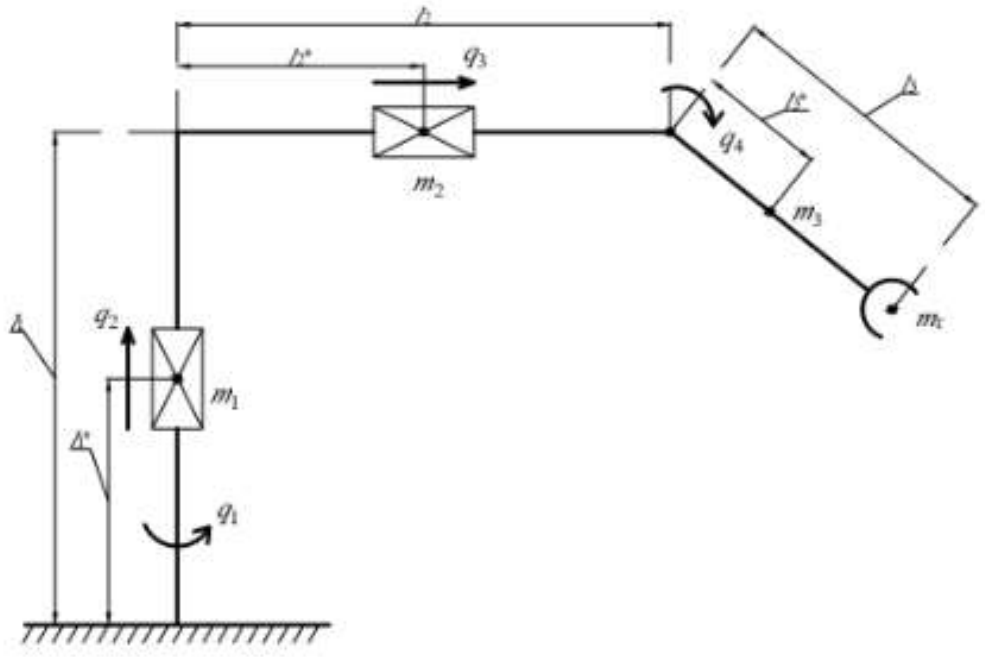

Рисунок 3.13 – Тестовый манипулятор №2

Как и в случае с существующими манипуляторами, программа выдала правильное решение. Собранные манипуляторы и решения для них показаны на рисунках 3.14 и 3.15.

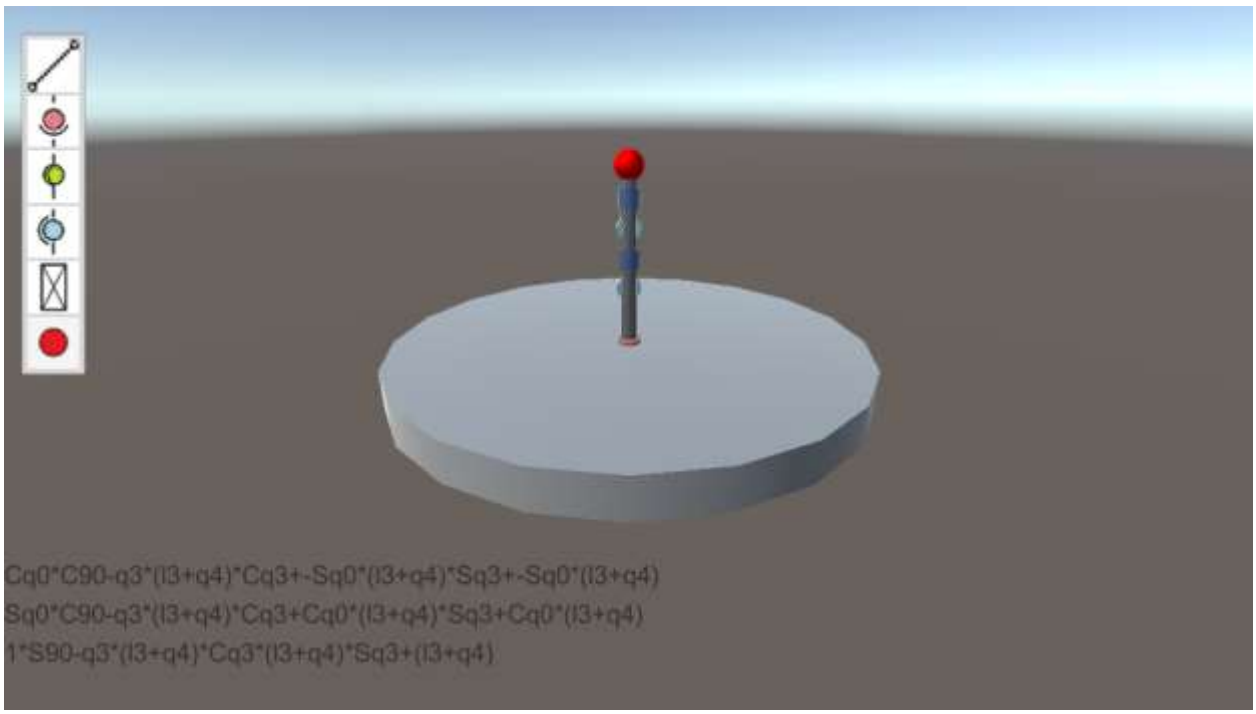

Рисунок 3.14 – Решение для тестового манипулятора №1

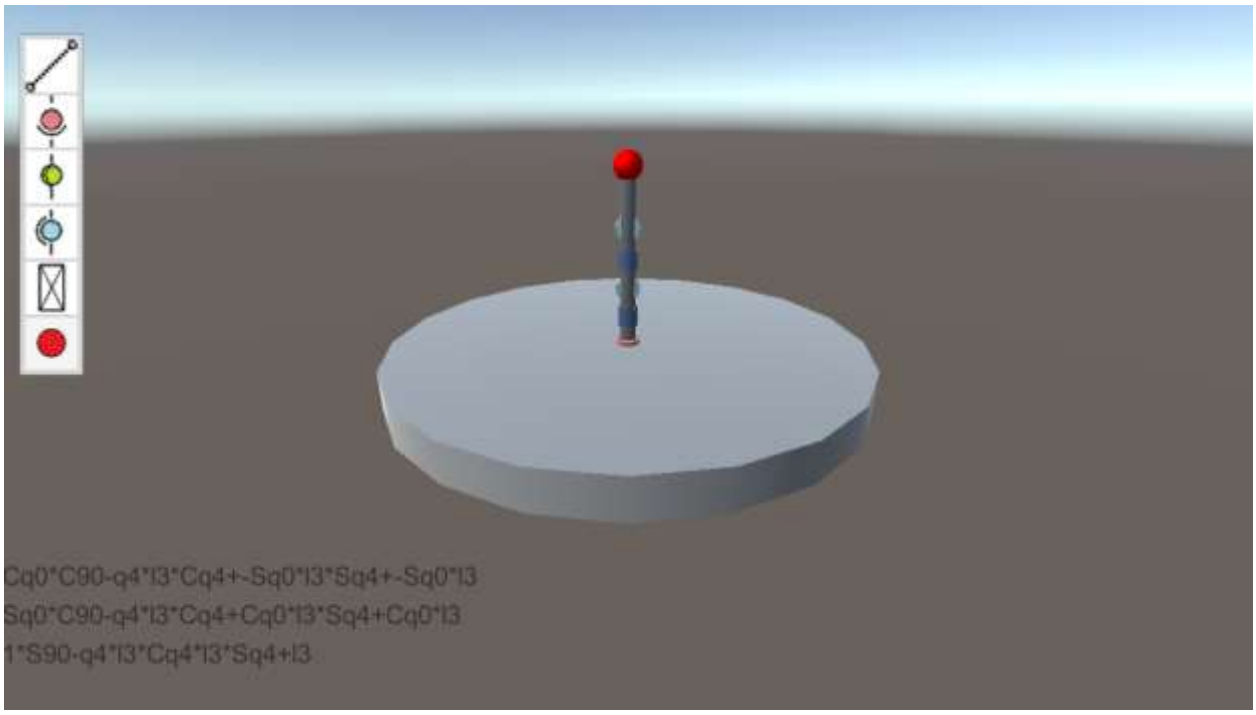

Рисунок 3.15 – Решение для тестового манипулятора №2

В результате анализа полученных решений были получены следующие выводы:

- Формулы могут занимать очень много места, как показано на рисунке

3.16 для решения прямой задачи кинематики для шестизвенного манипулятора;

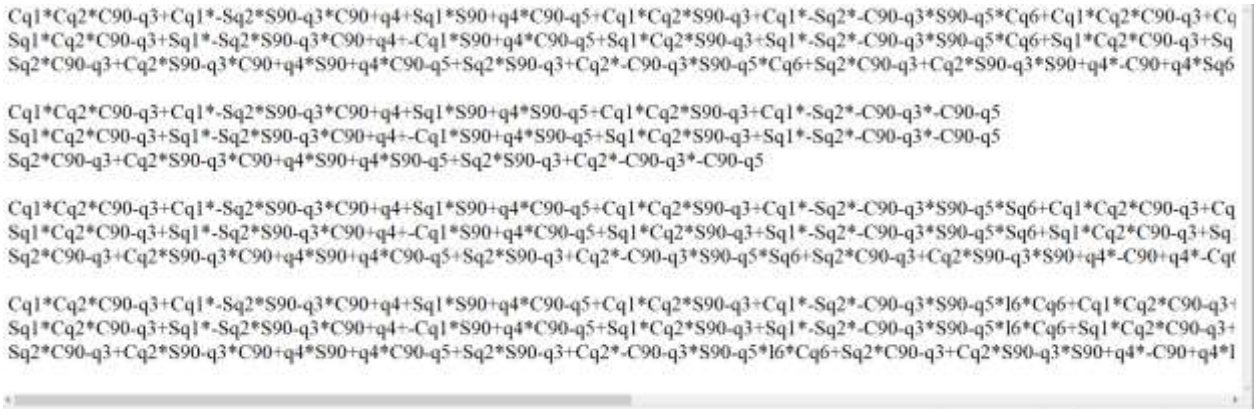

Рисунок 3.16 – Длинное решение

- Формулы необходимо дополнительно обрабатывать по правилам тригонометрических преобразований для удобства использования и скорости вычислений;

- Несмотря на указанные ранее пункты, поиск решения производится в разы быстрее по сравнению с ручным методом.

ЗАКЛЮЧЕНИЕ

В ходе выполнения магистерской работы было разработано приложение для поиска решения прямой задачи кинематики промышленного манипулятора в аналитическом виде.

В диссертации рассмотрен использующийся алгоритм поиска решения прямой задачи кинематики с помощью представления Денавита-Хартенберга.

Проведенный анализ программного обеспечения для решения подобных задач позволил определить окончательный функционал разрабатываемого приложения.

Было проведено функциональное тестирование приложения, а также проверка достоверности полученных в ходе его работы результатов с использованием реальных данных.

В дальнейшем возможно улучшение дизайна графического интерфейса, а также добавление контекстных подсказок для правильной и удобной работы пользователя с элементами программы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Вернер, Д. Ф. Разработка приложения «ПЗК манипулятора с произвольным числом звеньев» / Д. Ф. Вернер // Молодежь XXI века: шаг в будущее: Материалы XXIV региональной научно-практической конференции. –2023. – С. 151–152.

2 Вернер, Д. Ф. Разработка приложения для поиска решения прямой задачи кинематики манипулятора / Д. Ф. Вернер // Кооперация науки и общества – путь к модернизации и инновационному развитию: сборник статей Международной научно-практической конференции – Уфа: Аэтерна, 2024. – С. 36–38.

3 Вернер, Д.Ф. Применение представления Денавита-Хартенберга в приложении для поиска решения прямой задачи кинематики / Д. Ф. Вернер, Т.А. Галаган // Вестник Амурского государственного университета. – 2024. – Вып. 105: Сер. Естеств. и экон. науки. – С. 195–201.

4 Воронкин, Д. С. Решение прямой задачи кинематики для шестизвенного шарнирного робота-манипулятора / Д.С. Воронкин // Известия Тульского государственного университета. Технические науки. – 2018. – № 9. – С. 236–241.

5 Глазков, В. П. Математические модели и эффективные методы решения задач кинематики, динамики и управления роботами / В.П. Глазков, Саратов, 2005. – 164 с.

6 ГОСТ 19.101-77. Единая система программной документации. Виды программ и программных документов. – Введ. 01.01.1980. – М.: Стандартинформ, 2010. – 4 с.

7 ГОСТ 24.701-86. Единая система стандартов автоматизированных систем управления. Надежность автоматизированных систем управления. Основные положения. – Введ. 01.07.1987. – М.: Стандартинформ, 2009. – 12 с.

8 ГОСТ 24.702-85. Единая система стандартов автоматизированных систем управления. Эффективность автоматизированных систем управления. Основные положения. – Введ. 01.01.1987. – М.: Стандартинформ, 2009. – 5 с.

9 ГОСТ 24.703-85. Единая система стандартов автоматизированных систем управления. Автоматизированные системы управления. Общие требования. – Взамен ГОСТ 17195-76, ГОСТ 20912-75, ГОСТ 24205-80; введ. 01.01.1987. – М.: Стандартинформ, 2009. – 11 с.

10 ГОСТ 24.703-85. Единая система стандартов автоматизированных систем управления. Типовые проектные решения в АСУ. Основные положения. – Введ. 01.01.1987. – М.: Стандартинформ, 2009. – 4 с.

11 ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – Введ. 01.01.1990. – М.: Стандартинформ, 2009. – 12 с.

12 ГОСТ 34.603-92. Информационная технология. Виды испытаний автоматизированных систем. – Введ. 01.01.1993. – М.: Стандартинформ, 2009. – 6 с.

13 ГОСТ Р 51188-98. Защита информации. Испытания программных средств на наличие компьютерных вирусов. Типовое руководство. – Введ. 14.07.1999. – М.: Госстандарт России, 2003. – 9 с.

14 ГОСТ Р 51904-2002. Программное обеспечение встроенных систем. Общие требование к разработке. Типовое руководство. – Введ. 25.06.2002. – М.: Госстандарт России, 2005. – 67 с.

15 Гутгарц, Р. Д. Проектирование автоматизированных систем обработки информации и управления [Электронный ресурс] : учебное пособие для вузов / Р. Д. Гутгарц ; М. : Издательство Юрайт, 2024. – 351 с. – Режим доступа: https://urait.ru/bcode/541196 – 13.05.2024.

16 Евтеева, Е. В. Применение мехатронных систем в специальных и агрессивных средах / Е. В. Евтеева // Вестник ВУиТ. – 2010. – №16. – 3 с.

17 Жмудь, В. А. Введение в робототехнику / В. А. Жмудь, Я. Носек, Л. Димитров // Автоматика и программная инженерия. – 2019. – №4 (30). – С. 34-47.

18 Зараменских, Е. П. Управление жизненным циклом информационных систем [Электронный ресурс] : учебник и практикум для вузов / Е. П. Зарамен-

ских ; М. : Издательство Юрайт, 2020. – 431 с. – Режим доступа: https://urait.ru/bcode/451064 – 05.06.2024.

19 Зеленюк, Е.С. Решение прямой задачи кинематики для семизвенного манипулятора / Е. С. Зеленюк // Международный журнал гуманитарных и естественных наук. – 2023. – № 5-2 (80). – С. 89–94.

20 Казарин, О. В. Надежность и безопасность программного обеспечения [Электронный ресурс] : учебное пособие для вузов / О. В. Казарин, И. Б. Шубинский ; М. : Издательство Юрайт, 2022. – 342 с. – Режим доступа: https:// urait.ru/ bcode/493262 – 05.04.2024.

21 Каргинов, Л.А. Иерархический подход к решению обратной задачи кинематики / Л. А. Каргинов // Машиностроение и компьютерные технологии. – 2016. – №3. – С. 37–62.

22 Клокотов, И.Ю. Актуальность внедрения автоматизации технологических процессов и производств на современном этапе развития нашего общества / И. Ю. Клокотов // Международный журнал прикладных наук и технологий «Integral». – 2020. – №1. – С. 143–147.

23 Ковтун, М.В. Робототехника и автоматическое управление / М. В. Ковтун // Теория и практика современной науки. – 2023. – №4 (94). – С. 168-182.

24 Колтыгин, Д.С. Алгоритм определения обобщенных координат для робота Delta / Д. С. Колтыгин, И. А. Седельников, // Труды Братского гос. ун-та. Сер. Естественные и инженерные науки. – 2016. – Т. 2. – С. 127–130.

25 Колтыгин, Д.С. Метод и программа решения прямой и обратной задачи кинематики для управления роботом-манипулятором. /Д. С. Колтыгин, И. А. Седельников // Системы Методы Технологии. – БрГУ. Братск. – 2020. – № 4 (48). – С. 65–74.

26 Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем [Электронный ресурс] : учебник для вузов / Е. М. Лаврищева ; М. : Издательство Юрайт, 2021. – 432 с. – Режим доступа : https:// urait.ru/bcode/470923 – 07.06.2024.

27 Маглинец, Ю. А. Анализ требований к автоматизированным информационным системам: учебное пособие / Ю. А. Маглинец. – 3-е изд. – Москва, Саратов: ИнтернетУниверситет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. – 191 с.

28 Методы управления робототехническими приложениями: учебное пособие. / Борисов О.И., Громов В.С., Пыркин А.А. – СПб.: Университет ИТМО, 2016. – 108 с

29 Мкртычев, О. В. Теория механизмов и машин: практикум / О. В. Мкртычев – Москва : ИНФРА-М, 2021. – 327 с.

30 Моделирование систем и процессов. Практический курс [Электронный ресурс] : учебное пособие для вузов / В. Н. Волкова [и др.] ; М. : Издательство Юрайт, 2024. – 295 с. – Режим доступа: https://urait.ru/bcode/537202 – 23.05.2024.

31 Мякишев, Д. В. Принципы и методы создания надежного программного обеспечения АСУТП : учебное пособие / Д. В. Мякишев ; М., Вологда : Инфра-Инженерия, 2021. – 116 с.

32 Немного о роботах [Электронный ресурс]. – Режим доступа: http://roboty6.narod.ru/forwardKinematics.htm – 21.01.2024.

33 Пащенко, В. Н. Решение прямой задачи о положении шестистепенного манипулятора параллельной структуры на базе кривошипно-шатунного механизма. / В. Н. Пащенко, А. В. Романов, Е. В. Артемьев [и др.] // Электронные информационные системы. – 2017. – № 4(15). – С. 90–101.

34 Проектирование информационных систем [Электронный ресурс] : учебник и практикум для вузов / под общей редакцией Д. В. Чистова ; М. : Издательство Юрайт, 2022. – 258 с. – Режим доступа: https://urait.ru/bcode/489307 (дата обращения: 05.05.2024).

35 Пчелинцева, С. В. Математические методы в технике и технологиях: сб. трудов XV Междунар. науч. конф. / С. В. Пчелинцева // ТГТУ. Тамбов, 2002. – Т. 5. – С. 22–26.

36 Пчелинцева, С. В. Разработка методов математического моделирования кинематики промышленных манипуляторов / С. В. Пчелинцева // Саратов, 2006. – 205 с.

37 Синяков, Д.К. Темпы развития робототехники в России / Д. К. Синяков, О. Р. Ачкасов // Актуальные проблемы авиации и космонавтики. – 2016. – №12. – С. 33 – 35.

38 Системы автоматизированного проектирования. Структура. Виды обеспечений: учебное пособие / И. Л. Коробова, Д. В. Давыдова, С. А. Васильев, Д. С. Соловьёв. – Тамбов: Тамбовский государственный технический университет, ЭБС АСВ, 2019. – 89 с.

39 Смирнов, П. А, Решение прямой и обратной задач кинематики в системе позиционирования звеньев манипулятора / П. А. Смирнов, Р. Н. Яковлев // Мехатроника, автоматизация, управление. – 2019. – № 20(12). – С. 732–739.

40 Спыну, Г. А. Промышленные роботы: конструирование и применение. / Г. А. Спыну. – Киев : Вища школа, 1985. – 176 с.

41 Технология программирования : учебное пособие / Ю. Ю. Громов, О. Г. Иванова, М. П. Беляев, Ю. В. Минин. – Тамбов : Тамбовский государственный технический университет, ЭБС АСВ, 2013. – 173 с.

42 Уроки Flash – Анимация [Электронный ресурс] – Режим доступа: http://www.lessonsflash.ru/animation/ – 18.05.2024.

43 Химиченко, А.А. Перспективы робототехники / А. А. Химиченко // ЭКО. – 2008. – №9 (411). – С. 77 – 86.

44 Цымбал, А. М. Технологии программирования и робототехника / А. М. Цымбал, А. И. Бронников, А. В. Литвинова, О. Е. Чернышенко // ВЕЖПТ. – 2009. – №2 (39). – С. 56 – 60.

45 Чернухин, Ю.В. Введение в роботехнику: Учебное пособие / Ю. В. Чернухин. – Таганрог : ТРТИ, 1990 – 46 с.

46 Шаньгин, Е.С. Управление роботами и робототехническими системами. Конспект лекций / Е. С. Шаньгин. – Уфа. – 2005. – 179 с.

47 Штепа, Д.С., Перспективы внедрения робототехники в России / Д. С. Штепа, С. М. Бугрова // Инновационная наука. – 2016. – №4-3 (16). – С. 209 –210.

48 3ds MAX Support and learning [Электронный ресурс] – Режим доступа: https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2021/ENU/3DSMax-Animation/files/GUID-AE4A089-95F5-4199-A853-ABB8E0DB3439-htm.html – 18.02.2024.

49 Animating with Forward Kinematics [Электронный ресурс] – Режим доступа: https://help.autodesk.com/view/3DSMAX/2024/ENU/?guid=GUID-E8CE8A19-6C17-4121-ACA0-DE230B95B5A7 – 02.05.2024.

50 IKFK Solver [Электронный ресурс] – Режим доступа: https://www.joleanes.com/ikfksolver.html – 02.04.2024.

ПРИЛОЖЕНИЕ А

Техническое задание

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Полное наименование системы

Решение прямой задачи кинематики для промышленного манипулятора.

1.2 Наименование предприятий разработчика и заказчика системы

Разработчик: студент группы 2105-ом института компьютерных и инженерных наук Амурского государственного университета Вернер Денис Фёдорович.

Заказчик: Галаган Татьяна Алексеевна

1.3 Перечень документов

Перечень документов, на основе которых проектируется система:

– ГОСТ 34.602-89 – техническое задание на проектирование автоматизированной системы управления;

– инструкция по охране труда при работе на персональном компьютере;

– первичные документы;

– должностные инструкции сотрудников.

1.4 Плановые сроки начала и окончания работы

Плановые сроки начала и окончания работ по созданию системы: начало разработки – 29.01.2024 г., окончание – 19.06.2024 г.

2 ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

2.1 Основание для проведения разработки

Основанием для разработки является задание к магистерской работе по дисциплине: «Технология разработки программного обеспечения», полученное в организации Амурский Государственный Университет (ФГБОУ ВО «АмГУ»), Институт Компьютерных и Инженерных Наук, Кафедра информационных и управляющих систем.

2.2 Наименование и условное обозначение темы разработки

Полное наименование темы разработки: «Разработка приложения "Решение прямой задачи кинематики для промышленного манипулятора"»

Условное обозначение – “Расчёт ПЗК”.

3 НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ

3.1 Функциональное назначение программы

Функциональным назначением программы является расчёт прямой задачи кинематики в аналитическом виде для манипулятора, моделируемого пользователем.

3.2 Эксплуатационное назначение программы

Программа должна использоваться во время проектировки управляющих систем манипуляторов. Конечными пользователями программы являются инженеры профиля мехатроника и робототехника.

4 ТРЕБОВАНИЯ К ПРОГРАММЕ

4.1 Требования к функциональным характеристикам

4.1.1 Требования к составу выполняемых функций

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- моделирование манипулятора из линейных, вращательных и телескопических звеньев;
- расчёт аналитического решения ПЗК для полученного манипулятора;
- вывод полученного решения на экран;

4.1.2 Требования к организации выходных данных

Выходные данные должны представлять собой уравнения, являющиеся решением ПЗК манипулятора в текстовом виде.

4.1.3 Требования к временным характеристикам

Требования к временным характеристикам зависит от выполняемой задачи. При вычислении решения временные рамки увеличиваются пропорционально обрабатываемым данным.

4.2 Требования к надежности

4.2.1 Требования к обеспечению надежного (устойчивого) функционирования программы

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:

организацией бесперебойного питания технических средств;

выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;

выполнением требований ГОСТ 51188-98. Защита информации. Испытания программных средств на наличие компьютерных вирусов;

необходимым уровнем квалификации сотрудников профильных подразделений.

4.2.2 Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени, необходимого на перезагрузку операционной системы и запуск программы, при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

4.2.3 Отказы из-за некорректных действий оператора

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

4.3 Условия эксплуатации

4.3.1 Климатические условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

4.3.2 Требования к численности и квалификации персонала

Для работы программы необходим один оператор. Оператор должен иметь среднее профильное образование и меть навыки работы с графическим интерфейсом. Оператор должен владеть понятиями кинематика, ПЗК, а также иметь представление об основных модулях, из которых могут состоять манипуляторы типа «рука».

4.4 Требования к составу и параметрам технических средств

В состав технических средств должен входить персональный компьютер.

5 СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ СИСТЕМЫ

5.1 Стадии разработки

Разработка должна быть проведена в три стадии:

- разработка технического задания;

- рабочее проектирование;

- внедрение;

5.2 Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

- разработка программы;

- разработка программной документации;

- испытания программы.

На стадии внедрения должен быть выполнен этап разработки - подготовка и передача программы.

5.3 Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

- постановка задачи;

- определение и уточнение требований к техническим средствам;

- определение требований к программе;

- определение стадий, этапов и сроков разработки программы и документации на неё;

- выбор языков программирования;

- согласование и утверждение технического задания;

На этапе разработки программы должна быть выполнена работа по программированию и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77 и требованием п. «Предварительный состав программной документации» настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

- разработка, согласование и утверждение методики испытаний;

- проведение приемо-сдаточных испытаний;

- корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию.

5.4 Сроки выполнения

На разработку информационной системы отводится срок с 29 января 2024 по 19 июня 2024.

5.5 Состав организации исполнителя работ

Все работы выполняются студентом Амурского государственного университета Вернером Денисом Фёдоровичем.

5.6 Вид и порядок экспертизы технической документации

Экспертиза технической документации проводится в соответствии с утвержденным планом, включая анализ требований, проектные документы и результаты тестирования. Экспертиза осуществляется независимыми специалистами с целью удостоверения соответствия разрабатываемой системы заявленным требованиям и стандартам.

6 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ СИСТЕМЫ

6.1 Виды, состав, объем и методы испытания

Испытания приложения «Решение ПЗК» включают в себя следующие:

1 этап – проверка взаимодействия модулей системы;

2 этап – проверка корректности работы функций системы;

3 этап – оценка производительности системы при увеличении вычислительных нагрузок;

4 этап – проверка возможности внесения изменений в систему;

Объем испытаний включает в себя тестирование каждого функционала, компонента и интеграцию между ними.

Методы испытаний включают тестирование через пользовательский интерфейс, тестовые данные, анализ кода и другие средства проверки.

6.2 Общие требования приемки работ по стадиям

На каждой стадии создания системы предусмотрены следующие требования приемки:

- стадия анализа: проведение экспертизы требований и аналитической документации;

- стадия проектирования: проверка архитектуры и дизайна системы;

- стадия реализации: проверка корректности написанного кода и функциональности;

- стадия тестирования: проверка на соответствие системы утвержденным тестовым кейсам;

7 ТРЕБОВАНИЯ К ДОКУМЕНТИРОВАНИЮ

7.1 Перечень подлежащих обработке документов

Документирование включает в себя подготовку и утверждение следующих видов документации:

– техническое задание: подробное описание требований к системе и особенностей ее функционирования.

7.2. Перечень документов на машинных носителях

Все перечисленные документы должны быть представлены в электронном виде на машинных носителях для обеспечения удобства доступа, резервирования и долгосрочного хранения.

8 ДОКУМЕНТЫ И ИНФОРМАЦИОННЫЕ МАТЕРИАЛЫ, НА ОСНОВАНИИ КОТОРЫХ РАЗРАБАТЫВАЕТСЯ ТЕХНИЧЕСКОЕ ЗАДАНИЕ

- ГОСТ 24.104-85. Единая система стандартов автоматизированных систем управления. Общие требования;

- ГОСТ 34.601-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания;

- ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы;

- ГОСТ 34.603-92. Информационная технология. Комплекс стандартов на автоматизированные системы. Виды испытаний автоматизированных систем;

- ГОСТ 24.701-86. Единая система стандартов автоматизированных систем управления. Надежность автоматизированных систем управления. Основные положения;

- ГОСТ 24.702-85. Единая система стандартов автоматизированных систем управления. Эффективность автоматизированных систем управления. Основные положения;

- ГОСТ 24.703-85. Единая система стандартов автоматизированных систем управления. Типовые проектные решения в АСУ. Основные положения;

ПРИЛОЖЕНИЕ Б

Код класса “AddLink”

```
using System;
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEditor;
using UnityEngine;
using UnityEngine.UI;


public class AddLink : MonoBehaviour
{
  public LinkSet link;
  public void DoStuff(GameObject prefab)
  {
    Vector3 pos = ManManag.getPos();
    GameObject currLink = Instantiate(prefab, pos, Quaternion.identity);
    link = currLink.GetComponent<LinkSet>();
    link.UpdateMe( );

    if (link.LinkType == "Telescope")
    {
      Debug.Log(pos.x + " " + (pos.y-0.3f) + " " + pos.z + " ");
      currLink.transform.position = new Vector3(pos.x, pos.y-0.3f, pos.z);
    }

    ManManag.setPos(pos[0] + link.newPos[0], pos[1] + link.newPos[1], pos[2] +
link.newPos[2]);

    switch (link.LinkType)
    {
      case ("Hard_Bar"):
        link.link.Q = ("l" + Convert.ToString(ManManag.barCount));
        ManManag.barCount++;
        ManManag.Links.Add(link.link);
        break;
      case ("Telescope"):
        ManManag.Links[ManManag.Links.Count-1].Q     =     ("("     +     Man-
Manag.Links[ManManag.Links.Count - 1].Q +
                              "+q" + Convert.ToString(ManManag.jointCount)+")");
        ManManag.jointCount++;
        break;
      default:
        link.link.Q = ("q" + Convert.ToString(ManManag.jointCount));
        ManManag.jointCount++;
        ManManag.Links.Add(link.link);
        break;
    }
  }
}
```

ПРИЛОЖЕНИЕ В

Код класса “Solve”

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using UnityEngine.UI;
using static ManManag;

public class Solve : MonoBehaviour
{
  private bool doubled = false;
  public void FindSolution(Text display)
  {
  Debug.Log("Search started");
  {
    for (int i = 2; i < ManManag.Links.Count; i++)
    {
    if ((ManManag.Links[i].LinkType != "Vertical_Joint") &&
       (ManManag.Links[i].LinkType != "Horizontal_Joint") &&
       (ManManag.Links[i].LinkType != "Turn_Joint") &&
       (ManManag.Links[i].LinkType != "Tool"))
        {
          continue;
        }
        else
        {
          switch (ManManag.Links[i].LinkType)
          {
            case "Vertical_Joint":
              {
                switch (ManManag.Links[i - 2].LinkType)
                {
                  case "Vertical_Joint":
                    {
                      ManManag.TFMs.Add(new TFMatrix(ManManag.Links[i-2].Q, "0",
"0", (ManManag.Links[i - 1].Q)));
                      break;
                    }
                  case "Horizontal_Joint":
                    {
                      ManManag.TFMs.Add(new TFMatrix((ManManag.Links[i - 2].Q),
"90", (ManManag.Links[i - 1].Q), "0"));
                      break;
                    }
                  case "Turn_Joint":
                    {
                      ManManag.TFMs.Add( new TFMatrix(("90+" + ManManag.Links[i
- 2].Q), "90", "0", "0"));
                      break;
                    }
                  default: break;
                }
                break;
              }
            case "Horizontal_Joint":
              {
                switch (ManManag.Links[i - 2].LinkType)
                {
                  case "Vertical_Joint":
                    {
```

```
                    ManManag.TFMs.Add(  new  TFMatrix((ManManag.Links[i  -  2].Q),
"90", (ManManag.Links[i - 1].Q), "0"));
                      break;
                    }
                  case "Horizontal_Joint":
                    {
                      ManManag.TFMs.Add(new  TFMatrix((ManManag.Links[i  -  2].Q),
"90", (ManManag.Links[i - 1].Q), "0"));
                      break;
                    }
                  case "Turn_Joint":
                    {
                      ManManag.TFMs.Add(new   TFMatrix(("-"+ManManag.Links[i   -
2].Q), "0", (ManManag.Links[i - 1].Q), "0"));
                      break;
                    }
                  default: break;
                }
                break;
              }
            case "Turn_Joint":
              {
                switch (ManManag.Links[i - 2].LinkType)
                {
                  case "Vertical_Joint":
                    {
                      if (doubled)
                      {
                        ManManag.TFMs.Add( new TFMatrix((ManManag.Links[i-2].Q),
"-90", (ManManag.Links[i-3].Q + "+" + ManManag.Links[i-1].Q), "0"));
                      }
                      else
                        ManManag.TFMs.Add(  new  TFMatrix(ManManag.Links[i-2].Q,
"-90", ManManag.Links[i-1].Q, "0"));
                      break;
                    }
                  case "Horizontal_Joint":
                    {
                      if (doubled)
                      {
                        ManManag.TFMs.Add(  new  TFMatrix(ManManag.Links[i-2].Q,
"-90", (ManManag.Links[i-3].Q + "+" + ManManag.Links[i-1].Q), "0"));
                      }
                      else
                        ManManag.TFMs.Add(new    TFMatrix((ManManag.Links[i    -
2].Q), "-90", (ManManag.Links[i - 1].Q), "0"));
                      break;
                    }
                  case "Turn_Joint":
                    {
                      if (doubled)
                      {
                        ManManag.TFMs.Add(    new    TFMatrix(("90-"    +    Man-
Manag.Links[i-2].Q), "90", (ManManag.Links[i - 3].Q + "+" + ManManag.Links[i -
1].Q), "0"));
                      }
                      else
                        ManManag.TFMs.Add(    new    TFMatrix(("90-"    +    Man-
Manag.Links[i - 2].Q), "90", (ManManag.Links[i - 1].Q), "0"));
                      break;
                    }
```

```
                default: break;
              }
              break;
            }
          case "Tool":
            {
              switch (ManManag.Links[i - 2].LinkType)
              {
                case "Vertical_Joint":
                  {
                    ManManag.TFMs.Add( new TFMatrix((ManManag.Links[i - 2].Q),
"-90", "0", (ManManag.Links[i - 1].Q)));
                    break;
                  }
                case "Horizontal_Joint":
                  {
                    ManManag.TFMs.Add(new TFMatrix(("90+" + ManManag.Links[i -
2].Q), "90", (ManManag.Links[i - 1].Q), "0"));
                    break;
                  }
                case "Turn_Joint":
                  {
                    ManManag.TFMs.Add(new  TFMatrix((ManManag.Links[i  -  2].Q),
"90", "0", (ManManag.Links[i - 1].Q)));
                    break;
                  }
                default: break;
              }
              break;
            }
          default: break;
        }
      }
    }
    for (int i = 1; i < ManManag.TFMs.Count; i++)
    {
      ManManag.TFMs[0].TFMultiply(ManManag.TFMs[0], ManManag.TFMs[i]);
    }
    Debug.Log("Search ended");
    display.text  = FormatCheck(ManManag.TFMs[0].GetTF(0, 3)) + "\n" +
            FormatCheck(ManManag.TFMs[0].GetTF(1, 3)) + "\n" +
            FormatCheck(ManManag.TFMs[0].GetTF(2, 3));
    try
    {
      StreamWriter sw = new StreamWriter("Out.txt", false);
      for (global::System.Int32 i = 0; i < 4; i++)
      {
        for (global::System.Int32 j = 0; j < 3; j++)
        {
          sw.WriteLine(FormatCheck(ManManag.TFMs[0].GetTF(j, i)));
        }
      }
      sw.Close( );
    }
    catch (Exception ex)
    {
      Console.WriteLine("Exception: " + ex.Message);
    }
    finally
    {
      Console.WriteLine("Executing finally block.");}}}}
```

ПРИЛОЖЕНИЕ Г

Код класса “CameraMovement”

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class cameraMove : MonoBehaviour
{

    // VARIABLES
    public float panSpeed = 4.0f;
    public Transform cameraTarget;
    private Vector3 mouseOrigin;
    private bool isPanning;
    public float minimumAngle;
    public float maximumAngle;

    // Update is called once per frame
    void Update( )
    {

        if (Input.GetMouseButtonDown(1))
        {
            //right click was pressed
            mouseOrigin = Input.mousePosition;
            isPanning = true;
        }

        // cancel on button release
        if (!Input.GetMouseButton(1))
        {
            isPanning = false;
        }
        //move camera on X & Y
        if (isPanning)
        {
            Vector3  pos            =  Camera.main.ScreenToViewportPoint  (In-
put.mousePosition - mouseOrigin);

            // update x and y but not z
            Vector3 move    = new Vector3 (pos.x * panSpeed, pos.y * panSpeed,
0);

            cameraTarget.rotation *= Quaternion.Euler(move.y, -move.x, 0);
            var angleX = cameraTarget.localEulerAngles.x;
            Debug.Log(angleX);
            if (angleX > 180 && angleX < maximumAngle)
            {
                angleX = maximumAngle;
            }
            else if (angleX < 180 && angleX > minimumAngle)
            {
                angleX = minimumAngle;
            }

            cameraTarget.localEulerAngles  =  new  Vector3(angleX,  cameraTar-
get.localEulerAngles.y, 0);

        }
    }
}
```

ПРИЛОЖЕНИЕ Д

Код класса “ManManag”

```
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;

public class ManManag : MonoBehaviour
{
    public static ManManag instance;
    public static List<Link> Links = new List<Link>();
    public static List<TFMatrix> TFMs = new List<TFMatrix>();
    public static int barCount = 1;
    public static int jointCount = 0;
    public Vector3 position = new Vector3();

    private void Awake( )
    {
        instance = this;
        position[0] = 0;
        position[1] = 0;
        position[2] = 0;
    }

    public static Vector3 getPos()
    {
        return instance.position;
    }

    public static void setPos(float x, float y, float z )
    {
        Debug.Log("Old position is " + instance.position[0] + " " + in-
stance.position[1] + " " + instance.position[2] + " ");
        instance.position.Set(x, y, z);
        Debug.Log("New position is " + instance.position[0] + " " + in-
stance.position[1] + " " + instance.position[2] + " ");
        return;
    }

    public static string FormatCheck(string line)
    {
        bool done = false;
        while (!done)
        {
            done = true;
            if (line.StartsWith("+") || line.StartsWith("-"))
            {
                done = false;
                line = line.Remove(0, 1);
            }
            if (line.StartsWith("0+"))
            {
                done = false;
                line = line.Remove(0, 2);
            }
            if (line.Contains("+0"))
            {
                done = false;
                line = line.Remove(line.IndexOf("+0"), 2);
            }
            if (line.Contains("++"))
            {
```

```
                done = false;
                line = line.Replace("++", "+");
            }
            if (line.Contains("--"))
            {
                done = false;
                line = line.Replace("--", "+");
            }
        }
        return line;
    }

    // Start is called before the first frame update
    void Start()
    {
        ManManag.setPos(0, 0, 0);
    }

    // Update is called once per frame
    void Update()
    {

    }
}
```