Содержание

Введение

Современное гибкое производство невозможно без мобильных роботов, т.к. мобильность рабочих инструментов является необходимо для обеспечения гибкости. Для управления роботом необходимо найти зависимость положения и ориентации рабочего инструмента от углов поворотов звеньев, и, наоборот, зависимость углов поворотов звеньев от положения и ориентации робота. Первая является широко изученной и применяется повсеместно. Установление второй зависимости до сих пор является открытой проблемой, что является серьёзным препятствием для внедрения мобильных роботов в реальные гибкие производственные процессы. Данная диссертация посвящена универсальным алгоритмам решения прямой и обратной задач кинематики, а также инструментам симуляции роботических единиц.

На данный момент основным способом параметризации ориентации рабочего инструмента являются углы Эйлера, однако определения расстояния между двумя ориентациями они не подходят, т.к. не образуют биективное многообразие. Поэтому оптимальное решение обратной задачи кинематики при использовании углов Эйлера невозможно.

Для оптимального решения обратной задачи кинематики в данной диссертации был раскрыт подход параметризации ориентации рабочего инструмента при помощи единичных кватернионов поворота.

Для проверки работоспособности алгоритма управление роботом было апробировано на виртуальном роботе реализованном при помощи симулятора Gazebo.

1 Обзор существующих решений

1.1 Ориентация рабочего инструмента

Мировые координаты включают в себя три измерения, отвечающих за положение рабочего инструмента(энд-эффектора), и три измерения, отвечающих за ориентацию инструмента в пространстве. Чаще всего используют углы Эйлера [3]. Углы Эйлера представляют из себя три угла, определяющие последовательные повороты тела вдоль трёх осей. В зависимости от выбора осей меняется нотация. Подход к нахождению углов Эйлера детально рассмотрен в статье [6].

В инженерной практике принято определять ориентацию рабочего инструмента через углы Эйлера, это связано с историческим развитие теоретической механики и исторически сложившейся сферы применения(гироскопия, авионика). Использование углов Эйлера в управлении роботами является неэффективным инструментом в силу причин, описанных выше. Поэтому в рамках данной магистерской работы ориентация рабочего инструмента будет задаваться при помощи параметров Родрига-Гамильтона [10].

Для параметризации поворота в статье [9] раскрыт способ перейти к описанию конечного поворота при помощи трёх параметров.

1.2 Обратная задача кинематики

Подход, предложенный в [4] подразумевает геометрический подход в решении обратной задачи кинематики и работают только для артикулированных роботов. Т.е. первые три звена робота переводят рабочий инструмент в заданное положение, а вторые три - в заданную ориентацию. В рамках данной магистерской диссертации рассматривается пятизвенный неартикулированный робот, причём часть обобщённых координат соответствуют движению тележки, а не роботической руки.

1.2.1 CCD

CCD - это итеративный численный алгоритм. Поскольку CCD является итеративным, поэтому при его использовании нельзя применять ограничения на каждом шаге, просто беря результирующую ориентацию для любого соединения и заставляя его оставаться в допустимом диапазоне. Однако это влияет на способность CCD сходиться. CCD - это слепой

алгоритм горизонтального скручивания; он перемещается по местности, определенной функцией ошибки, пытаясь найти самую высокую точку. Подробное описание алгоритма можно найти в статье [16].

1.2.2 Алгоритм светлячка

Алгоритм Firefly(FA) или алгоритм светлячка, предложенный доктором Синь-Янь, является метаэвристическим алгоритмом оптимизации, основанный на социальном поведении светлячков в тропических климатов. Алгоритм светлячка легко реализуется и значительно превосходит другие алгоритмы с точки зрения точности и эффективности. FFA основано на поведении ошибок светлячков, включая световое излучение, поглощение света и взаимное притяжение, которое было разработано для решения задач непрерывной оптимизации Поскольку алгоритм Firefly является новым алгоритмом, он редко используется в обратной задаче кинематики. В существующих исследованиях, определяющих критерий оптимальности, основанный на частоте ошибок, представляющих собой расстояния между целевым и фактическим положением. Подробное описание алгоритма можно найти в статье [17].

1.2.3 Матрица Якоби

Согласно теории [4], матрица Якоби связывает скорости рабочего инструмента в декартовом пространстве со скоростями вращения сочленений робота:

$$J_f(a) := \left(\frac{\partial f_i}{\partial x_j}(a)\right)_{i=1,\dots,m;\ j=1,\dots,n} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(a) & \frac{\partial f_1}{\partial x_2}(a) & \dots & \frac{\partial f_1}{\partial x_n}(a) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(a) & \frac{\partial f_m}{\partial x_2}(a) & \dots & \frac{\partial f_m}{\partial x_n}(a) \end{pmatrix} \quad (1)$$

В случае мобильных роботов x_j - это углы поворота звеньев, n - количество звеньев, f_i - это координаты положения рабочего инструмента, которое является функцией от углов поворотов звеньев.

После нахождения связи между скоростью рабочего инструмента, можно построить регулятор, "поворачивающий"звенья так, чтобы линейная скорость рабочего инструмента была направлена в сторону целевого положения. Подробное описание алгоритма можно найти в статье [18].

1.2.4 Колония муравьёв

Метод колонии муравьёв(Ant Colony Optimisation) - относительно новый подход к решению проблем, который создан на основе наблюдений за социальным поведением некоторых видов муравьёв. Эти муравьи помещают феромон на землю, чтобы отметить какой-то благоприятный путь, за которым должны следовать другие члены колонии. Оптимизация методом колонии муравьев использует аналогичный механизм для решения задач оптимизации. С начала девяностых годов, когда был предложен первый алгоритм оптимизации муравьиной колонии. Подробное описание алгоритма можно найти в статье [19]

2 Симуляция

Так как работа с реальными роботами сопряжена с риском порчи оборудования и, более того, не всегда существует возможность прямого доступа к тому или иному роботу, то зачастую прибегают к использованию физических симуляторов для отладки алгоритмов. В рамках данной диссертации симуляция выполнена с использованием физического симулятора Gazebo, управление было реализовано при помощи ROS(Robot Operation System), математические вычисления были выполнены при помощи программного обеспечения Matlab.

2.1 ROS

ROS - это промежуточное программное обеспечение для робототехники. Он представляет из себя набор программных фреймворков(готовых проектов) для разработки программного обеспечения роботов. Хотя ROS в действительности не является операционной системой, он предоставляет следующие возможности:

а низкоуровневое управление устройствами

б аппаратная абстракция

в внедрение широко используемых функций

г передача сообщений между процессами и управление пакетами

Взаимодействие процессов в ROS основано на архитектуре графа, где обработка выполняется в узлах, которые могут принимать, отправлять и мультиплексировать показания датчиков, управления, состояния, планирования, исполнительные механизм и другие сообщения.

Несмотря на важность управления роботом в режиме реального времени(т.е. время между итерациями фиксировано и заранее определено), сам ROS не является оперативной ОС реального времени(ОСРВ), хотя есть возможность интегрировать ROS с кодом реального времени.

Поддержку систем реального времени планируется реализовать при создании ROS 2.0.

Программное обеспечение в ROS экосистемы можно разделить на три группы:

а независимые от языка и платформы инструменты, используемые для создания и распространения программного обеспечения на основе ROS;

б реализации клиентской библиотеки ROS такие как: roscpp, rospy и roslisp;

в пакеты, содержащие приложения, который использует одну или несколько клиентских библиотек ROS.

Как независимые от языка инструменты, так и основные клиентские библиотеки (C ++, Python и Lisp) выпускаются в соответствии с условиями лицензии BSD. Это открытое программное обеспечением. Его использование бесплатно как для коммерческих, так и для исследовательских целей.

Основные клиентские библиотеки ROS (C++, Python и Lisp) ориентированы на Unix-подобную систему, в первую очередь из-за их зависимости от продуктов с открытым исходным кодом, которые в основном реализованы только для unix-систем.

2.2 Gazebo

Моделирование роботов является важным инструментом в реальных производственных и научно-исследовательских задачах. Хорошо спроектированный симулятор позволяет быстро тестировать алгоритмы, создавать роботов, проводить физическую симуляцию и тренировать систему ИИ с использованием реалистичных сценариев.

Gazebo предоставляет возможность точно и эффективно моделировать не только роботов, но и сцены(физическое окружение робота).

Робот или сцена описывается при помощи файлов формата URDF(The Universal Robotic Description Format) - это формат, основанный на XML-разметке. Он используется не только в Gazebo, но и непосредственно в ROS для описания всех элементов робота. Чтобы использовать файл URDF в Gazebo, для правильной работы необходимо добавить некоторые дополнительные теги, отвечающие за работу симулятора.

В соответствии с описанием сцены, Gazebo запускает симуляцию и позволяет управлять приводами робота. Причём симулятор корректно моделирует не только кинематику, т.е. отрисовывает звенья робота в соответствии с углами поворота, но и учитывает динамику. Для этого каждое звено и каждое сочленение робота нужно описать не только кинематически, но и динамически.

2.3 Звенья робота

Как уже было сказано выше, всё описание сцены помещено в XML-файл с набором предопределённых тегов. В первую очередь любая система состоит из тел или звеньев робота(Links). Каждое тело представляет собой парный тег <link>...</link>. Внутри тега при помощи атрибута «name» необходимо указать имя звена. Это необходимо для наложения взаимосвязей и обращения к данным, связанным с этим звеном.

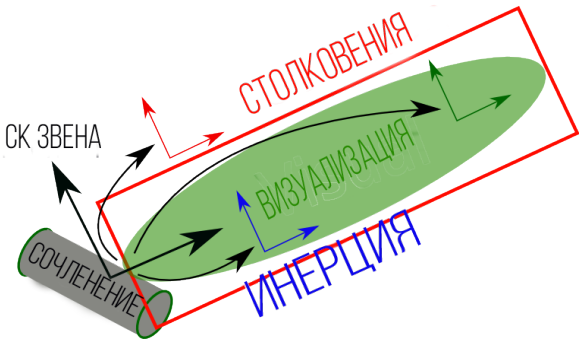

Рисунок 1 – Звено робота

Для каждого тела задаются следующие параметры:

а <inertia> этот блок отвечает за инерционные свойства звена.

1 <origin> - преобразование системы координат для перехода в СК центра тяжести тела. По умолчанию представляет собой нулевое смещение и нулевые повороты по углам Эйлера. Иначе говоря, однородная матрица преобразования является единичной, не изменяющей СК. У этого тега два атрибута:

i. xyz - смещение, по умолчанию - нулевой вектор

ii. rpy - углы Эйлера в радианах: крен, тангаж и рыскание

2 <mass> - тег, определяющий массу звена.

3 <inertia> - тензор инерции, представляющий собой симметричную матрицу 3x3. Тогда для однозначного определения такой матрицы достаточно определить 6 наддиагональных элементов, используя атрибуты ixx, ixy, ixz, iyy, iyz, izz.

б <visual> определяет визуальные свойства звена. Этот элемент определяет форму объекта (прямоугольный параллелепипед, цилиндр и т. д.) Несколько экземпляров тегов <visual> могут

быть определены внутри одного звена, это эквивалентно объединению геометрии, которую определяют указанные теги.

1 <origin> - преобразование системы координат для перехода в СК описываемого геометрического объекта. По умолчанию представляет собой нулевое смещение и нулевые повороты по углам Эйлера. Иначе говоря, однородная матрица преобразования является единичной, не изменяющей СК. У этого тега два атрибута:
   i. xyz - смещение, по умолчанию - нулевой вектор
   ii. rpy - углы Эйлера в радианах: крен, тангаж и рыскание

2 <geometry> - этот тег описывает внешний вид звена, может содержать один из следующих тегов:
   i. <box> - прямоугольный параллелепипед, атрибут size содержит три числа, отвечающие за ширину, высоту и длину параллелепипеда, соответственно. СК проведена через центр параллелепипеда.
   ii. <cylinder> - цилиндр, определяется атрибутами «radius» и «length» - радиус и высота, соответственно. СК проведена через центр цилиндра.
   iii. <sphere> - сфера, определяется атрибутом «radius», соответствующем радиусу сферы. СК проведена через центр сферы.
   iv. <mesh> - триагональный меш(trimesh), т.е. описание трёхмерного объекта при помощи треугольников-полигонов. Описание треугольников хранится в файле, путь к которому указывается в атрибуте «filename». Наиболее подходящий формат - это формат Collada *.dae, хотя поддерживается также и формат *.stl.

3 <material> - описание материала визуального элемента(цвет и текстура)
   i. <color> - rgba цвет материала, заданного набором из четырех чисел, представляющих красный / зеленый / синий / альфа, каждое из которых находится в диапазоне от [0,1].
   ii. <texture> - текстура материала определяется именем файла (атрибут «filename»)

1 <collision> - тег описывающий свойства звена с точки зрения столкновений(коллизий). Чаще всего описание коллизий не совпадает с описанием визуального отображения. Это связано с тем, что визуальное описание должно соответствовать максимальной реалистичности. Проверка на коллизии гораздо более ресурсоёмкая операция, поэтому зачастую сложную визуальную геометрию упрощают: уменьшают количество полигонов, делают объекты выпуклыми.

2 <origin> - преобразование системы координат для перехода в СК описываемого геометрического объекта. По умолчанию представляет собой нулевое смещение и нулевые повороты по углам Эйлера. Иначе говоря, однородная матрица преобразования является единичной, не изменяющей СК. У этого тега два атрибута:

   i. xyz - смещение, по умолчанию - нулевой вектор
   ii. rpy - углы Эйлера в радианах: крен, тангаж и рыскание

3 <geometry> - этот тег описывает внешний вид звена, может содержать один из следующих тегов:

   i. <box> - прямоугольный параллелепипед, атрибут size содержит три числа, отвечающие за ширину, высоту и длину параллелепипеда, соответственно. СК проведена через центр параллелепипеда.
   ii. <cylinder> - цилиндр, определяется атрибутами «radius» и «length» - радиус и высота, соответственно. СК проведена через центр цилиндра.
   iii. <sphere> - сфера, определяется атрибутом «radius», соответствующем радиусу сферы. СК проведена через центр сферы.
   iv. <mesh> - триагональный меш(trimesh), т.е. описание трёхмерного объекта при помощи треугольников-полигонов. Описание треугольников хранится в файле, путь к которому указывается в атрибуте «filename». Наиболее подходящий формат - это формат Collada *.dae, хотя поддерживается также и формат *.stl.

2.4 Сочленения робота

После того, как все звенья заданы, необходимо определить их сочленения(joints). Каждое сочленение определяет связь двух звеньев. Причём такая связь имееет иерархическую направленность: у каждого сочленения определены родительское(parent) и дочернее(child) звенья. В логике рассматриваемого симулятора для каждой сцены возможно определить только один корень, т.е. главный элемент иерархического дерева. Иными словами, только одно звено может не иметь родительского звена, определённого через сочленения.

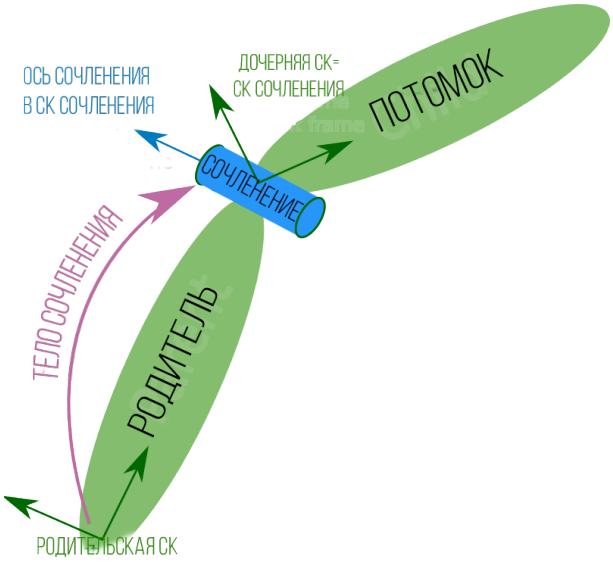

Рисунок 2 – Звено робота

Каждое сочленение задаётся при помощи парного тега <joint>...</joint>. У данного тега есть два аргумента: «name» - имя сочленения и «type» - описывает тип сочленения:

а revolute - шарнирное соединение, которое вращается вдоль оси и имеет ограниченный диапазон, указанный верхним и нижним пределами.

б continuous - непрерывный шарнирное соединение, которое вращается вокруг оси и не имеет верхних и нижних пределов

в prismatic - призматическое соединение, которое скользит вдоль оси и имеет ограниченный диапазон, указанный верхним и нижним пределами.

г fixed - фиксированная связь, т.е. неподвижное звено (все степени

свободы заблокированы). Этот тип соединения не требует оси, калибровки, динамики и пределов.

д floating - плавающее сочленение, позволяет изменяться по всем шести степеням свободы.

е planar - плоское соединение, позволяет двигаться в плоскости, перпендикулярной оси.

Для каждого сочленения задаются следующие параметры:

а <origin> - преобразование системы координат для перехода в СК описываемого сочленения. По умолчанию представляет собой нулевое смещение и нулевые повороты по углам Эйлера. Иначе говоря, однородная матрица преобразования является единичной, не изменяющей СК. У этого тега два атрибута:

1 xyz - смещение, по умолчанию - нулевой вектор

2 rpy - углы Эйлера в радианах: крен, тангаж и рыскание

б <parent> - родительское звено с обязательным атрибутом «link», определяющем имя родительского звена

в <child> - дочернее звено с обязательным атрибутом «link», определяющем имя дочернего звена

г <axis> - ось сочленения. Это ось вращения для вращательных сочленений, ось трансляции - для призматических и нормальная поверхность для плоского движения. Ось указана в системе координат сочленения. Фиксированные и плавающие соединения не используют этот параметр. В качестве аргумента «xyz» передаётся вектор единичной длины.

д <dynamic> - элемент, определяющий демпфирующие свойства сустава: сухое и вязкое трения. В качестве аргументов передаются следующие два аргумента:

1 damping - коэффициент вязкого трения сочленения ($\frac{N \cdot s}{m}$ для призматических, $\frac{N \cdot m \cdot s}{rad}$ - для вращательных).

2 friction - коэффициент сухого трения сочленения (N для призматических, $N \cdot m$ для вращательных)

е <limit> (требуется только для вращательных и призматических сочленений) - ограничения на сочленения. Элемент может содержать следующие атрибуты:

ж lower - минимальное значение положения сочленения (угла поворота - для вращательных(радианы), положения - для призматических суставов(метры))

з upper - максимальное значение положения сочленения (угла поворота - для вращательных(радианы), положения - для призматических суставов(метры))

и effort - максимальный модуль усилия, развиваемого в сочленении(сила для призматических, момент для вращательных)

к velocity - максимальный модуль скорости сочленения (поступательная для призматических, угловая для вращательных)

2.5 Трансмиссии

После того, как определены звенья и сочленения, необходимо описать двигатели, приводящие сочленения, а вместе с ними звенья. Это делается при помощи трансмиссий. Для каждого нефиксированного звена можно определить привод, в концепции симулятора Gazebo они называются «трансмиссиями».

Для каждой трансмиссии можно определить следующие элементы(вложенные теги):

а <type> - указывает тип трансмиссии

б <joint> - соединение, к которому подключена трансмиссия. Сочленение определяется его атрибутом «name» и подэлементами <hardwareInterface> - он указывает поддерживаемый аппаратный интерфейс на совместном пространстве. Есть три основных интерфейса:

1 hardware_interface/EffortJointInterface - интерфейс управления по усилию

2 hardware_interface/VelocityJointInterface - интерфейс управления по скорости

3 hardware_interface/PositionJointInterface - интерфейс управления по положению

Причём каждый интерфейс учитывает динамические и кинематические свойства описанного робота.

в <actuator> - привод, к которому подключена передача. Привод определяется его атрибутом «name» и следующим подэлемен-

том: <mechanReduction> - определяет передаточное соотношение между приводом и сочленением.

2.6 Контроллеры

Для управления виртуальным роботом, необходимо при помощи ROS определить механизмы управления указанными трансмиссиями. Это делается при помощи механизма тем(topic).

Темы - это именованные шины, по которым узлы ROS обмениваются сообщениями. Темы имеют анонимную семантику публикация/подписка, которая отделяет механизмы, как в эту шину информация помещается и как считывается. Узлы не знают, с кем они общаются, а только помещают или извлекают информацию из шины. Вместо этого узлы, заинтересованные в данных, подписываются на соответствующую тему(topic), а узлы, которые генерируют данные, публикуются в соответствующей теме. В теме могут быть несколько издателей и несколько подписчиков.

Соответственно, в зависимости от реализуемого интерфейса в трансмиссии необходимо создать профильный топик и подписаться на него. Как только в него будет записываться новая информация, необходимо передавать её трансмиссии, которая в свою очередь будет перемещать сочленения робота.

За такого рода механизмы отвечают специальные инструменты ROS - контроллеры [1].

Контроллер - это специальный сервис, который позволяет связывать трансмиссии Gazebo или реальные физические устройства с темами ROS.

В рамках данной магистерской диссертации будет рассмотрено управление мобильным роботом Kuka Youbot, представленными на Рисунке 3. Он состоит из мобильной четырёхколёсной базы с всенаправленными(omni) колёсами и пятизвенного робота-манипулятора.

При помощи контроллеров можно реализовать как управление сочленениями роботов, так и измерение показаний виртуальных датчиков.

Т.к. в составе мобильного робота есть пятизвенный манипулятор, то нам необходимо реализовать следующие три типа контроллеров управления:

a effort_controllers/JointEffortController - контроллер управле-

Рисунок 3 – Робот Кука Юбот

ния по усилию, он использует интерфейс трансмиссии hardware_interface/EffortJointInterface

б velocity_controllers/JointVelocityController - контроллер управления по скорости, он использует интерфейс трансмиссии hardware_interface/VelocityJointInterface

в position_controllers/JointPositionController - контроллер управления по положению, он использует интерфейс трансмиссии hardware_interface/PositionJointInterface

. Каждый контроллер сочленения подписывается на тему и обрабатывает управление, которое поступает извне.

Также необходим один контроллер измерений joint_state_controller/JointStateController. Этот контроллер является издателем и публикует в соответствующую тему данные о каждом из сочленений: текущее положение, скорость и усилие.

Kuka Youbot - мобильный робот, поэтому помимо описанных в предыдущем разделе контроллеров, управляющих пятизвенным мани-

пулятором, необходимо реализовать интерфейс управления мобильной базой и определения текущего положения.

Для этой цели используется контроллер управления мобильной платформой из плагина libgazebo_ros_planar_move. Этот контроллер позволяет управлять плоским движением вдоль горизонтальной поверхности за счёт управления поступательной скоростью тележки, а также скорости вращения вокруг центра масс. В качестве датчиков положения реализована виртуальная одометрия, возвращающая текущее положение центра тележки, а также её угол поворота

2.7 Результат

2.7.1 Симуляция

При помощи симулятора Gazebo была создана динамическая модель Kuka Youbot 4

(a) (b)

Рисунок 4 – Симуляция Кука Юбот:
(a)Реальный робот; (b) Виртуальный;

2.7.2 Управление

Реализованный первый контур системы управления роботом Kuka Youbot представлен на Рисунке 5

В указанной системе есть три контура управления манипулятором:

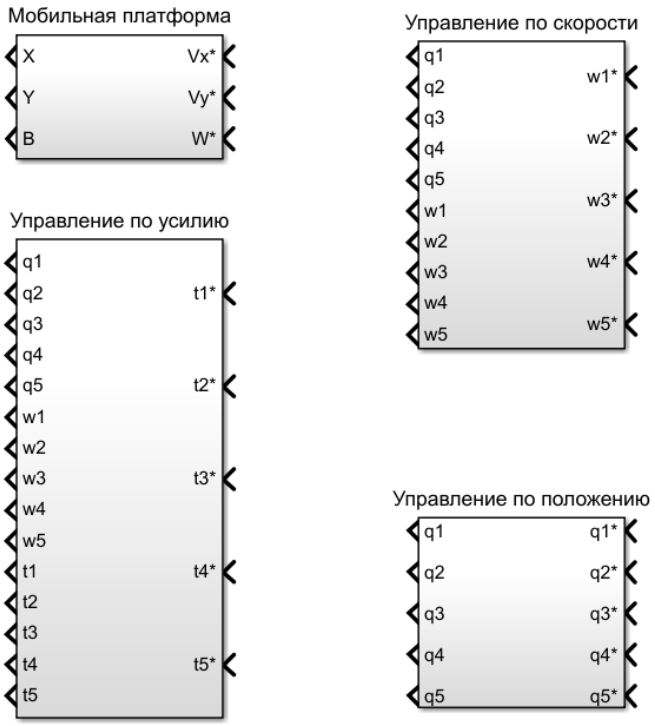

Рисунок 5 – Контуры управления роботом Кука Юбот

а Управление по положению. В этом режиме на вход системе управления подаются задания по положению для набора звеньев q_i^* (необязательно для всех), на выходе системы в режиме реального времени возвращаются реальные углы всех сочленений робота.

б Управление по скорости. В этом режиме на вход системе управления подаются задания по скорости для набора звена q_i^* (необязательно для всех), на выходе системы в режиме реального времени возвращаются реальные углы всех сочленений робота, а также скорости.

в Управление по усилию. В этом режиме на вход системе управления подаются задания по усилию для набора звена q_i^* (необязательно для всех), на выходе системы в режиме реального времени возвращаются реальные углы всех сочленений робота, а также скорости и усилия.

Также присутствует контур управления мобильной платформой.

На вход системе подаются задания по поступательной и угловой скоростям, на выходе в режиме реального времени возвращаются положение и ориентация мобильной платформы.

3 Кинематика

Для управления роботом в первую очередь необходимо построить кинематическую модель. Для этого разделим задачу на две: кинематическая модель колёсной базы и кинематическая модель робота-манипулятора.

3.1 Кинематическая модель манипулятора

Манипулятор состоит из пяти вращательных сочленений. Кинематическая схема представлена на Рисунке ??. Определим параметры Денавита-Хартенберга [2]. Эти параметры позволяют однозначно определить робота. Переход из СК предыдущего звена в текущую можно описать четырьмя параметрами:

- d - смещение вдоль оси z СК предыдущего звена
- θ - угол поворота вдоль оси z СК предыдущего звена для совмещения x координат предыдущей СК и текущей
- a - смещение вдоль оси x СК нового звена
- α - угол поворота вдоль оси x СК следующего звена для совмещения z координат предыдущей СК и текущей

Иными словами, необходимо определить как сочленение связывает системы координат родительского и дочернего звеньев. У роботов это можно сделать при помощи двух смещений и двух поворотов.

Любой промышленный робот имеет как минимум две системы координат:

- Мировые координаты включают в себя три измерения, отвечающих за положение рабочего инструмента(энд-эффектора), и три измерения, отвечающих за ориентацию инструмента в пространстве. Чаще всего используют углы Эйлера [3]. Углы Эйлера представляют из себя три угла, определяющие последовательные повороты тела вдоль трёх осей. В зависимости от выбора осей меняется нотация. Соответствие углов Эйлера и их матриц:

  а $X_1Z_2X_3 = \begin{bmatrix} c_2 & -c_3s_2 & s_2s_3 \\ c_1s_2 & c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 \\ s_1s_2 & c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 \end{bmatrix}$

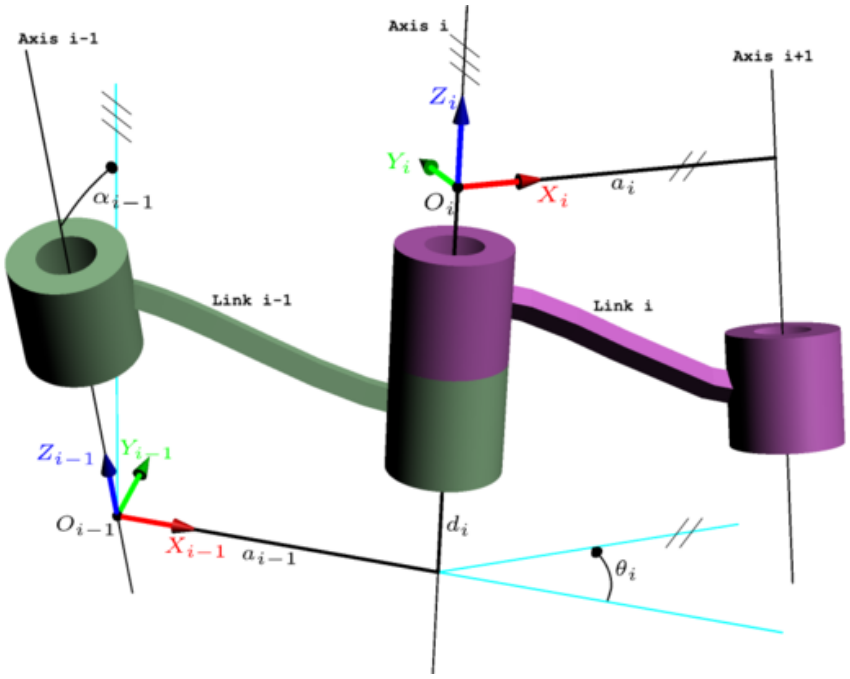

Рисунок 6 – Параметры Денавита-Хартенберга

б $X_1Z_2Y_3 = \begin{bmatrix} c_2c_3 & -s_2 & c_2s_3 \\ s_1s_3 + c_1c_3s_2 & c_1c_2 & c_1s_2s_3 - c_3s_1 \\ c_3s_1s_2 - c_1s_3 & c_2s_1 & c_1c_3 + s_1s_2s_3 \end{bmatrix}$

в $X_1Z_2Y_3 = \begin{bmatrix} c_2c_3 & -s_2 & c_2s_3 \\ s_1s_3 + c_1c_3s_2 & c_1c_2 & c_1s_2s_3 - c_3s_1 \\ c_3s_1s_2 - c_1s_3 & c_2s_1 & c_1c_3 + s_1s_2s_3 \end{bmatrix}$

г $X_1Y_2X_3 = \begin{bmatrix} c_2 & s_2s_3 & c_3s_2 \\ s_1s_2 & c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 \\ -c_1s_2 & c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 \end{bmatrix}$

д $X_1Y_2X_3 = \begin{bmatrix} c_2 & s_2s_3 & c_3s_2 \\ s_1s_2 & c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 \\ -c_1s_2 & c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 \end{bmatrix}$

е $X_1Y_2Z_3 = \begin{bmatrix} c_2c_3 & -c_2s_3 & s_2 \\ c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 & -c_2s_1 \\ s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 & c_1c_2 \end{bmatrix}$

ж $X_1Y_2Z_3 = \begin{bmatrix} c_2c_3 & -c_2s_3 & s_2 \\ c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 & -c_2s_1 \\ s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 & c_1c_2 \end{bmatrix}$

з $Y_1X_2Y_3 = \begin{bmatrix} c_1c_3 - c_2s_1s_3 & s_1s_2 & c_1s_3 + c_2c_3s_1 \\ s_2s_3 & c_2 & -c_3s_2 \\ -c_3s_1 - c_1c_2s_3 & c_1s_2 & c_1c_2c_3 - s_1s_3 \end{bmatrix}$

и $Y_1X_2Y_3 = \begin{bmatrix} c_1c_3 - c_2s_1s_3 & s_1s_2 & c_1s_3 + c_2c_3s_1 \\ s_2s_3 & c_2 & -c_3s_2 \\ -c_3s_1 - c_1c_2s_3 & c_1s_2 & c_1c_2c_3 - s_1s_3 \end{bmatrix}$

к $Y_1X_2Z_3 = \begin{bmatrix} c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 & c_2s_1 \\ c_2s_3 & c_2c_3 & -s_2 \\ c_1s_2s_3 - c_3s_1 & c_1c_3s_2 + s_1s_3 & c_1c_2 \end{bmatrix}$

л $Y_1X_2Z_3 = \begin{bmatrix} c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 & c_2s_1 \\ c_2s_3 & c_2c_3 & -s_2 \\ c_1s_2s_3 - c_3s_1 & c_1c_3s_2 + s_1s_3 & c_1c_2 \end{bmatrix}$

м $Y_1Z_2Y_3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_1s_2 & c_3s_1 + c_1c_2s_3 \\ c_3s_2 & c_2 & s_2s_3 \\ -c_1s_3 - c_2c_3s_1 & s_1s_2 & c_1c_3 - c_2s_1s_3 \end{bmatrix}$

н $Y_1Z_2Y_3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_1s_2 & c_3s_1 + c_1c_2s_3 \\ c_3s_2 & c_2 & s_2s_3 \\ -c_1s_3 - c_2c_3s_1 & s_1s_2 & c_1c_3 - c_2s_1s_3 \end{bmatrix}$

о $Y_1Z_2X_3 = \begin{bmatrix} c_1c_2 & s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 \\ s_2 & c_2c_3 & -c_2s_3 \\ -c_2s_1 & c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 \end{bmatrix}$

п $Y_1Z_2X_3 = \begin{bmatrix} c_1c_2 & s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 \\ s_2 & c_2c_3 & -c_2s_3 \\ -c_2s_1 & c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 \end{bmatrix}$

р $Z_1Y_2Z_3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 & c_1s_2 \\ c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 & s_1s_2 \\ -c_3s_2 & s_2s_3 & c_2 \end{bmatrix}$

с $Z_1Y_2Z_3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 & c_1s_2 \\ c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 & s_1s_2 \\ -c_3s_2 & s_2s_3 & c_2 \end{bmatrix}$

т $Z_1Y_2X_3 = \begin{bmatrix} c_1c_2 & c_1s_2s_3 - c_3s_1 & s_1s_3 + c_1c_3s_2 \\ c_2s_1 & c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 \\ -s_2 & c_2s_3 & c_2c_3 \end{bmatrix}$

$$\text{у } Z_1Y_2X_3 = \begin{bmatrix} c_1c_2 & c_1s_2s_3 - c_3s_1 & s_1s_3 + c_1c_3s_2 \\ c_2s_1 & c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 \\ -s_2 & c_2s_3 & c_2c_3 \end{bmatrix}$$

$$\text{ф } Z_1X_2Z_3 = \begin{bmatrix} c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 & s_1s_2 \\ c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 & -c_1s_2 \\ s_2s_3 & c_3s_2 & c_2 \end{bmatrix}$$

$$\text{х } Z_1X_2Z_3 = \begin{bmatrix} c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 & s_1s_2 \\ c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 & -c_1s_2 \\ s_2s_3 & c_3s_2 & c_2 \end{bmatrix}$$

$$\text{ц } Z_1X_2Y_3 = \begin{bmatrix} c_1c_3 - s_1s_2s_3 & -c_2s_1 & c_1s_3 + c_3s_1s_2 \\ c_3s_1 + c_1s_2s_3 & c_1c_2 & s_1s_3 - c_1c_3s_2 \\ -c_2s_3 & s_2 & c_2c_3 \end{bmatrix}$$

$$\text{ч } Z_1X_2Y_3 = \begin{bmatrix} c_1c_3 - s_1s_2s_3 & -c_2s_1 & c_1s_3 + c_3s_1s_2 \\ c_3s_1 + c_1s_2s_3 & c_1c_2 & s_1s_3 - c_1c_3s_2 \\ -c_2s_3 & s_2 & c_2c_3 \end{bmatrix},$$

где s_1 - синус угла поворота вдоль первой оси, s_2 - синус угла поворота вдоль второй оси, s_3 - синус угла поворота вдоль третьей оси, c_1 - косинус угла поворота вдоль первой оси, c_2 - косинус угла поворота вдоль второй оси, c_3 - косинус угла поворота вдоль третьей оси,

– Обобщённые координаты представляют собой углы поворота сочлененьев робота. Поэтому размерность этой системы координат совпадает с числом активных сочленений робота.

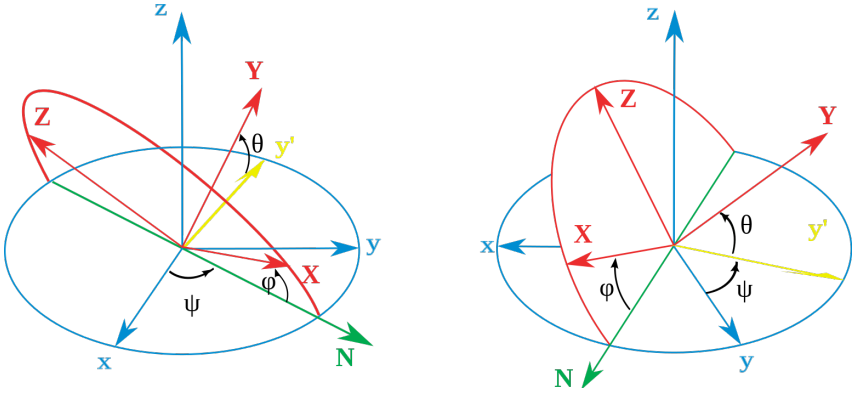

Рисунок 7 – Углы Эйлера

Для каждого из звеньев робота можно ввести локальную систему координат. Тогда для связи двух указанных СК вводится матрица преоб-

разования из мировой системы координат в систему координат рабочего инструмента(последнего звена). пример назначения систем координат робота-манипулятора представлен на Рисунке 8

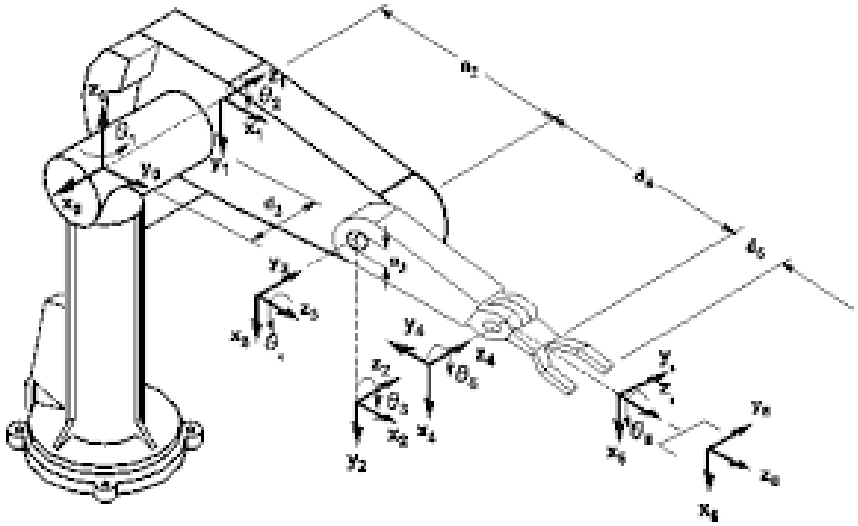

Рисунок 8 – Системы координат звеньев робота

Алгоритм перехода из $n-1$ системы координат в n принято формализовывать при помощи матриц перехода размерности 4x4.

$$^{n-1}T_n = \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right],$$

где R - матрица поворота 3x3, соответствие нотации углов Эйлера и матрицы поворота было приведено выше, T - вектор смещения 3x1,

При умножении матрицы перехода на радиус-вектора 4x1 в системе координат, в которую нужно перейти, можно получить радиус-вектор точки в исходной системе координат. Такой радиус вектор состоит из трёх координат и единицы: $v = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$

Т.к. параметры Денавита Хартенберга описывают переход из системы координат родительского звена в дочернюю, то для каждого перехода между системами координат можно установить связь между матрицами перехода и параметрами $\theta_n, d_n, a_n, \alpha_n$ [2]:

$$
{}^{n-1}T_n = \left[\begin{array}{ccc|c}
\cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & a_n\cos\theta_n \\
\sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & a_n\sin\theta_n \\
0 & \sin\alpha_n & \cos\alpha_n & d_n \\
\hline
0 & 0 & 0 & 1
\end{array}\right] \qquad (2)
$$

В инженерной практике распространён подход определения параметров Денавита-Хартенберга в положении "свечи т.е. положение, в котором рабочий инструмент робота находится на максимальной высоте. Робот в положении свечи представлен на Рисунке 9

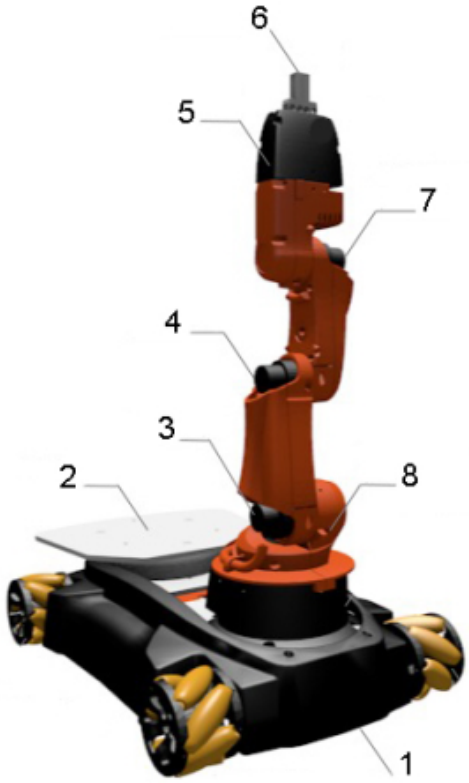

Рисунок 9 – Робот в положении свечи
1-колёсная база, 2-рабочая площадка, 3 - второе звено робота, 4-третье звено робота, 5- пятое звено робота, 6 - рабочий инструмент робота, 7 - четвёртое звено робота, 8 - первое звено робота

Ему соответствует кинематическая схема, представленная на Рисунке 10

Параметры Денавита-Хартенберга для положения "свечи"представлены в Таблице 1

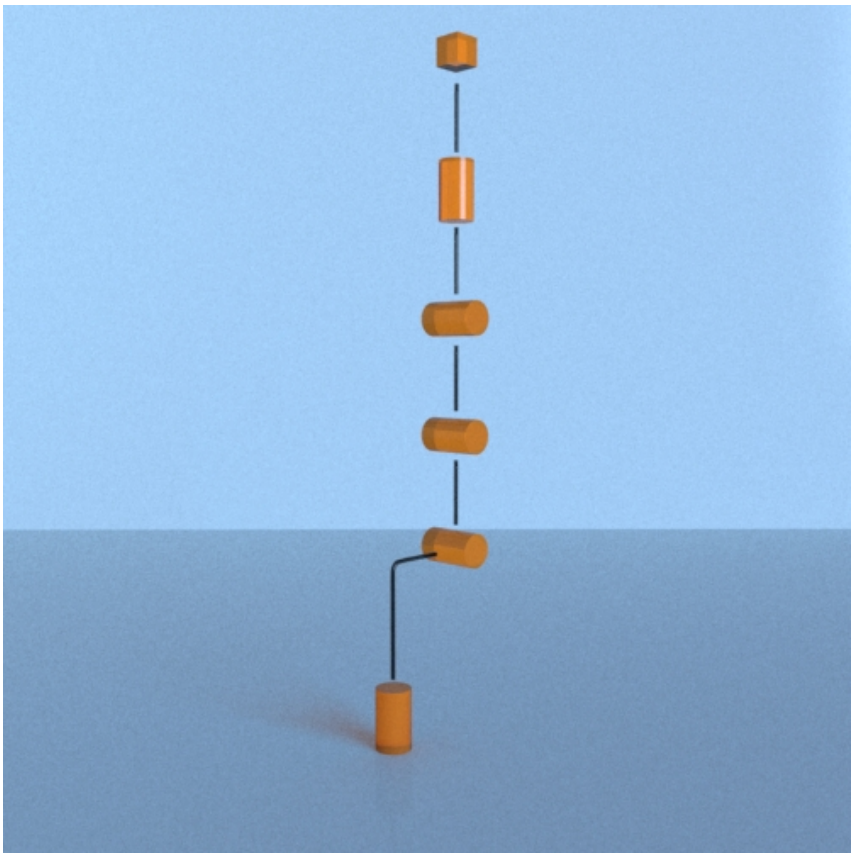

Рисунок 10 – Кинематическая схема робота в положении свечи

Таблица 1 - Параметры Денавита-Хартенберга для положения свечи

№ звена	θ	d	a	α
1	θ_1	147.0	33.0	90.0°
2	θ_2	0.0	155.0	0.0°
3	θ_3	0.0	135.0	0.0°
4	$\theta_4 + 90°$	0.0	0.0	90.0°
5	θ_5	211.0	0.0	0.0°

θ_i - угол поворота i-го звена

Однако такое положение не соответствует нулевым углам поворота звеньев. Нулевые углы поворота звеньев соответствует конфигурации робота представленной на Рисунке 11

Ему соответствует кинематическая схема, представленная на Ри-

Рисунок 11 – Конфигурация робота при нулевых углах поворота звеньев

сунке 12

Параметры Денавита-Хартенберга для смещённой СК представлены в Таблице 2

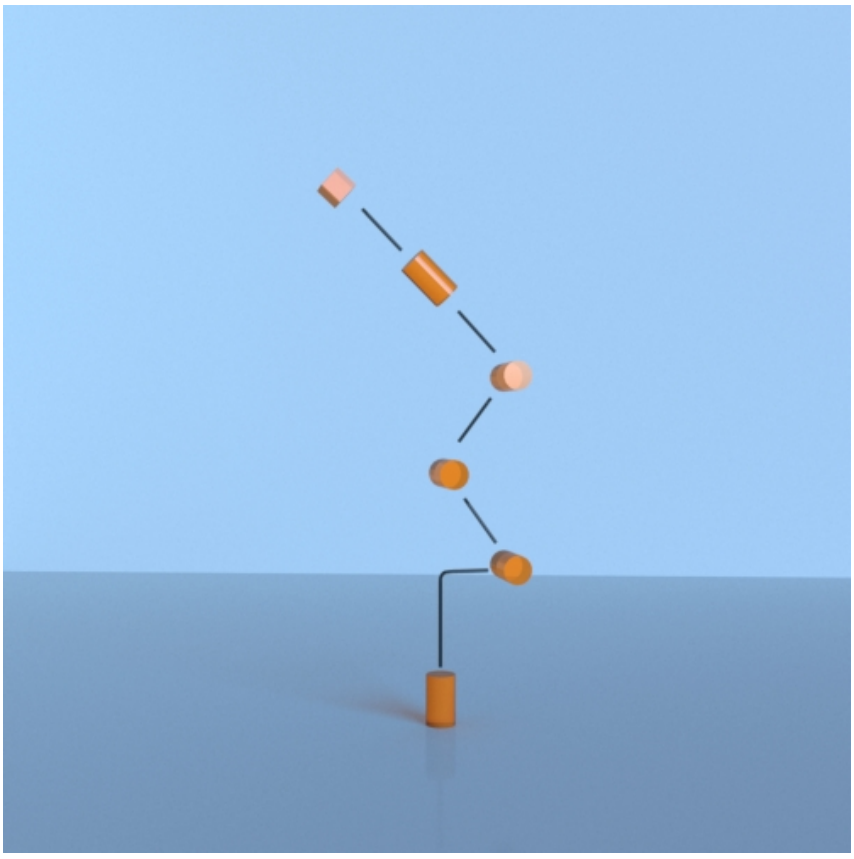

Рисунок 12 – Кинематическая схема робота при нулевых углах поворота звеньев

Таблица 2 - Параметры Денавита-Хартенберга для положения свечи

№ звена	θ	d	a	α
1	$\theta_1 + 170.0°$	147.0	33.0	90.0°
2	$\theta_2 + 25.0°$	0.0	155.0	0.0°
3	$\theta_3 + 146.0°$	0.0	135.0	0.0°
4	$\theta_4 - 12.5°$	0.0	0.0	90.0°
5	$\theta_5 + 12.5°$	211.0	0.0	0.0°

θ_i - угол поворота i-го звена

Тогда, измеряя углы в радианах, матрицы преобразования можно

записать в виде:

$$ {}^0T_1 = \begin{pmatrix} -\sin(q_1+1.4) & 0 & \cos(q_1+1.4) & -33.0\sin(q_1+1.4) \\ \cos(q_1+1.4) & 0 & \sin(q_1+1.4) & 33.0\cos(q_1+1.4) \\ 0 & 1.0 & 0 & 166.0 \\ 0 & 0 & 0 & 1.0 \end{pmatrix} \quad (3) $$

$$ {}^1T_2 = \begin{pmatrix} \cos(q_2+0.44) & -\sin(q_2+0.44) & 0 & 177.0\cos(q_2+0.44) \\ \sin(q_2+0.44) & \cos(q_2+0.44) & 0 & 177.0\sin(q_2+0.44) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{pmatrix} \quad (4) $$

$$ {}^2T_3 = \begin{pmatrix} -\sin(q_3+0.98) & -\cos(q_3+0.98) & 0 & -144.0\sin(q_3+0.98) \\ \cos(q_3+0.98) & -\sin(q_3+0.98) & 0 & 144.0\cos(q_3+0.98) \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{pmatrix} \quad (5) $$

$$ {}^3T_4 = \begin{pmatrix} \cos(q_4-0.22) & 0 & \sin(q_4-0.22) & 0 \\ \sin(q_4-0.22) & 0 & -\cos(q_4-0.22) & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 \end{pmatrix} \quad (6) $$

$$ {}^4T_5 = \begin{pmatrix} \cos(q_5+0.22) & -\sin(q_5+0.22) & 0 & 0 \\ \sin(q_5+0.22) & \cos(q_5+0.22) & 0 & 0 \\ 0 & 0 & 1.0 & 211.0 \\ 0 & 0 & 0 & 1.0 \end{pmatrix} \quad (7) $$

Тогда матрицу преобразования из мировой системы координат(так называют систему координат, связанную с основанием робота) в систему координат рабочего инструмента можно записать как [4]:

$$ {}^0T_5 = {}^0T_1 \cdot {}^1T_2 \cdot {}^2T_3 \cdot {}^3T_4 \cdot {}^4T_5 \quad (8) $$

3.2 Кинематическая модель колёсной базы

Примем произвольную точку плоскости, по которой перемещается колёсная база робота за начало системы отсчёта. Систему координат тележки определим следующим образом: введём правую систему координат, ось x которой направлена вперёд по отношению к колёсной базе, ось z вверх. Совместим начало мировой системы координат с началом выбранной системы отсчёта. Тогда положение тележки определяется радиус-вектором её центра R и углом поворота a.

Тогда матрица перехода из тела отсчёта в систему координат колёсной базы будет иметь вид [5]:

$$T_b = \begin{pmatrix} \cos(a) & -\sin(a) & 0 & R_x \\ \sin(a) & \cos(a) & 0 & R_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{9}$$

Манипулятор робота Kuka Youbot закреплён не в центре робота, центр основания робота смещён относительно центра колёсной базы на вектор $v_a = \begin{bmatrix} 190 & 0 & 60 \end{bmatrix}^T$

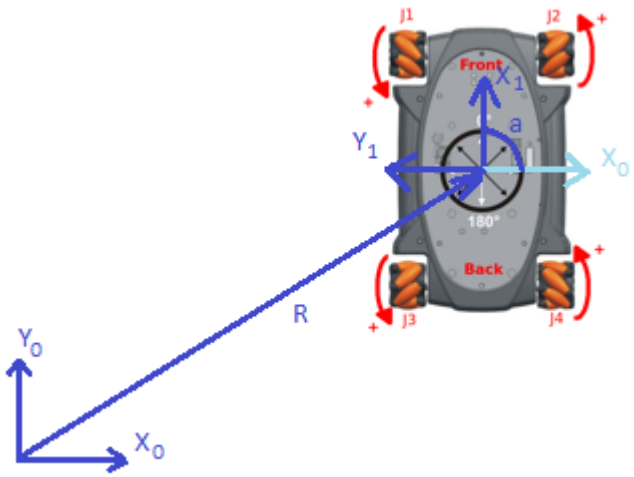

Рисунок 13 – Кинематика колёсной базы

Тогда матрица перехода из системы координат колёсной базы в систему координат основания робота определяется следущим образом:

$$T_a = \begin{pmatrix} 1 & 0 & 0 & 190 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 60 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{10}$$

Тогда матрица перехода из мировой системы координат в систему координат рабочего инструмента с учётом 8 следующим образом:

$$T = T_b \cdot T_a \cdot^0 T_5 = T_b \cdot T_a \cdot^0 T_1 \cdot^1 T_2 \cdot^2 T_3 \cdot^3 T_4 \cdot^4 T_5 \tag{11}$$

4 Прямая задача кинематики

Прямая задача кинематики подразумевает поиск функциональной зависимости обощённых и мировых координат рабочего инструмента. Т.к. помимо рабочего инструмента, управление возможно также и колёсной базой, то вектор обощённых координат принимает вид:

$$V = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ R_x \\ R_y \\ a \end{pmatrix} \tag{12}$$

Положение можно получить из матрицы перехода T напрямую. За смещение отвечают первые три элемента четвёртого столбца матрицы, поэтому выражение для положения имеет вид:

$$\begin{pmatrix} x(V) \\ y(V) \\ z(V) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot T(V) \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{13}$$

Подход к нахождению углов Эйлера детально рассмотрен в статье [6]. Выберем XYZ-нотацию углов Эйлера, т.е.

первый поворот выполняется на угол ψ вдоль оси x и определён как:

$$R_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{pmatrix}$$

второй поворот выполняется на угол θ вдоль оси y и определён как:

$$R_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

третий поворот выполняется на угол ϕ вдоль оси z и определён как:

$$R_z(\phi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Тогда матрица поворота R_{xyz} или поворотная компонента матрицы перехода можно представить в виде:

$$R_{xyz} = R_z(\psi)R_y(\theta)R_x(\phi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

$$R_{xyz} = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \quad (14)$$

Для нахождения угла θ, воспользуемся элементом R_{13} матрицы поворота R_{xyz}:

$$R_{31} = -\sin(\theta) \quad (15)$$

Тогда угол может быть равен одному из двух значений:

$$\begin{cases} \theta_1 = -\sin^{-1}(R_{31}) \\ \theta_2 = \pi - \sin^{-1}(R_{31}) \end{cases}, \quad (16)$$

т.к. функция арксинуса определена на отрезке, длиной π

Чтобы найти значение угла ψ, воспользуемся элементами R_{32} и R_{33} матрицы поворота R_{xyz}:

$$\frac{R_{32}}{R_{33}} = \tan\psi \quad (17)$$

$$\psi = \text{atan2}(R_{32}, R_{33}), \quad (18)$$

где $\text{atan2}(y, x)$ - функция двух переменных x и y, возвращающая угол между осью x и радиус-вектором, проведённым из начала системы координат, это похоже на классический арктангенс. У классического арктангенса множество значений $\left[-\frac{\pi}{2}; \frac{\pi}{2}\right]$, аргументом является отношение x/y, что позволяет определить по тангенсу угол в диапазоне длиной π, что не позволяет однозначно восстановить угол радиус-вектора по координатам, поэтому была введена функция atan2 [7], имеющая множеством значений диапазон $\left[-\pi; \pi\right]$

Необходимо также учесть. что в зависимости от знака θ будут меняться аргументы в 18, поэтому полное выражение для ψ имеет вид:

$$\psi = \text{atan2}(\frac{R_{32}}{\cos\theta}, \frac{R_{33}}{\cos\theta})), \tag{19}$$

Но такой подход порождает проблему решения при $\theta = 0$, этот случай выделяется как особый и разбирается отдельно.

$$\begin{cases} \psi_1 = \text{atan2}(\frac{R_{32}}{\cos\theta_1}, \frac{R_{33}}{\cos\theta_1})) \\ \psi_2 = \text{atan2}(\frac{R_{32}}{\cos\theta_2}, \frac{R_{33}}{\cos\theta_2})) \end{cases} \tag{20}$$

Для нахождения ϕ воспользуемся подходом, раскрытым при нахождении ψ. Для нахождения ψ необходимо рассмотреть элементы R_{21} и R_{11} матрицы поворота R_{xyz}.

$$\frac{R_{21}}{R_{11}} = \tan\phi \tag{21}$$

$$\phi = \text{atan2}(R_{21}, R_{11}), \tag{22}$$

следует отметить, что при вычислении угла ϕ также как и в случае вычисления угла ψ есть два варианта значений, а также особый случай $\theta = 0$:

$$\begin{cases} \phi_1 = \text{atan2}(\frac{R_{21}}{\cos\theta_1}, \frac{R_{11}}{\cos\theta_1})) \\ \phi_2 = \text{atan2}(\frac{R_{21}}{\cos\theta_2}, \frac{R_{11}}{\cos\theta_2})) \end{cases} \tag{23}$$

Резюмируя, при $\cos(\theta)$ не равном нулю, есть два решения:

$$\begin{pmatrix} \theta \\ \psi \\ \phi \end{pmatrix} = \left\{ \begin{pmatrix} \theta_1 \\ \psi_1 \\ \phi_1 \end{pmatrix} \begin{pmatrix} \theta_2 \\ \psi_2 \\ \phi_2 \end{pmatrix} \right. \tag{24}$$

Рассмотрим теперь особый случай при $\cos\theta = 0$:

При $\theta = \frac{\pi}{2}$:

$$\psi - \phi = atan2(R_{12}, R_{13})$$

$$\psi = \phi + atan2(R_{12}, R_{13}) \quad (25)$$

При $\theta = -\frac{\pi}{2}$:

$$\psi + \phi = atan2(R_{12}, R_{13})$$

$$\psi = -\phi + atan2(R_{12}, R_{13}) \quad (26)$$

Таким образом, ψ и ϕ - взаимосвязанные величины и при $\cos(\theta) = 0$ существует бесконечное множество решений. Этот эффект называется "эффектом складывания рамок". [8]

4.1 Кватернионы

В инженерной практике принято определять ориентацию рабочего инструмента через углы Эйлера, это связано с историческим развитие теоретической механики и исторически сложившейся сферы применения(гироскопия, авионика). Использование углов Эйлера в управлении роботами является неэффективным инструментом в силу причин, описанных выше. Поэтому в рамках данной магистерской работы ориентация рабочего инструмента будет задаваться при помощи параметров Родрига-Гамильтона [10].

Пусть ориентация в пространстве задана относительно, т.е. задан единичный вектор, вокруг которого происходит вращение и угол, на который это вращение производится. Тогда такую структуру можно записать как гиперкомплексный объект - кватернион:

$$q = a + bi + cj + dk, \quad (27)$$

где i,j,k — мнимые единицы, удовлетворяющие следующему равенству:

$$i^2 = j^2 = k^2 = ijk = -1, \quad (28)$$

Также действует правило перемножения мнимых единиц (Рисунок 14)

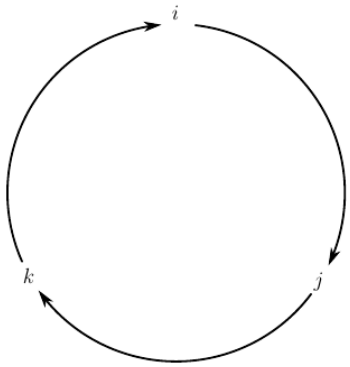

Рисунок 14 – Мнимые единицы гиперкомплексного числа

$$\begin{cases} i \cdot j = -j \cdot i = k \\ j \cdot k = -k \cdot j = i \\ k \cdot i = -i \cdot k = j \end{cases} \tag{29}$$

Тогда можно ввести композицию поворотов как композицию кватернионов [11]. Т.е. если первый поворот r_1 описывает кватернион $q_1 = a_1 + b_1 i + c_1 j + d_1 k$, а второй r_2 описывает кватернион $q_2 = a_2 + b_2 i + c_2 j + d_2 k$, тогда поворот r_3, соответствующий композиции поворотов: сначала поворот r_1, потом r_2 будет описываться кватернионом $q_3 = a_3 + b_3 i + c_3 j + d_3$:

$$\begin{aligned} a_3 &= a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2; \\ b_3 &= a_1 b_2 + b_1 a_2 - c_1 d_2 + d_1 c_2; \\ c_3 &= a_1 c_2 + c_1 a_2 - d_1 b_2 + b_1 d_2; \\ d_3 &= a_1 d_2 + d_1 a_2 - b_1 c_2 + c_1 b_2. \end{aligned} \tag{30}$$

Кватернион вращения полагается единичным, т.е.

$$\|q\|^2 = a^2 + b^2 + c^2 + d^2 = 1. \tag{31}$$

Стоит также отметить, что кватернионы (a, b, c, d) и $(-a, -b, -c, -d)$ описывают одно и тоже вращение.

Пусть поворот выполняется вокруг вектора $k = (k_x, k_y, k_z)$ на угол ϕ, тогда координаты кватерниона можно выразить через параметры Родрига-Гамильтона [10]:

$$\begin{aligned} a &= \cos\frac{\varphi}{2};\\ b &= k_x \sin\frac{\varphi}{2};\\ c &= k_y \sin\frac{\varphi}{2};\\ d &= k_z \sin\frac{\varphi}{2}. \end{aligned} \tag{32}$$

Тогда матрица преобразования имеет вид:

$$R = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2(bc-ad) & 2(bd+ac) \\ 2(bc+ad) & a^2+c^2-b^2-d^2 & 2(cd-ab) \\ 2(bd-ac) & 2(cd+ab) & a^2+d^2-b^2-c^2 \end{pmatrix} \tag{33}$$

Для параметризации поворота в статье [9] раскрыт способ перейти к описанию конечного поворота при помощи трёх параметров. Для этого введём вектор $\boldsymbol{v} = (b, c, d)$, тогда в силу 31:

$$a = \sqrt{1-b^2-c^2-d^2} = \sqrt{1-\boldsymbol{v}^2} \tag{34}$$

Тогда матрицу поворота можно переписать в следующем виде:

$$R = \begin{pmatrix} 1-2c^2-2d^2 & 2(bc-\sqrt{1-\boldsymbol{v}^2}d) & 2(bd+\sqrt{1-\boldsymbol{v}^2}c) \\ 2(bc+\sqrt{1-\boldsymbol{v}^2}d) & 1-2c^2-2d^2 & 2(cd-\sqrt{1-\boldsymbol{v}^2}b) \\ 2(bd-\sqrt{1-\boldsymbol{v}^2}c) & 2(cd+\sqrt{1-\boldsymbol{v}^2}b) & 1-2c^2-2d^2 \end{pmatrix} \tag{35}$$

Тогда

$$\mathrm{Tr}R = 1+\cos\varphi = 3-4\boldsymbol{v}^2 \tag{36}$$

$$1+\mathrm{Tr} = 4(1-\boldsymbol{v}^2) \tag{37}$$

Найдём кососимметричную компоненту матрицы поворота:

$$(R-R^T)_{ij} = 4\sqrt{1-\boldsymbol{v}^2}e^{ikj}v_k, \tag{38}$$

где e^{ijk} - символ Леви Чивиты [12], который определяется следу-

ющим образом:

$$e_{ijk} = \begin{cases} +1 & \text{if } (i,j,k) \text{ is } (1,2,3),(2,3,1), \text{ or } (3,1,2), \\ -1 & \text{if } (i,j,k) \text{ is } (3,2,1),(1,3,2), \text{ or } (2,1,3), \\ 0 & \text{if } i=j, \text{ or } j=k, \text{ or } k=i \end{cases} \tag{39}$$

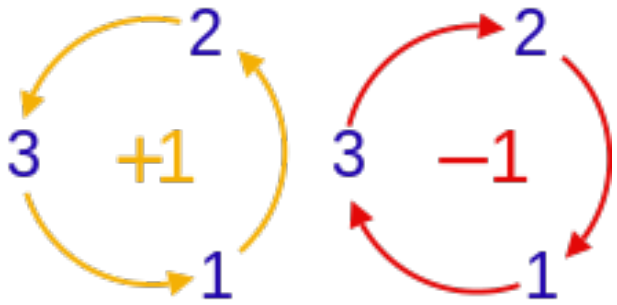

Рисунок 15 – Расчёт символа Леви-Чивиты

Если все индексы различные, то знак можно определить по направлению обхода 15. Обход по часовой стрелке соответствует значению -1, направлению против часовой стрелки - значение 1.

Объединив уравнения 37 и 38 получим:

$$v_k = \frac{(R - R^T)_{ij} e^{ikj}}{4\sqrt{1 + \mathrm{Tr}R}} \tag{40}$$

Уравнение 40 следует читать согласно соглашению Эйнштейна [13]. Согласно нему, суммирование происходит по повторяющимся разноуровневым индексам. В явном виде уравнение 40 записывается следующим образом:

$$v_n = \sum_{i,j} \frac{(R - R^T)_{ij} e^{inj}}{4\sqrt{1 + \mathrm{Tr}R}} \tag{41}$$

Остаётся два особых случая: $\boldsymbol{v}^2 = 1$, т.е. первая компонента кватерниона, отвечающая за поворот$(a = \cos\frac{\varphi}{2})$ равна 0, т.е. поворот является нулевым. Поэтому матрица поворота, соответствующая ему должна быть единичной.

Если же т.е. первая компонента кватерниона, отвечающая за поворот по модулю нулю. Такому случаю соответствует поворот на 180° вдоль произвольной оси, а кватернион определяется как $(\pm 1, 0, 0, 0)$.

Вернёмся к рассмотрению общего случая.

Для получения параметров Родрига из уравнения 32, необходимо найти вектор k:

$$\boldsymbol{v} = k \sin \frac{\varphi}{2} \tag{42}$$

Т.к. это равенство векторное, а k - единичный вектор, то

$$|\boldsymbol{v}| = \sin \frac{\varphi}{2} \tag{43}$$

$$k = \begin{pmatrix} \frac{(R-R^T)_{ij} e^{i1j}}{4\sqrt{1+\mathrm{Tr}R}\sin\frac{\varphi}{2}} \\ \frac{(R-R^T)_{ij} e^{i2j}}{4\sqrt{1+\mathrm{Tr}R}\sin\frac{\varphi}{2}} \\ \frac{(R-R^T)_{ij} e^{i3j}}{4\sqrt{1+\mathrm{Tr}R}\sin\frac{\varphi}{2}} \end{pmatrix} \tag{44}$$

Обозначим $(R - R^T) = R_s$, $\frac{1}{4\sqrt{1+\mathrm{Tr}R}} = k_s$ и выпишем в явном виде все три координаты вектора k:

$\boldsymbol{v}_x = k_s(R_{s_{11}}e^{111} + R_{s_{12}}e^{112} + R_{s_{13}}e^{113} + R_{s_{21}}e^{211} + R_{s_{22}}e^{212} + R_{s_{23}}e^{213} + R_{s_{31}}e^{311} + R_{s_{32}}e^{312} + R_{s_{33}}e^{313})$

$\boldsymbol{v}_y = k_s(R_{s_{11}}e^{121} + R_{s_{12}}e^{122} + R_{s_{13}}e^{123} + R_{s_{21}}e^{221} + R_{s_{22}}e^{222} + R_{s_{23}}e^{223} + R_{s_{31}}e^{321} + R_{s_{32}}e^{322} + R_{s_{33}}e^{323})$

$\boldsymbol{v}_z = k_s(R_{s_{11}}e^{131} + R_{s_{12}}e^{132} + R_{s_{13}}e^{133} + R_{s_{21}}e^{231} + R_{s_{22}}e^{232} + R_{s_{23}}e^{233} + R_{s_{31}}e^{331} + R_{s_{32}}e^{332} + R_{s_{33}}e^{333})$

Отбросим все слагаемые с символами Леви-Чивиты с повторяющимися индексами,т.к. согласно правилу 39 они равны нулю:

$$\begin{aligned} \boldsymbol{v}_x &= k_s(R_{s_{23}}e^{213} + R_{s_{32}}e^{312}) \\ \boldsymbol{v}_y &= k_s(R_{s_{13}}e^{123} + R_{s_{31}}e^{321}) \\ \boldsymbol{v}_z &= k_s(R_{s_{12}}e^{132} + R_{s_{21}}e^{231}) \end{aligned} \tag{45}$$

Тогда в явном виде вектор $\boldsymbol{v}$ можно выразить следующим образом:

$$\boldsymbol{v} = \frac{1}{4\sqrt{1+\mathrm{Tr}R}} \begin{pmatrix} R_{s_{23}}e^{213} + R_{s_{32}}e^{312} \\ R_{s_{13}}e^{123} + R_{s_{31}}e^{321} \\ R_{s_{12}}e^{132} + R_{s_{21}}e^{231} \end{pmatrix} \tag{46}$$

$$\boldsymbol{v} = \frac{1}{4\sqrt{1+\mathrm{Tr}R}} \begin{pmatrix} -R_{s_{23}} + R_{s_{32}} \\ R_{s_{13}} - R_{s_{31}} \\ -R_{s_{12}} + R_{s_{21}} \end{pmatrix} \tag{47}$$

Теперь подставим $R_s = R - R^T$:

$$\boldsymbol{v} = \frac{1}{4\sqrt{1+\mathrm{Tr}R}} \begin{pmatrix} -(R-R^T)_{23} + (R-R^T)_{32} \\ (R-R^T)_{13} - (R-R^T)_{31} \\ -(R-R^T)_{12} + (R-R^T)_{21} \end{pmatrix} \tag{48}$$

Т.к. $R_{ij} \equiv R^T_{ji}$, то мы можем раскрыть скобки и выразить вектор $\boldsymbol{v}$ через элементы матрицы поворота:

$$\boldsymbol{v} = \frac{1}{4\sqrt{1+\mathrm{Tr}R}} \begin{pmatrix} -(R_{23} - R_{32}) + (R_{32} - R_{23}) \\ (R_{13} - R_{31}) - (R_{31} - R_{13}) \\ -(R_{12} - R_{21}) + (R_{21} - R_{12}) \end{pmatrix}$$

$$\boldsymbol{v} = \frac{1}{4\sqrt{1+\mathrm{Tr}R}} \begin{pmatrix} -2R_{23} + 2R_{32} \\ 2R_{13} - 2R_{31} \\ -2R_{12} + 2R_{21} \end{pmatrix}$$

Тогда векторный инвариант тензора поворота, представленного в матричном виде будет равен:

$$\boldsymbol{v} = \frac{1}{2\sqrt{1+R_{11}+R_{22}+R_{33}}} \begin{pmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{pmatrix} \tag{49}$$

Найдём длину вектора $|v| = \sin\frac{\varphi}{2}$:

$$\sin\frac{\varphi}{2} = \sqrt{\frac{(R_{32}-R_{23})^2 + (R_{13}-R_{31})^2 + (R_{21}-R_{12})^2}{4(1+R_{11}+R_{22}+R_{33})}} \tag{50}$$

Тогда вектор поворота можно получить из векторного инварианта, нормализовав его:

$$k = \frac{v}{|v|}$$

$$k = \frac{1}{(R_{32}-R_{23})^2 + (R_{13}-R_{31})^2 + (R_{21}-R_{12})^2} \begin{pmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{pmatrix} \tag{51}$$

Т.к. $\sin\frac{\varphi}{2}$ биективно отображает область определения $[0;2\pi]$ на множество значений на интервале $[1;-1]$, то угол φ можно однозначно определить на интервале $[0;2\pi]$ следующим образом:

$$\varphi = 2\arcsin|v|$$

$$\varphi = 2\arcsin\sqrt{\frac{(R_{32}-R_{23})^2+(R_{13}-R_{31})^2+(R_{21}-R_{12})^2}{4(1+R_{11}+R_{22}+R_{33})}} \tag{52}$$

Также нужно найти $\cos\frac{\varphi}{2}$:

$$\mathrm{Tr}R = 1+2\cos\varphi$$

$$\cos\varphi = \frac{1}{2}(\mathrm{Tr}R-1)$$

Т.к. $\cos^2\frac{\varphi}{2} = \frac{1}{2}(1+\cos\varphi)$, то:

$$\cos\frac{\varphi}{2} = \frac{1}{2}\sqrt{1+\mathrm{Tr}R} \tag{53}$$

Тогда кватернион поворота $q = a+bi+cj+dk$, соответствующий матрице поворота R можно записать в следующей форме:

$$\begin{cases} a = \frac{1}{2}\sqrt{1+\mathrm{Tr}R} \\ b = \frac{R_{32}-R_{23}}{4a} \\ c = \frac{R_{13}-R_{31}}{4a} \\ d = \frac{R_{21}-R_{12}}{4a} \end{cases} \tag{54}$$

Аналогичный результат получен в статье [14].

4.2 Проверка

Для проверки найденной связи были сгенерированы 20 случайных кватернионов $\{q_i\}$ методом, описанным в статье [15], после чего при помощи формулы 30 было получено 20 матриц поворота, после чего при помощи формулы 54 были получены новые кватернионы $\{^nq_i\}$.

На рисунке 16 кружками отображены исходные кватернионы, крестиками - полученные из матрицы преобразования

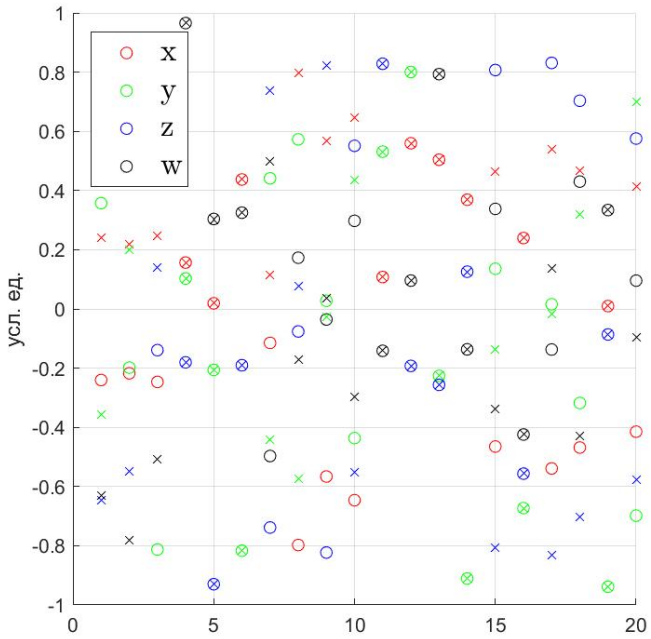

Рисунок 16 – Проверка формулы получения кватерниона из матрицы поворота

Не все кватернионы совпадают, это связано с тем, что один и тот же поворот можно описать двумя кватернионами (a, b, c, d) и $(-a, -b, -c, -d)$, поэтому новые кватернионы ${}^{n}q_i$ необходимо сравнить поэлементно по знаку a и, если он отличается инвертировать новый кватернион. Результаты с инвертированной частью кватернионов представлены на рисунке 17

Из рисунка 17 видно, что алгоритм работает правильно. В данном разделе установлена связь между обобщёнными координатами робота и положением и ориентацией рабочего инструмента:

$$P = \begin{pmatrix} x \\ y \\ z \\ w_r \\ x_r \\ y_r \\ z_r \end{pmatrix} = F_{fk}(T(V)),$$

где x, y, z - координаты рабочего инструмента в пространстве, w_r

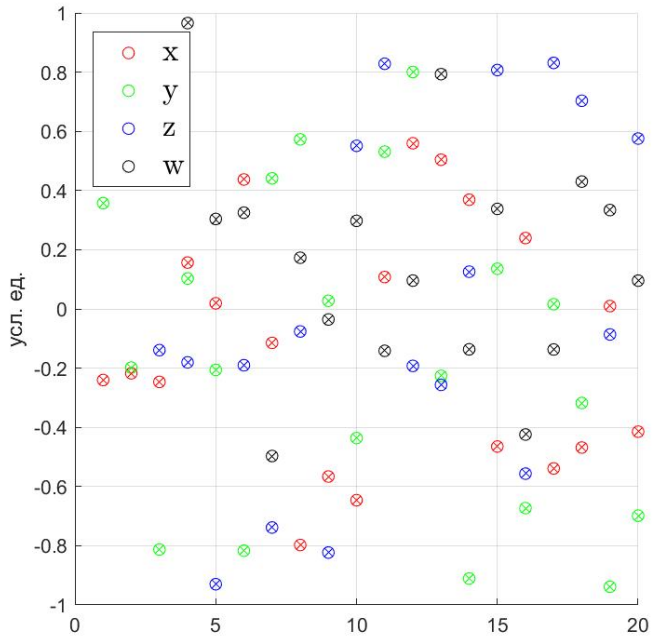

Рисунок 17 – Корректная проверка формулы получения кватерниона из матрицы поворота

- скалярная компонента кватерниона вращения, задающего ориентацию рабочего инструмента в пространстве, r_x, r_y, r_z - векторная компонента, $V = \begin{pmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & R_x & R_y & a \end{pmatrix}^T$ - вектор обобщённых координат, T - матрица перехода из системы координат рабочего инструмента в мировую систему координат.

5 Обратная задача кинематики

Теперь необходимо установить обратную зависимость:

$$V = F_{ik}(T(P)), \tag{55}$$

Т.е. необходимо найти связь между положением и ориентацией рабочего инструмента(вектор P) и обобщёнными координатами робота.

Установим явную зависимость матрицы перехода от вектора P, воспользовавшись формулой 33:

$$R = \begin{pmatrix} w_r^2 + x_r^2 - y_r^2 - z_r^2 & 2(x_r y_r - w_r z_r) & 2(x_r z_r + w_r y_r) & x \\ 2(x_r y_r + w_r z_r) & w_r^2 + y_r^2 - x_r^2 - z_r^2 & 2(y_r z_r - w_r x_r) & y \\ 2(x_r z_r - w_r y_r) & 2(y_r z_r + w_r x_r) & w_r^2 + z_r^2 - x_r^2 - y_r^2 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{56}$$

Для углов Эйлера θ, ψ,ϕ матрица перехода определяется следующим образом:

$$R_{xyz} = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & r_x \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & r_y \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta & r_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{57}$$

Т.е. задача сводится к нахождению углов поворота сочленений по матрице перехода:

$$V = F_{ik}(T) \tag{58}$$

Подход, предложенный в [4] подразумевает геометрический подход в решении обратной задачи кинематики и работают только для артикулированных роботов. Т.е. первые три звена робота переводят рабочий инструмент в заданное положение, а вторые три - в заданную ориентацию. В рамках данной магистерской диссертации рассматривается пятизвенный неартикулированный робот, причём часть обобщённых координат соответствуют движению тележки, а не роботической руки.

Поэтому необходимо решить задачу аналитически.

5.1 Аналитическое решение обратной задачи кинематики

Аналитические способы решения обратной задачи кинематики основываются на итерационных алгоритмах. Решения, основанные на ней-

ронных сетях в рамках данной диссертации не будут рассмотрены.

Все они основаны на минимизации функции ошибки. Ошибка в данном случае означает расстояние между положением, соответствующим текущей конфигурации и целевым положением, конфигурация изменяется так, чтобы расстояние уменьшалось. Такие подходы ещё называют эвристическими.

Рассмотрим несколько алгоритмов решения обратной задачи кинематики:

5.1.1 CCD

CCD - это итеративный численный алгоритм. Поскольку CCD является итеративным, поэтому при его использовании нельзя применять ограничения на каждом шаге, просто беря результирующую ориентацию для любого соединения и заставляя его оставаться в допустимом диапазоне. Однако это влияет на способность CCD сходиться. CCD - это слепой алгоритм горизонтального скручивания; он перемещается по местности, определенной функцией ошибки, пытаясь найти самую высокую точку. Подробное описание алгоритма можно найти в статье [16].

5.1.2 Алгоритм светлячка

Алгоритм Firefly(FA) или алгоритм светлячка, предложенный доктором Синь-Янь, является метаэвристическим алгоритмом оптимизации, основанный на социальном поведении светлячков в тропических климатов. Алгоритм светлячка легко реализуется и значительно превосходит другие алгоритмы с точки зрения точности и эффективности. FFA основано на поведении ошибок светлячков, включая световое излучение, поглощение света и взаимное притяжение, которое было разработано для решения задач непрерывной оптимизации Поскольку алгоритм Firefly является новым алгоритмом, он редко используется в обратной задаче кинематики. В существующих исследованиях, определяющих критерий оптимальности, основанный на частоте ошибок, представляющих собой расстояния между целевым и фактическим положением. Подробное описание алгоритма можно найти в статье [17].

5.1.3 Матрица Якоби

Согласно теории [4], матрица Якоби связывает скорости рабочего инструмента в декартовом пространстве со скоростями вращения сочленений робота:

$$J_f(a) := \left(\frac{\partial f_i}{\partial x_j}(a)\right)_{i=1,\ldots,m;\ j=1,\ldots,n} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(a) & \frac{\partial f_1}{\partial x_2}(a) & \ldots & \frac{\partial f_1}{\partial x_n}(a) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(a) & \frac{\partial f_m}{\partial x_2}(a) & \ldots & \frac{\partial f_m}{\partial x_n}(a) \end{pmatrix} \quad (59)$$

В случае мобильных роботов x_j - это углы поворота звеньев, n - количество звеньев, f_i - это координаты положения рабочего инструмента, которое является функцией от углов поворотов звеньев.

Тогда скорость энд-эффектора v связана со скоростями вращения ω:

$$v = J\omega \quad (60)$$

В случае, если матрица якоби является квадратной, то за исключением сингулярных точек(конфигураций, при которых определитель матрицы Якоби равен 0), задачу можно решишь найдя обратную матрицу Якоби:

$$\omega = J^{-1}v \quad (61)$$

Для случаев сингулярных точек применяется регуляризация [20]:

Если матрица Якоби является неквадратной, то необходимо использовать псевдообратную матрицу Якоби:

$$J^{+} = (J^T J)^{-1} J^T \quad (62)$$

$$v = J^{+}\omega \quad (63)$$

После нахождения связи между скоростью рабочего инструмента, можно построить регулятор, "поворачивающий"звенья так, чтобы линейная скорость рабочего инструмента была направлена в сторону целевого положения. Подробное описание алгоритма можно найти в статье [18].

5.1.4 Колония муравьёв

Метод колонии муравьёв(Ant Colony Optimisation) - относительно новый подход к решению проблем, который создан на основе наблюдений за социальным поведением некоторых видов муравьёв. Эти муравьи помещают феромон на землю, чтобы отметить какой-то благоприятный путь, за которым должны следовать другие члены колонии. Оптимизация методом колонии муравьев использует аналогичный механизм для решения задач оптимизации. С начала девяностых годов, когда был предложен первый алгоритм оптимизации муравьиной колонии. Подробное описание алгоритма можно найти в статье [19]

5.2 Оптимизационный подход к решению обратной задачи кинематики

В рамках данной диссертации будет рассмотрен классический подход к поиску минимума функции.

Для начала необходимо ввести функцию ошибки. Если рассматривать только положение рабочего инструмента, то функция ошибки является тривиальной и равна длине вектора, представляющего разность текущего положения и целевого.

Если же необходимо учитывать ещё и ориентацию, то необходимо определить расстояние между двумя единичными кватернионами, описывающими текущее положение и заданное.

Т.к. кватернион поворота - единичный вектор и один и тот же поворот может описывать как кватернион q, так и кватернион $-q$, то подход, использованный при нахождении расстояний в пространстве не подходит.

Расстояние между кватернионами можно определить следующим образом:

$$d(q_1, q_2) = 1 - (q1, q2)^2, \tag{64}$$

где $(q1, q2)$ - скалярное произведение кватернионов.

$$(a_1 + b_1 i + c_1 j + d_1 k, a_2 + b_2 i + c_2 j + d_2 k) = a_1 a_2 + b_1 b_2 + c_1 c_2 + d_1 d_2 \tag{65}$$

По сути, расстояние между кватернионами определяется через угол между ними в четырёхмерном пространстве

Тогда функция ошибки имеет вид:

$$E(V, r^*, q^*) = k_r(r(V) - r^*)^2 + k_q(1 - (q(V), q^*)), \quad (66)$$

Где V - обобщённые координаты робота, r^* - целевое положение рабочего инструмента, q^* - кватернион целевой ориентации робота, $r(V)$ - положение рабочего инструмента, соответствующее конфигурации V, $q(V)$- кватернион поворота.

В итоге задача сводится к поимку минимума функции ошибки.

5.2.1 Оптимизация

Для проверки обратной задачи кинематики при помощи решённой прямой задачи был создан тестовый набор данных, каждая запись которого представляет из себя конфигурацию робота, положение его рабочего инструмента и ориентацию, заданную кватернионом.

Для решения задачи оптимизации было использовано программное обеспечение Matlab. Функция ошибки была выведена при помощи аппарата символьных выражений, после чего упрощена функцией simlify.

После чего при помощи функции fmincon был проведён ряд экспериментов для установления наилучшего алгоритма поиска минимума. На вход функции подаётся выражение, которое необходимо минимизировать, стартовое значение параметров(в нашем случае, обобщённых координат) и границы значений для каждого из параметров.

Matlab поддерживает несколько методов оптимизации:

а «interior-point» - крупномасштабный алгоритм, работающий для широкого круга задач

б «sqp» - Алгоритм может восстанавливаться из результатов NaN или Inf. Это не крупномасштабный алгоритм;

в «sqp-legacy» - тоже самое, что «sqp», но работает дольше и может занять больше памяти

г «active-set» - может делать большие шаги, что увеличивает скорость работы. Алгоритм эффективен для некоторых задач с негладкими ограничениями. Это не крупномасштабный алгоритм

д «trust-region-reflective» - требует градиент и допускали только ограничения или ограничения линейного равенства, но не оба.

В рамках этих ограничений алгоритм эффективно обрабатывает как большие разреженные задачи, так и малые плотные. Это крупномасштабный алгоритм. Алгоритм может использовать специальные методы для экономии использования памяти, такие как функция умножения Гессиана.

Алгоритм оптимизации имеет большой масштаб, когда он использует линейную алгебру, которой не нужно хранить и не работать с полными матрицами. Это можно сделать внутренне, сохраняя разреженные матрицы и используя, по возможности, разреженную линейную алгебру для вычислений. Кроме того, внутренние алгоритмы либо сохраняют разреженность, например, разреженную декомпозицию Холецкого.

Напротив, средние методы внутренне создают полные матрицы и используют плотную линейную алгебру. Если проблема достаточно велика, полные матрицы занимают значительную часть памяти, а для плотной линейной алгебры может потребоваться долгое время.

5.2.2 Моделирование

Моделирование выполнялось следующим образом: для $k_r = 1$, $k_q = 1$, $k_r = 1$, $k_q = 100$, $k_r = 1$, $k_q = 200$ 50 раз брались две случайные точки из подготовленных данных. Одна из точек полагалась как текущая, вторая, как целевая. Далее при помощи различных алгоритмов поиска оптимального решения по заданным положению и ориентации целевой точки находились конфигурации, соответствующие данной точке. Поиск минимума останавливался на значении критерия, равном 1.

Результаты моделирования представлены на рисунках 18,19, 20, 21,22, 23, 24,25, 26, 27,28, 29.

Сборная информация о качестве алгоритмов представлена в таблице 3.

а k_q - вес расстояния между кватернионами

б mean(e) - среднее значение ошибки

в std(e) - стандартное отклонение

г mean(it) - среднее число итераций

д overlays - число точек, определённых с точностью, меньше требуемой.

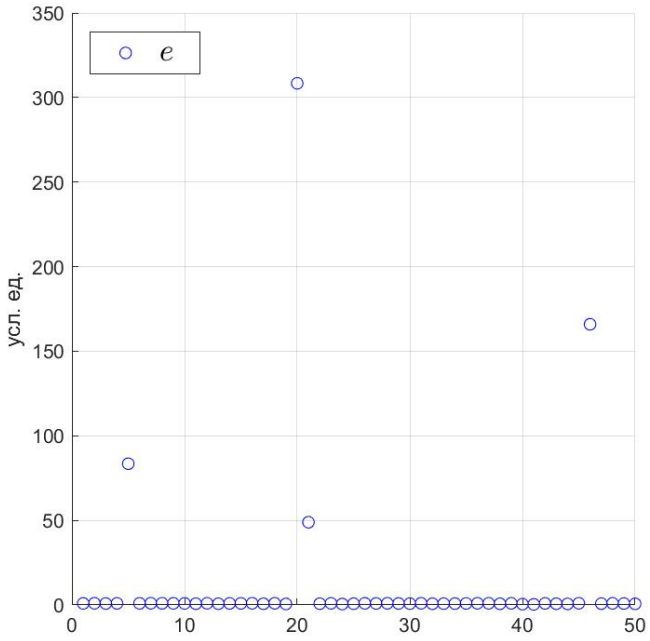

Рисунок 18 – Ошибка решения обратной задачи кинематики алгоритм: интериор-поинт, $k_r = 1,\ k_q = 1$

5.3 Выводы

В ходе моделирования было установлено, что с увеличением веса расстояния кватернионов, увеличивается надёжность решения обратной задачи кинематики при помощи алгоритмов «interior-point» и «active-set», алгоритмы «sqp» и «sqp-legacy» наоборот с увеличением веса показали результаты хуже.

Результаты работы «sqp» и «sqp-legacy», действительно показали очень близкие результаты.

Наилучшим алгоритмом для решения обратной задачи кинематики является «interior-point». Из этого можно сделать вывод, что для поиска решения обратной задачи кинематики нужны крупномасштабные алгоритмы.

Иллюстрация работы алгоритма «interior-point» с весом расстояния кватернионов $k_q = 100$ приведена на рисунках 31,30, 32, 33, 34, 35.

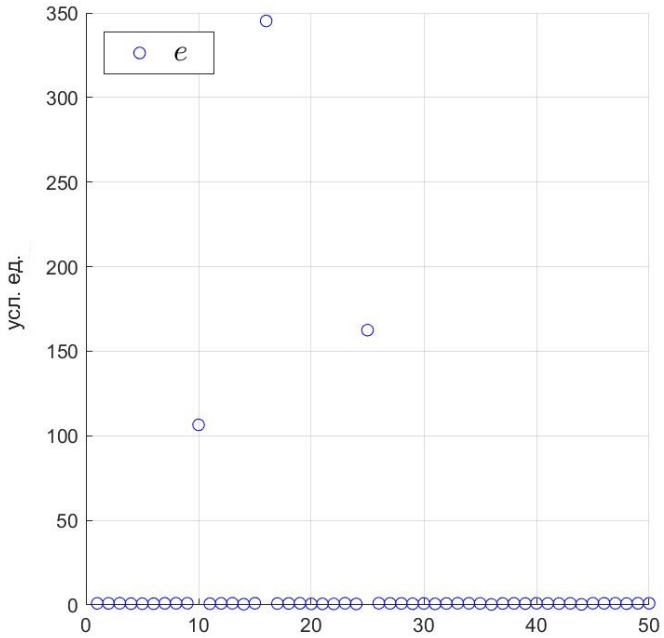

Рисунок 19 – Ошибка решения обратной задачи кинематики алгоритм: интериор-поинт, $k_r = 1,\ k_q = 100$

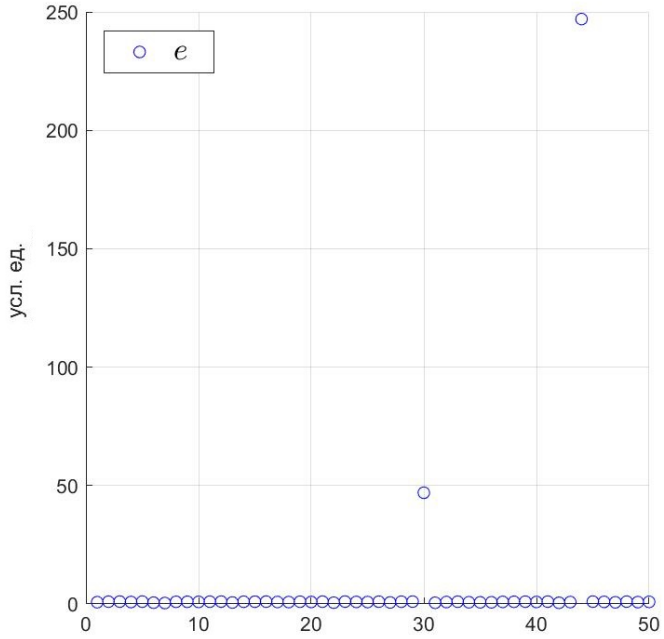

Рисунок 20 – Ошибка решения обратной задачи кинематики алгоритм: интериор-поинт, $k_r = 1,\ k_q = 200$

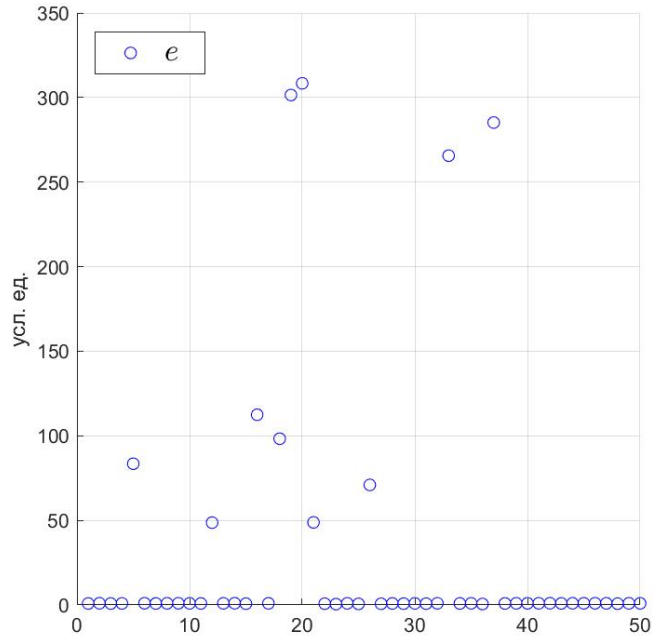

Рисунок 21 – Ошибка решения обратной задачи кинематики алгоритм: скп, $k_r = 1,\ k_q = 1$

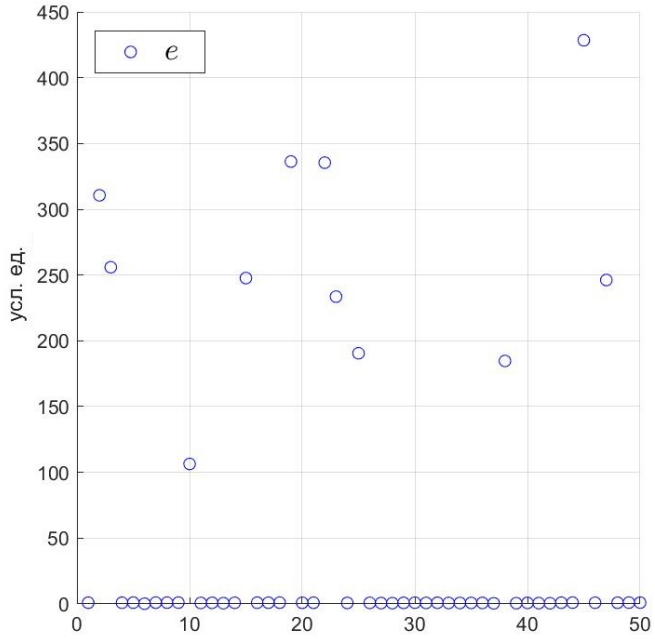

Рисунок 22 – Ошибка решения обратной задачи кинематики алгоритм: скп, $k_r = 1,\ k_q = 100$

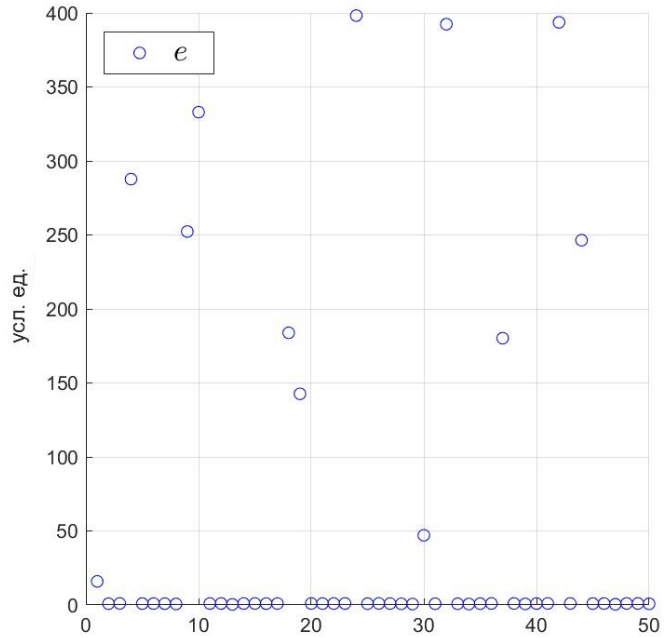

Рисунок 23 – Ошибка решения обратной задачи кинематики алгоритм: скп, $k_r = 1,\ k_q = 200$

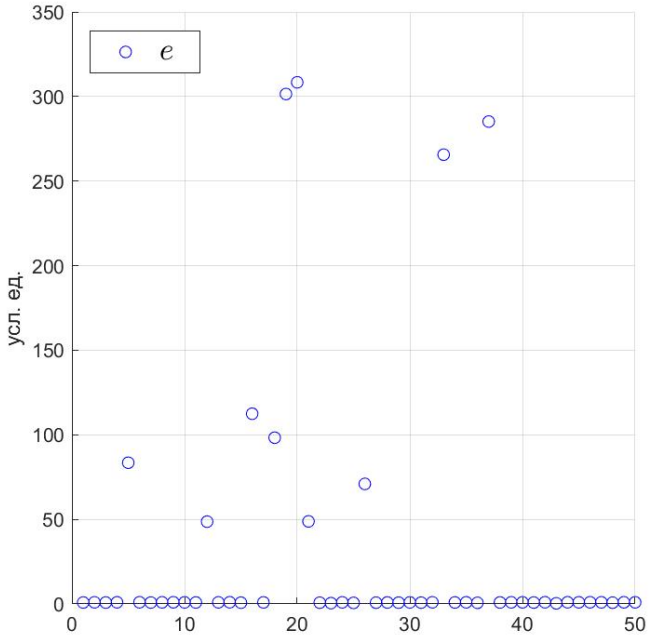

Рисунок 24 – Ошибка решения обратной задачи кинематики алгоритм: скп-легаси, $k_r = 1,\ k_q = 1$

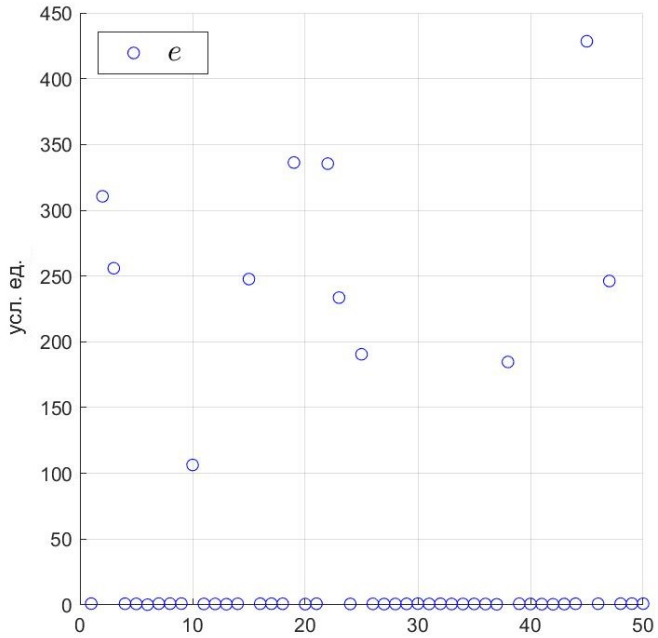

Рисунок 25 – Ошибка решения обратной задачи кинематики алгоритм: скп-легаси, $k_r = 1,\ k_q = 100$

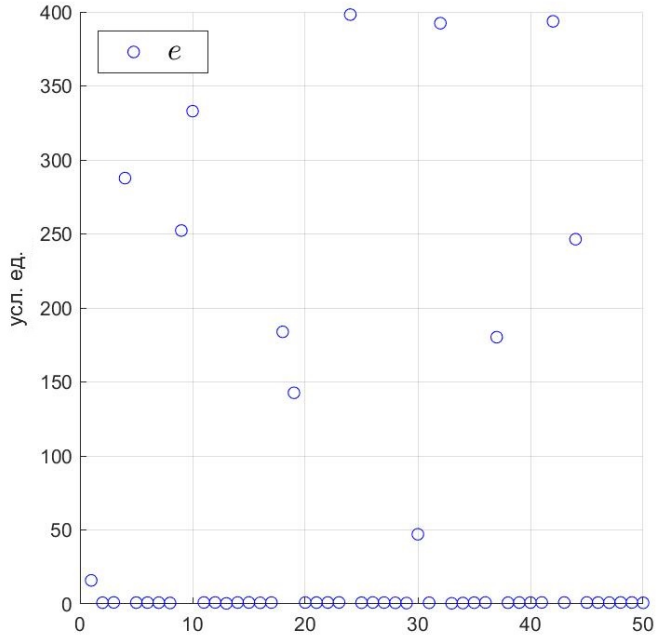

Рисунок 26 – Ошибка решения обратной задачи кинематики алгоритм: скп-легаси, $k_r = 1,\ k_q = 200$

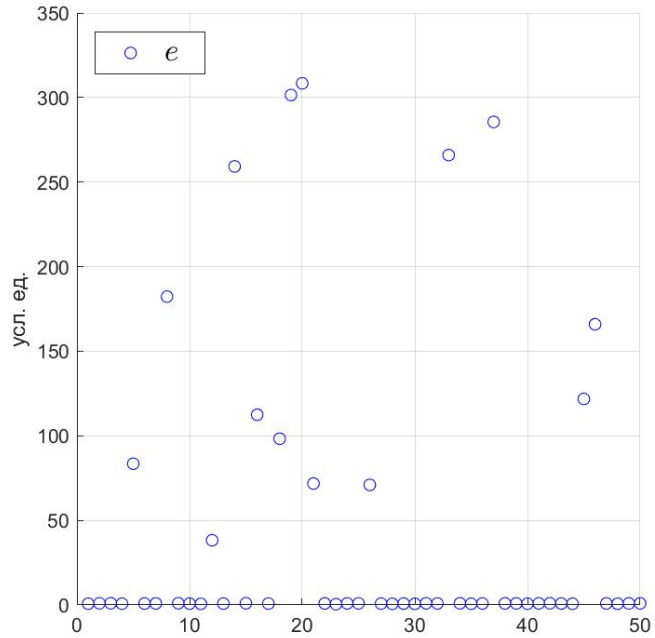

Рисунок 27 – Ошибка решения обратной задачи кинематики алгоритм: актив-сет, $k_r = 1,\ k_q = 1$

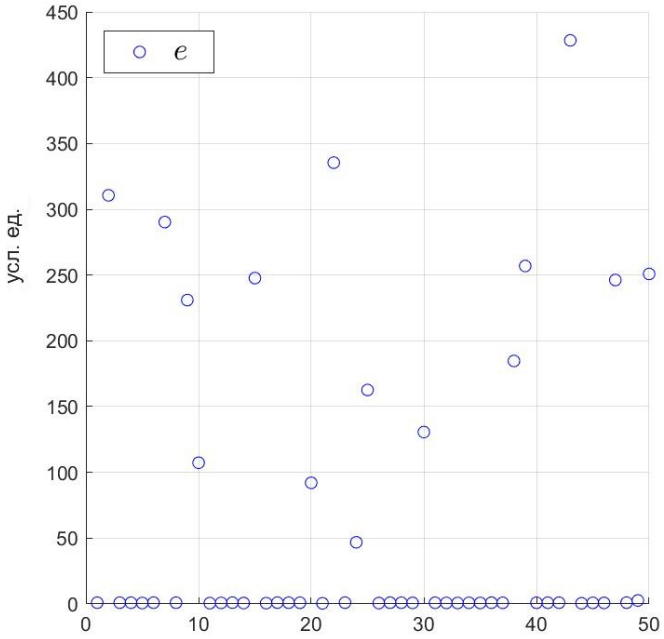

Рисунок 28 – Ошибка решения обратной задачи кинематики алгоритм: актив-сет, $k_r = 1,\ k_q = 100$

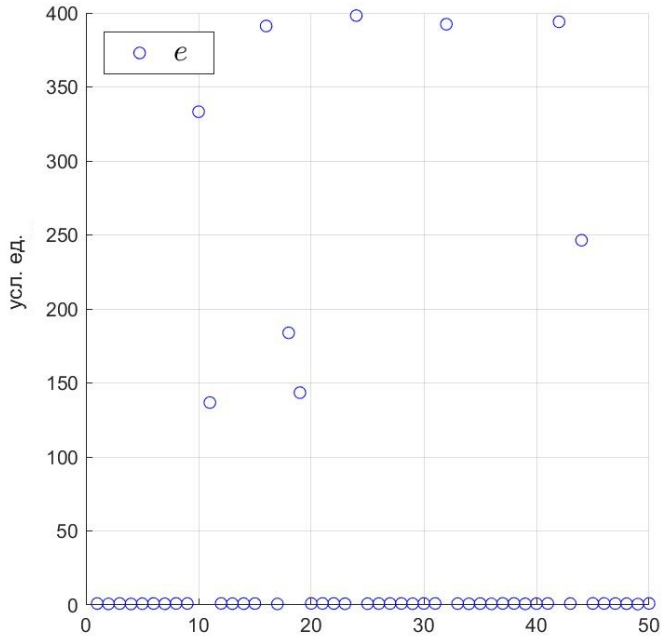

Рисунок 29 – Ошибка решения обратной задачи кинематики алгоритм: актив-сет, $k_r = 1,\ k_q = 200$

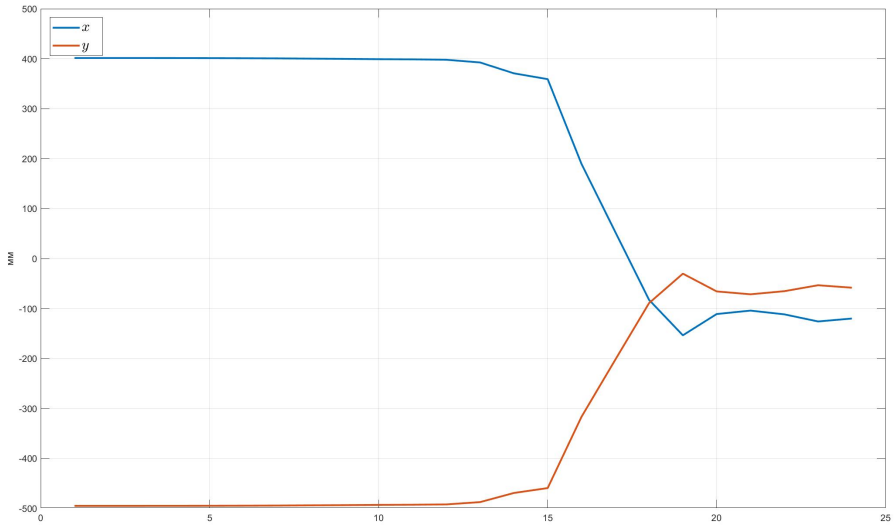

Рисунок 30 – Поиск положения тележки, соответствующего заданному положению рабочего инструмента

Таблица 3 - Результаты сравнения алгоритмов поиска минимума

	k_q	mean(e)	std(e)	mean(it)	overlays
interior-point	1	12.901	50.243	34.700	4
	100	13.072	55.036	31.600	3
	200	6.683	35.286	32.360	2
sqp	1	33.171	81.057	36.960	10
	100	58.155	116.089	34.720	11
	200	58.125	120.007	36.440	12
sqp-legacy	1	33.161	81.061	37.220	10
	100	58.146	116.093	34.460	11
	200	58.129	120.005	36.240	12
active-set	1	47.921	91.156	36.240	15
	100	67.055	115.705	33.240	16
	200	53.119	121.602	32.940	9

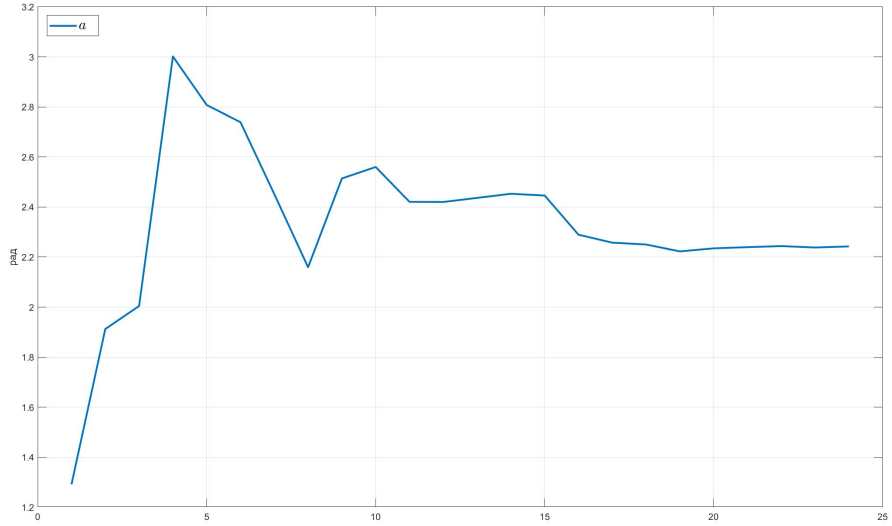

Рисунок 31 – Поиск ориентации тележки, соответствующего заданному положению рабочего инструмента

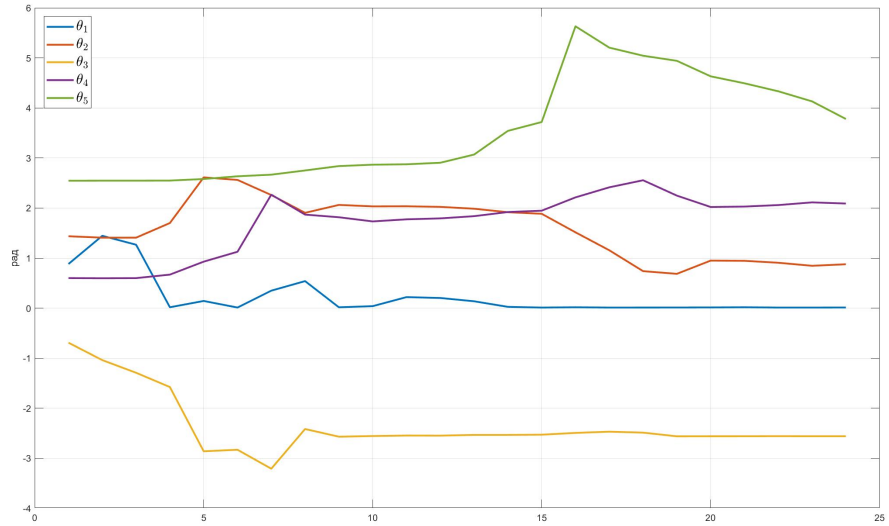

Рисунок 32 – Поиск углов поворота звеньев манипулятора, соответствующих заданному положению рабочего инструмента

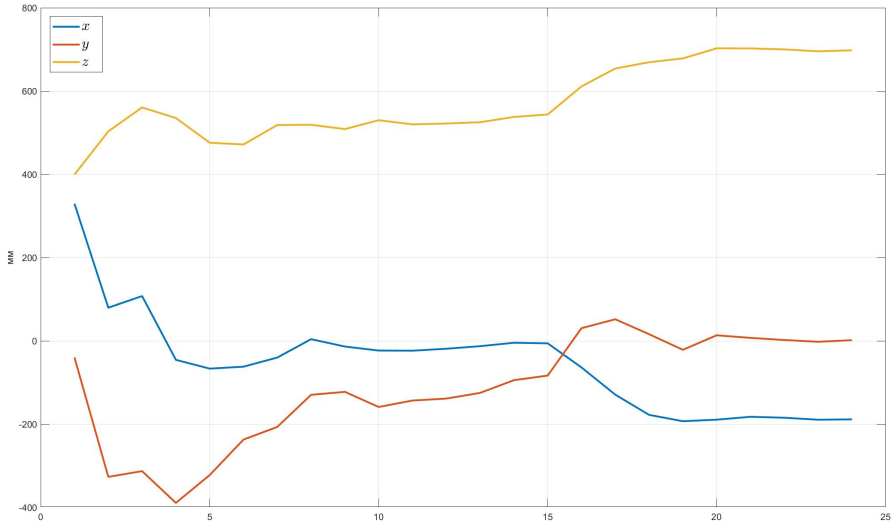

Рисунок 33 – Промежуточные положения рабочего инструмента

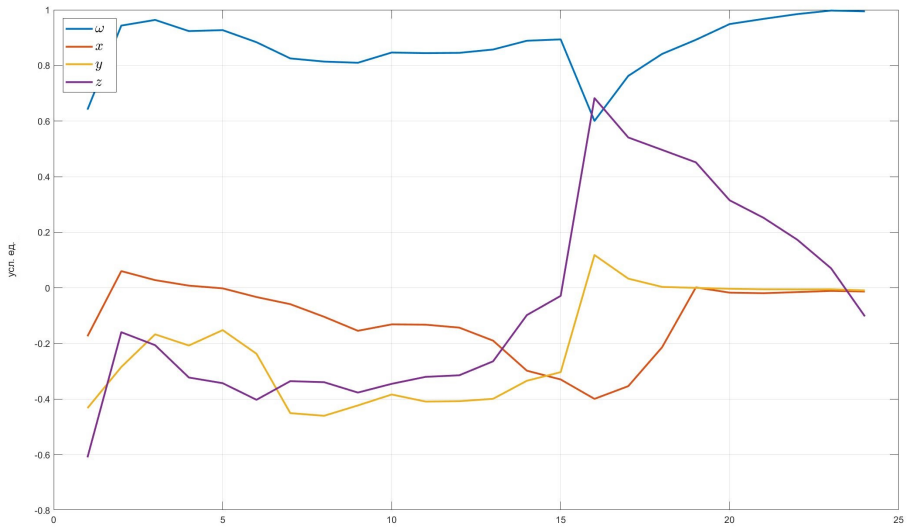

Рисунок 34 – Промежуточные ориентации рабочего инструмента

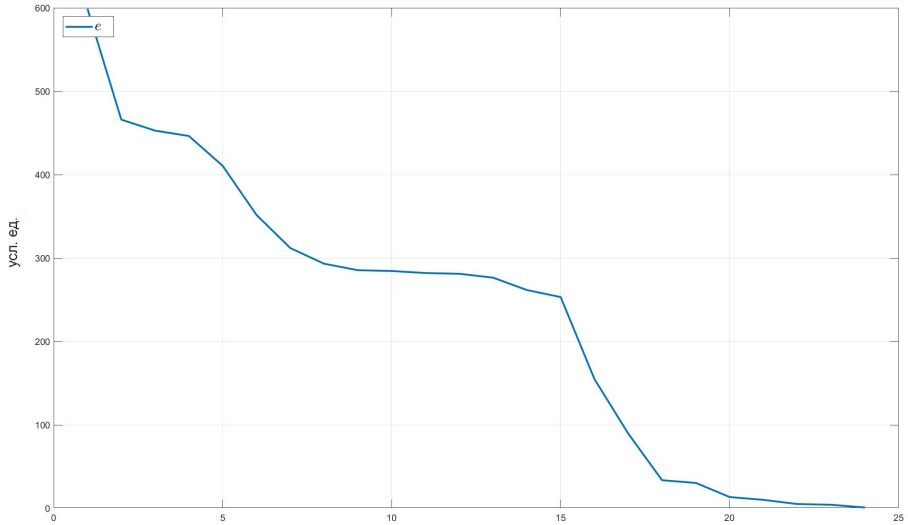

Рисунок 35 – График расстояния между положением, соответствующем промежуточной конфигурации и целевым

6 Управление

Для проверки полученных результатов был спланирован путь в декартовой системе координат. После чего при помощи симулятора Gazebo и алгоритмов, полученных в рамках данной диссертации было реализовано управление, обеспечивающее прохождение инструментом заданных точек(положение и ориентация).

6.1 Построение траектории

первая точка траектории - это начальное положение робота $r = r(V_0),\ q = q(V_0)$, где

$$V_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Остальные точки получены при помощи прямой задачи кинематики от случайной конфигурации робота. Единственное требование к конфигурации заключается в том, чтобы манипулятор не имел самопересечений и не пересекался с тележкой или рабочей поверхностью. Это было проверено эмпирически.

Результаты планирования траекторий представлены на рисунках 36,37,38, 39,40

Как видно из графиков, хотя ориентации и положения находятся верно, но найденные конфигурации и исходные различаются. Это связано с тем, что система имеет 7 степеней свободы, поэтому одному и тому же положению могут соответствовать несколько конфигураций.

6.2 Отработка траектории

Опорные точки траектории в декартовом пространстве представлены в таблице 4.

Опорные точки траектории в конфигурационном пространстве представлены в таблице 5.

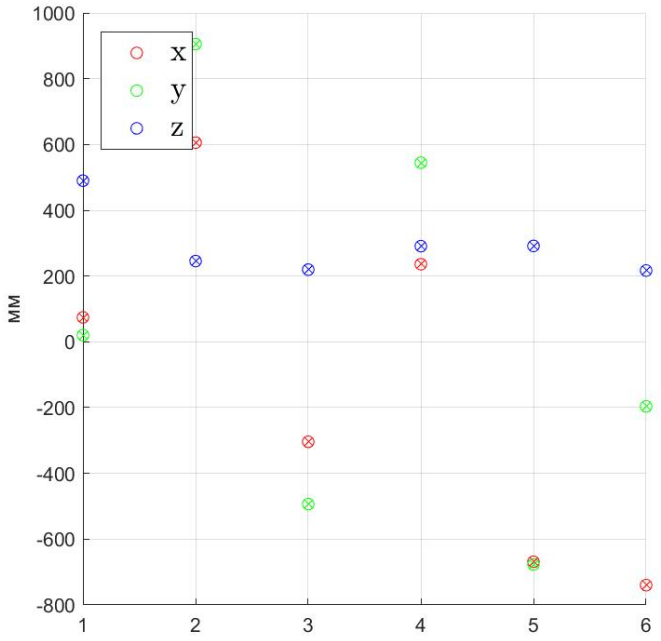

Рисунок 36 – Опорные положения траектории

Результаты отработки траекторий представлены на рисунках 41,42,43,44,45,

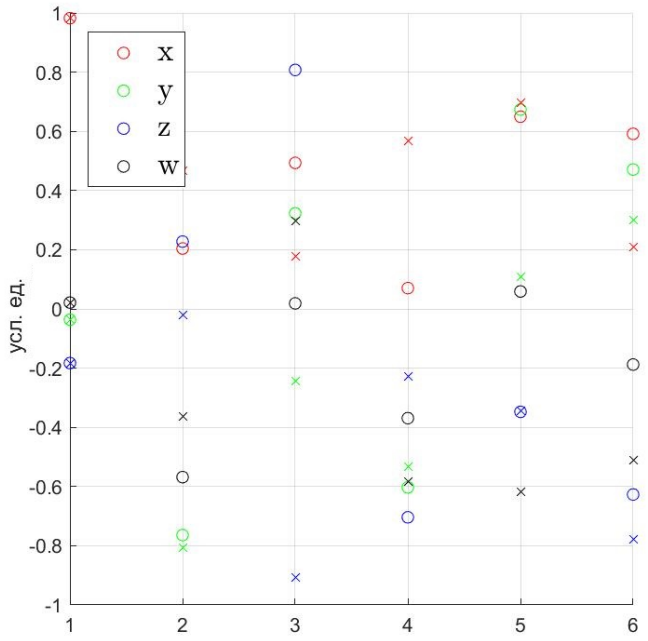

Рисунок 37 – Опорные ориентации траектории

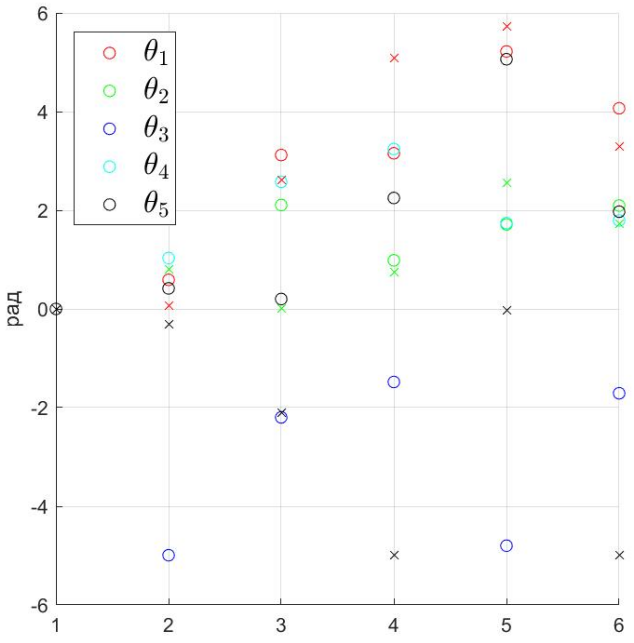

Рисунок 38 – Опорные конфигурации манипулятора траектории

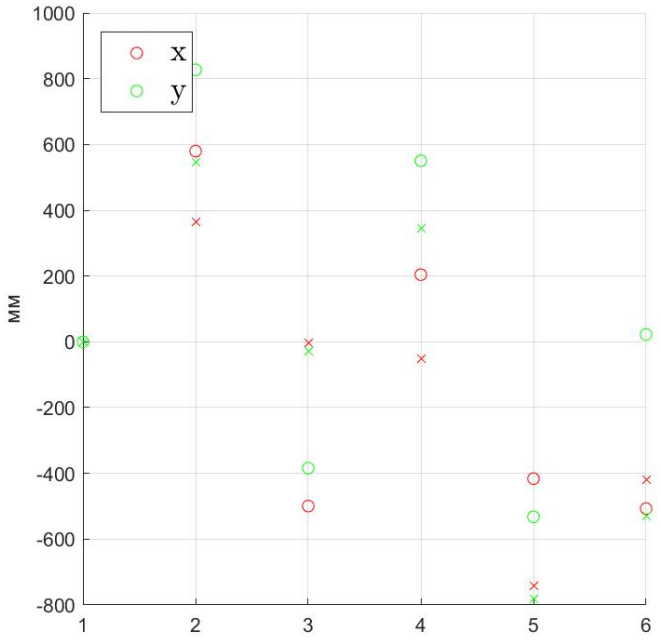

Рисунок 39 – Опорные положения колёсной базы траектории

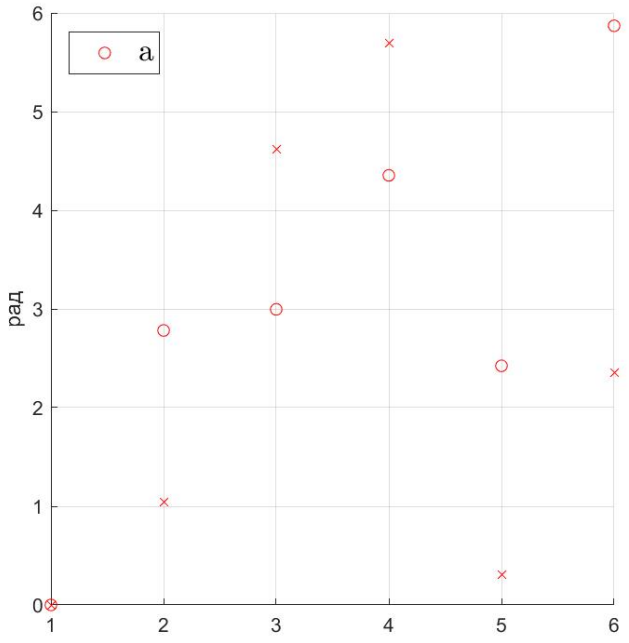

Рисунок 40 – Опорные ориентации колёсной базы траектории

Таблица 4 - Опорные точки траектории

r_x	r_y	r_z	q_ω	q_x	q_y	q_z
74.3131	20.3987	489.9426	0.9822	-0.0364	-0.1829	0.0214
605.7269	905.2726	245.4581	0.2041	-0.7640	0.2274	-0.5682
-303.9059	-493.3208	219.4856	0.4934	0.3232	0.8073	0.0188
235.9207	544.5788	290.8006	0.0703	-0.6029	-0.7038	-0.3691
-668.6119	-676.6945	291.5218	0.6495	0.6736	-0.3478	0.0590
-739.7803	-196.3418	216.7992	0.5915	0.4710	-0.6269	-0.1879

Таблица 5 - Опорные конфигурации траектории

θ_1	θ_2	θ_3	θ_4	θ_5	a	x	y
0	0	0	0	0	0	0	0
0.0689	0.7945	-0.3021	1.7566	3.8080	1.0413	364.5966	544.4541
2.6096	0.0114	-2.1013	0.0338	1.0760	4.6193	-3.2475	-29.4380
5.0818	0.7485	-4.9997	3.3888	0.1506	5.6905	-50.0071	345.0369
5.7339	2.5534	-0.0329	3.3975	5.0326	0.3039	-741.8579	-781.8819
3.2939	1.7188	-4.9821	1.6655	4.5182	2.3523	-419.0646	-528.6642

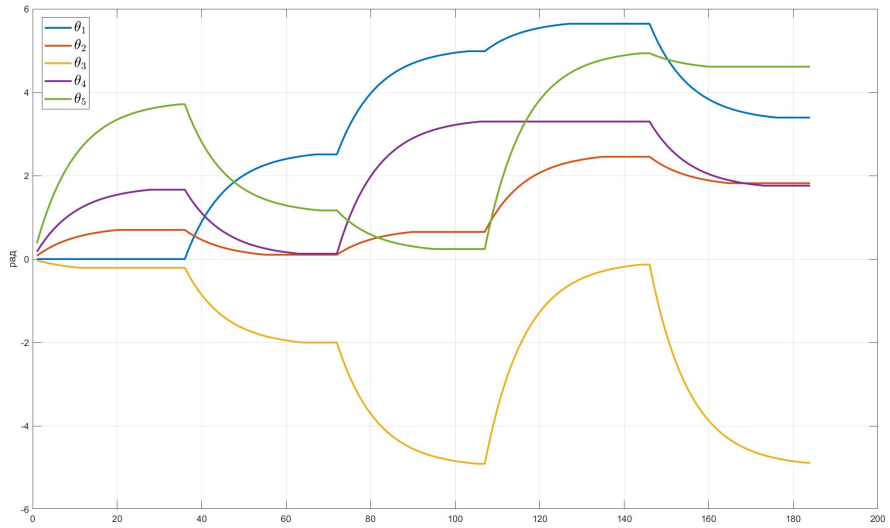

Рисунок 41 – Отработка траектории: конфигурации манипулятора

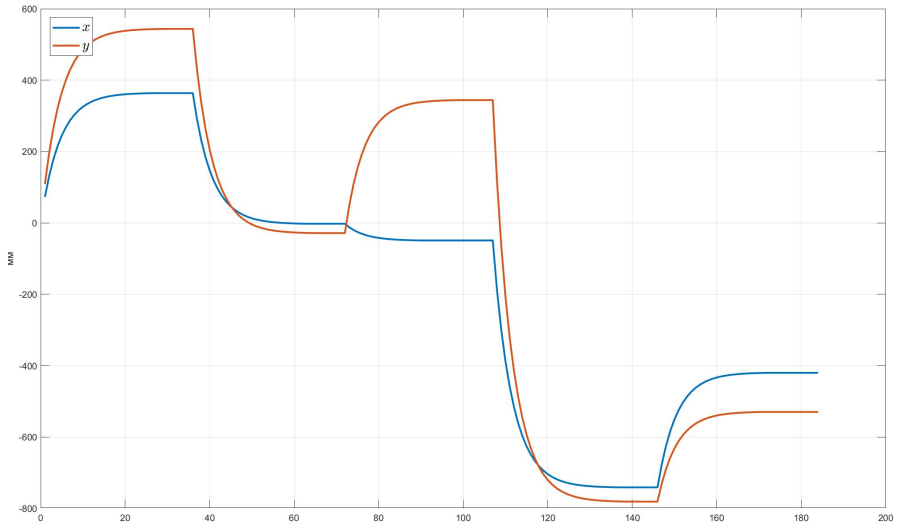

Рисунок 42 – Отработка траектории: положение колёсной базы

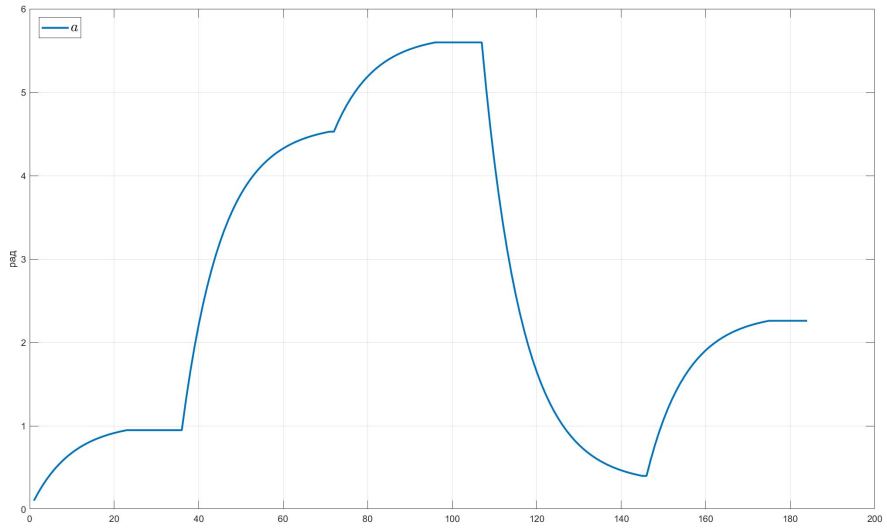

Рисунок 43 – Отработка траектории: ориентация колёсной базы

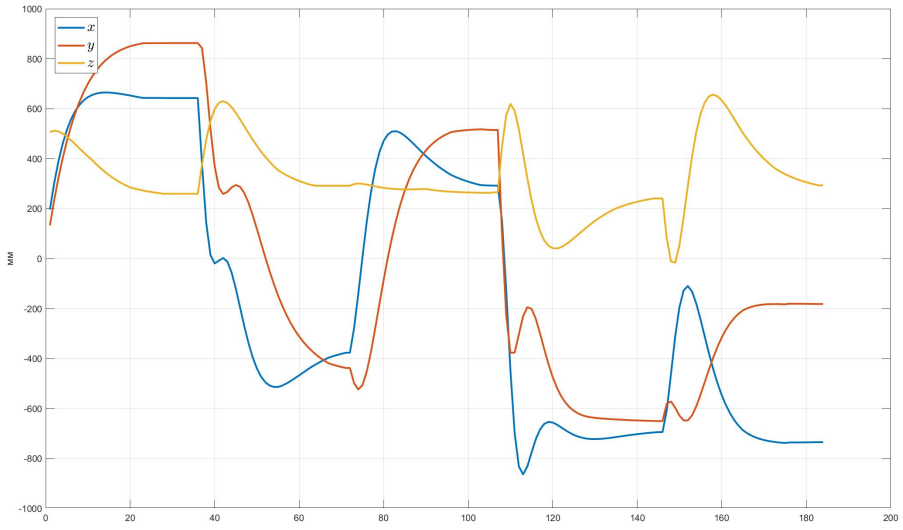

Рисунок 44 – Отработка траектории: положение рабочего инструмента

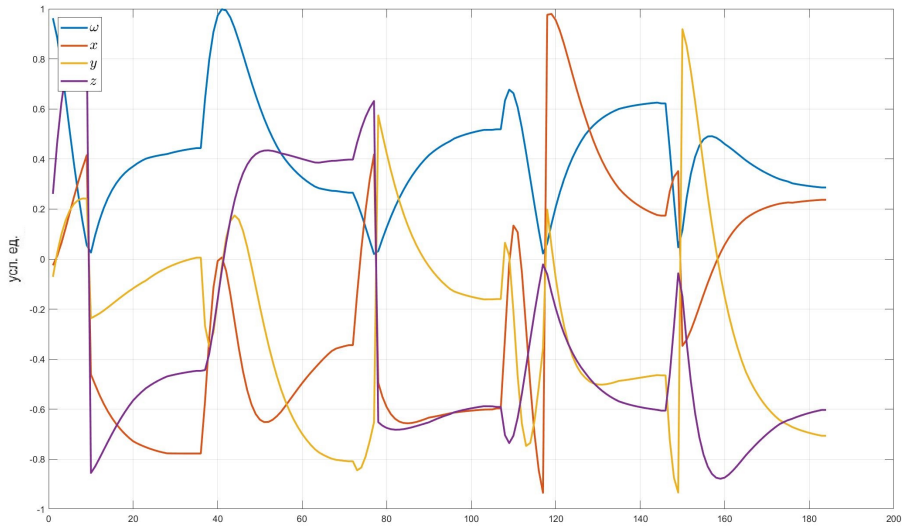

Рисунок 45 – Отработка траектории: ориентация рабочего инструмента

Заключение

В рамках данной магистерской диссертации был предложен новый способ задания ориентации рабочего инструмента робота, были рассмотрены стандартные способы решения обратной задачи кинематики. Были получены аналитические выражения для прямой и обратной задач кинематики мобильного робота, также были рассмотрены различные алгоритмы поиска минимума функции расстояния между двумя положениями робота, также было введено расстояние между ориентациями робота.

Полученные результаты позволяют повысить эффективность позиционирования рабочего инструмента робота, что очень важно в гибком производстве, т.к. целевые конфигурации устанавливаются динамически, а сходимость решения не гарантируется. Представленный подход повышает надёжность системы и позволяет реализовывать более сложное управление. Также представленный подход может быть задействован в алгоритмах «быстрого решения» обратной задачи кинематики.

Помимо представленных результатов, данный подход можно использовать в планировании траекторий, т.к. скорость изменения углов Эйлера связана с тензором угловой скорости нетривиально, скорость изменения кватернионов - довольно изученная тема, что предоставляет фундамент для дальнейших исследований управления и моделирования роботов.

Список используемых источников

1. C. S. Marder-Eppstein, E. Meeussen, W. Pradeep, V. R. Tsouroukdissian, A. Bohren, J. Coleman, D. Magyar, B. Raiola, G. Lüdtke, M. Fernández Perdomo "ros_control: A generic and simple control framework for ROS"The Journal of Open Source Software 2017
2. J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices"Trans ASME J. Appl. Mech. 23: 215-221 1955
3. L.D. Landau and E. M. Lifshitz, Mechanics (3rd ed.) Oxford: Butterworth-Heinemann 1996 - 168 c.,
4. Mark W. Spong, S. Hutchinson and M. Vidyasagar "Robot Modeling and Control"2005 - 496
5. James E. Gentle, Matrix Algebra: Theory, Computations, and Applications in Statistics. Springer, 2007 - 529 c.
6. Gregory G Slabaugh "Computing Euler angles from a rotation matrix"Retrieved on August 1999 cc/ 39-63
7. Tildon H. Glisson, Introduction to Circuit Analysis and Design,Springer Science & Business Media 2011
8. Eric M. Jones "Gimbal Angles, Gimbal Lock, and a Fourth Gimbal for Christmas"Paul Fjeld 2006
9. S. I. Kruglova and V. Barzdab,Modified Gibbs?s representation of rotation matrix Annales de la Fondation Louis de Broglie, Vol. 42, No 2 , 2017, c. 1-15
10. Simon Altmann,Rotations, Quaternions and Double Groups,Clarendon Press, Oxford 1986
11. J. B. Kuipers,Quaternions and Rotation Sequences. Princeton University Press 1999
12. D. E. Neuenschwander,Tensor Calculus for Physics Johns Hopkins University Press 2015
13. L. P. Kuptsov,Hazewinkel, Michiel, Encyclopedia of Mathematics, Springer Science+Business Media B.V. / Kluwer Academic Publishers 2001
14. Mebius, Johan, Derivation of the Euler?Rodrigues formula for three-dimensional rotations from the general formula for four-dimensional rotations 2007

15. Steven M. LaValle, Generating a random element of SO(3) Cambridge Univercity Press 2006
16. Li-Chun Tommy Wang, Chih Cheng Chen "A Combined Optimization Method for Solving the Inverse Kinematics Problem of Mechanical Manipulators, IEEE Transactions on Robotics and Automation, Vol 7 No 4, August, 1991
17. X.S. Yang,Firefly algorithms for multimodal optimization. In Stochastic algorithms: foundations and applications.Springer Heidelberg 2009 - c.169-178
18. K. Tchon and J. Jakubiak, JACOBIAN INVERSE KINEMATICS. Advances in Robot Kinematics: Mechanisms and Motion 2006 465 c.
19. M. Dorigo and M. Birattari and T. Stutzle,Ant colony optimization.Computational Intelligence Magazine, IEEE, 1(4), 2006, cc. 28-39
20. Daniel Andras Drexler, Istvan Harmati, "Regularized Jacobian for the differential inverse positioning problem of serial revolute joint manipulators Intelligent Systems and Informatics (SISY), IEEE 11th International Symposium on 2013