Руководство пользователя (документация) по использованию программного инструмента «Моделирование роботов»

Приложение было разработано Ротахиной А.А. в рамках технического проекта для выпускной квалификационной работы по теме «Программные средства для моделирования исполнительных подсистем манипуляционных роботов» на кафедре проблем управления РТУ МИРЭА в 2022–2023. За основу был взят и технически усовершенствован аналогичный программный комплекс, разработанный в 1993 году Тягуновым О.А. и Козловым Д.В.

В качестве среды разработки было использовано бесплатное кроссплатформенное программное обеспечение PyCharm. Приложение было реализовано на языке программирования Python при использовании доступных библиотек и модулей для конструирования интерфейса, моделирования уравнений кинематики и динамики роботов и визуализации результатов моделирования в зависимости от задаваемых пользователем параметров.

Содержание

Словарь терминов

Робот-Декарт – двухзвенный робот, имеющий степени подвижности поступательного типа, одно из его звеньев заканчивается схватом, в котором может располагаться груз. Звенья перемещаются по координатной плоскости, вдоль осей x и y.

Робот-Цилиндр – двухзвенный робот, первая степень подвижности которого вращательного типа, а вторая – поступательного. Одно звено вращается вокруг шарнира в горизонтальной плоскости, второе – совершает поступательное перемещение.

Робот-Скара – двухзвенный робот, имеющий степени подвижности вращательного типа. Оба звена робота соединены с шарниром и вращаются в горизонтальной плоскости.

Робот-Колер – двухзвенный робот, первая степень подвижности которого поступательного типа, а вторая – вращательного. Одно звено совершает поступательное перемещение, второе – вращается вокруг шарнира в горизонтальной плоскости.

Рабочая область – геометрическое пространство, где может располагаться рабочий орган манипулятора при всех возможных значениях обобщённых координат в степенях подвижности манипулятора.

Рабочий орган (схват) – устройство для захватывания и удержания объектов манипулирования, служащее для взаимодействия с объектами внешней среды и перемещающееся в пределах рабочей области.

Контур – моделируемое перемещение рабочего органа в пределах рабочей области, как правило, имеет вид замкнутой фигуры.

Позиционное движение (управление) – управление роботом, реализуемое за счёт перемещения рабочего органа робота от одной позиции/точки к другой.

Контурное движение (управление) – управление роботом, при котором задаётся вся траектория движения и скорости перемещения по осям, траектория может иметь вид прямой или окружности.

Json – текстовый формат обмена и передачи данных, в случае работы приложения представляет собой сохраняемый и загружаемый файл, в котором содержатся все настройки программного инструмента от выбранного типа робота или управления до значений циклограммы.

Циклограмма – прямоугольная матрица, в которой заданы последовательность моментов времени и соответствующие положения

обобщённых координат, моменты времени задаются в возрастающем порядке.

Обобщённые координаты – преобразованные декартовые координаты или параметры, необходимые для определения положения механической системы робота в пространстве, соответствуют числу степеней свободы манипулятора, для поступательных звеньев характеризуют перемещение в метрах, для вращательных – угол поворота в радианах.

Конструктивные параметры – физические характеристики манипулятора, определяющие массу, моменты инерции и длины его звеньев, необходимы для реализации уравнений динамики роботов.

Ограничения по координатам – минимальные и максимальные значения поступательных и вращательных звеньев, определяющие размерность рабочей области.

Параметры двигателей – набор физических характеристик приводов манипулятора, которые осуществляют управление поворотами соответствующих звеньев робота в горизонтальной плоскости, необходимы для реализации уравнений динамики роботов.

J – момент инерции ротора.

n – передаточный коэффициент редуктора.

Umax – максимальное напряжение двигателя.

Ku – коэффициент управления по напряжению.

Kq – коэффициент противо-ЭДС.

Параметры регуляторов – набор коэффициентов, состоящий из пропорционального, интегрального и дифференцирующего коэффициентов ПИД-регулятора, которые отвечают за отладку процесса регулирования позиции, скорости и ускорения перемещения рабочего органа и используются для обеспечения стабильного и точного процесса моделирования.

Вычислитель – вычислительное устройство, осуществляющее управление приводами манипулятора.

Разрядность – параметр, определяющий точность решения траекторных задач прямой и обратной задач кинематики.

Цикл обмена – такт выдачи уставок на вычислительное устройство.

Цикл управления – такт выдачи управляющих сигналов на преобразователь импульсов.

Постоянная экспоненциального фильтра – ограничение скоростных перегрузок механических узлов робота.

Задачи приложения

Разработанное приложение позволяет решать следующие задачи моделирования манипуляционных роботов и может быть использовано для:

- ❏ изучения систем управления роботов и манипуляторов, где рассматриваются вопросы построения рабочих областей манипулятора при заданных характеристиках звеньев (длины звеньев, диапазоны углов поворота);
- ❏ изучения законов изменения обобщённых координат и их производных для различных траекторий рабочего органа манипулятора;
- ❏ решения задач статической и динамической точности роботов при выбранной структуре системы управления.

Для выполнения поставленных задач программный инструмент предоставляет следующие возможности:

- ❏ выбирать кинематическую схему манипулятора и задавать его параметры (конструктивные параметры и ограничения по координатам);
- ❏ строить рабочую область манипулятора в зависимости от выбранного типа робота (Декарт, Цилиндр, Скара, Колер);
- ❏ задавать параметры системы управления, то есть параметры двигателей и регуляторов, и вычислительного устройства;
- ❏ исследовать точность работы робота для различных режимов работы: позиционное и контурное.

Обзор используемых библиотек

Ключевыми библиотеками и программными расширениями языка программирования Python, лежащими в основе разработки структуры и функционала программного комплекса, являются **PyQt6**, **Numpy** и **MatPlotLib**.

- **PyQt6** – обширная библиотека и инструмент для реализации графического интерфейса приложения посредством добавления виджетов (окон, полей ввода).
- **Numpy** – расширение языка Python, предполагающее работу с многомерными массивами, векторами и позволяющее выполнять математические операции высокого уровня.
- **MatPlotLib** – универсальный и многогранный инструментарий для работы с визуализацией и построением графиков различной степени сложности. В случае работы над приложением, отвечает за построение рабочих областей и отрисовку перемещения схвата при позиционном и контурном управлении.

Библиотека **PyQt6** отвечает за реализацию интерфейса приложения посредством создания рабочих окон и наполнения их виджетами, то есть функциональными элементами. **PyQt6** оперирует тремя основными модулями **QtCore**, **QtGui**, **QtWidgets**.

QtCore – параметрический модуль, отвечающий за корректировку внутрипрограммных настроек каждого отдельного виджета. В контексте работы приложения позволяет выравнивать текстовые виджеты по центру.

QtGui – графический модуль, отвечающий за косметическую составляющую интерфейса, в нашем случае, иллюстрируя рабочие вкладки опознавательными ярлыками и выводя изображения кинематических схем роботов на экран. Более того данный модуль является связующим звеном между виджетами и внутрипрограммными функциями и работает по принципу: пользователь кликает на виджет → класс **QAction** инициирует действие → происходит вызов функции.

QtWidgets – модуль виджетов, являющейся основой всего приложения и реализующий его посредством классов. Класс **QApplication** формирует отдельное приложение, контролирует его запуск и завершение.

QMainWindow выводит стартовое окно, внутри которого классом **QWidget** распределяются элементы интерфейса – виджеты. **QGridLayout** условно представляет собой пространственную сетку, где для каждого виджета указывается порядок в виде номера ряда и столбца.

Из всего многообразия к задействованным виджетам относятся:

- **QPushButton** – командная кнопка для подтверждения действия.
- **QLabel** – текстовый ярлык для краткого описания.
- **QCheckBox** – список с полями для выбора одного из предложенных вариантов.
- **QMenuBar** – меню внутри основного окна с рядом функциональных вкладок.
- **QPixmap** – виджет с загруженным изображением.
- **QDoubleSpinBox** – поле для ввода и сохранения данных с регулируемыми ограничениями ввода.

Библиотека **Numpy** отвечает за математические операции внутри приложения и работу с массивами данных.

- **np.sin, np.cos, np.tg, np.pi** – нахождение углов поворота звеньев, работа с обобщёнными координатами и прочие тригонометрические операции.
- **np.nan_to_num** – преобразование неопределённых значений с плавающей запятой для повышения точности расчётов и результатов моделирования.
- **np.linspace, np.arange** – генерация списков с общим шагом для упрощённого формата хранения и использования данных.
- **np.concatenate** – работа с массивами данных и их объединением для их использования при построении рабочей области.

Библиотека **MatPlotLib** отвечает за визуализацию результатов моделирования роботов, то есть за визуализацию рабочих областей и построение контура перемещения схвата при позиционном и контурном управлении.

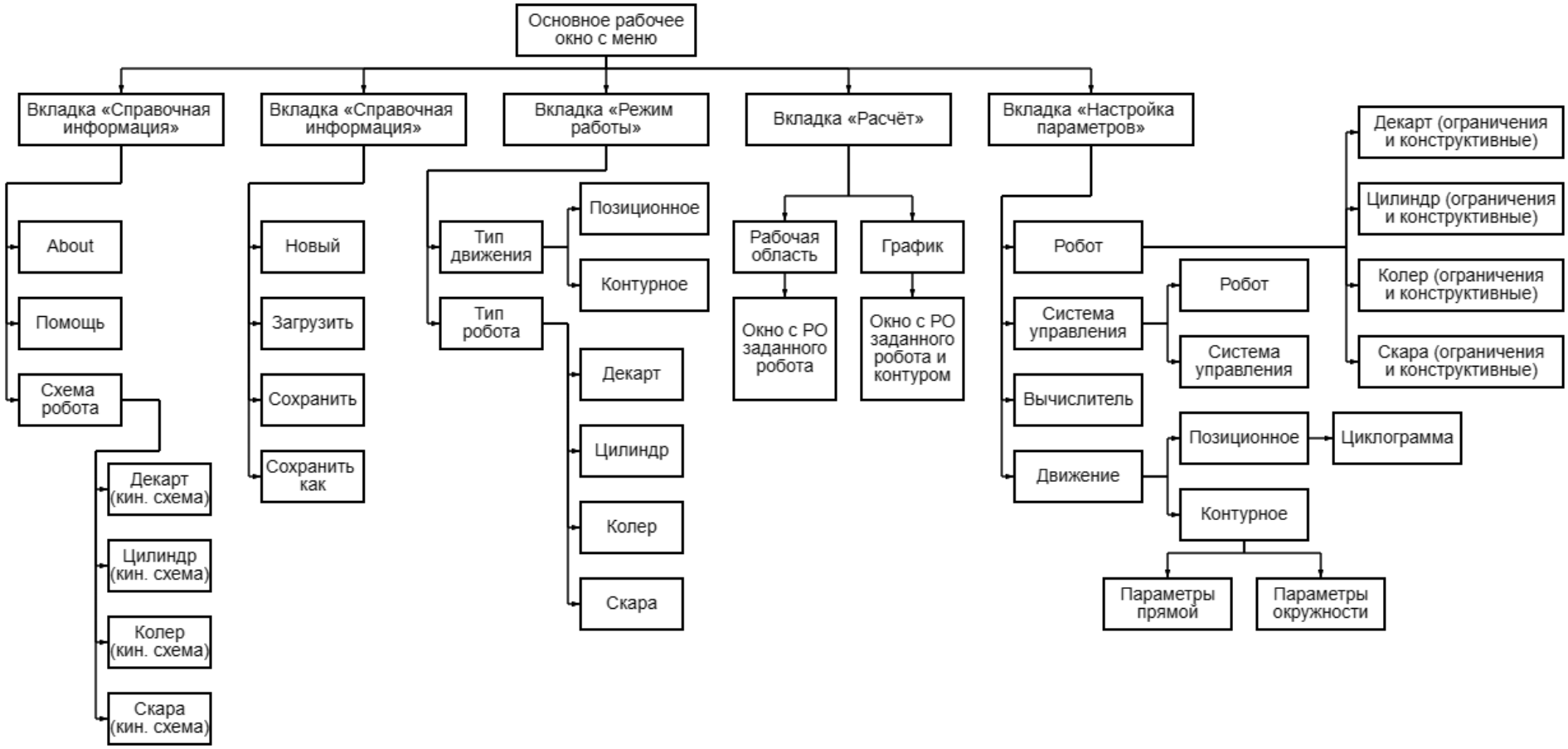
- **MatPlotLib.pyplot, MatPlotLib.figure** – визуализация окна с декартовой координатной сеткой и осями абсцисс и ординат.
- **MatPlotLib.animation** – перемещение рабочего органа в режиме реального времени при позиционном в зависимости от моментов времени и контурном управлении.
- **MatPlotLib.patches** – построение рабочих областей посредством объединения простых фигур в сложные.
- **MatPlotLib.path** – выявление списков координат для построения составных частей составных частей сложной фигуры с целью их последующего объединения в одну область.

Структура интерфейса приложения

Интерфейс приложения можно разделить на пять частей в соответствии с задачами, которые каждая из них выполняет, и количеством основных рабочих вкладок.

- **«Справочная информация»** – содержит информацию о разработчиках, кинематические схемы роботов, краткое описание комплекса.
- **«Сохранения»** – содержит функции: создать новый файл, загрузить существующий, сохранить и сохранить как, предполагает работу с загрузкой и сохранением файлов json с настройками параметров и текущего прогресса приложения.
- **«Режим работы»** – позволяет выбрать тип робота и тип его движения/управления, с которыми система работает в настоящий момент, представляет собой два окна с полями для выбора необходимого типа.
- **«Настройка параметров»** – отвечает за настройку параметров роботов, системы управления, вычислителя и движения, представляет собой ряд окон с вводом данных.
- **«Расчёт»** – отвечает за вызов окна с визуализацией рабочей области и отрисовку движения рабочего органа в зависимости от типа робота и его движения.

Наиболее подробный вариант структуры изображён на схеме ниже.


Основное рабочее окно с меню
Вкладка «Справочная информация»
Вкладка «Справочная информация»
Вкладка «Режим работы»
Вкладка «Расчёт»
Вкладка «Настройка параметров»
About
Помощь
Схема робота
Декарт (кин. схема)
Цилиндр (кин. схема)
Колер (кин. схема)
Скара (кин. схема)
Новый
Загрузить
Сохранить
Сохранить как
Тип движения
Позиционное
Контурное
Тип робота
Декарт
Цилиндр
Колер
Скара
Рабочая область
Окно с РО заданного робота
График
Окно с РО заданного робота и контуром
Робот
Декарт (ограничения и конструктивные)
Цилиндр (ограничения и конструктивные)
Колер (ограничения и конструктивные)
Скара (ограничения и конструктивные)
Система управления
Робот
Система управления
Вычислитель
Движение
Позиционное
Циклограмма
Контурное
Параметры прямой
Параметры окружности

Общие назначения функций

Функции вкладки **«Справочная информация»**

- **def __init__(self)** – функция инициализации класса для создания приложения, вызывает стартовое рабочее окно, добавляет меню-виджет с пятью основными вкладками и при помощи класса **QAction** соединяет их и их содержимое с последующими функциями.
- **def show1_about_window(self, checked)** – функция, вызывающая окно с краткой информацией о разработчиках.
- **def show1_robot_img_window(self, checked)** – функция, вызывающая окно с изображением кинематической схемы манипуляционного робота, с которым система работает в данный момент.
- **def show1_help_window(self, checked)** – функция, отвечающая за вызов окна с полной справочной информацией.

Функции вкладки **«Сохранения»**

- **def show2_new_window(self, checked)** – функция, сбрасывающая весь прогресс и возвращающая систему к настройкам по умолчанию.
- **def show2_load_window(self, checked)** – функция, загружающая существующие файлы сохранения в формате **json**.
- **def show2_save_window(self, checked)** – функция, сохраняющая все произведённые пользователем изменения в уже существующий файл сохранения, если такового нет, создаёт его.
- **def show2_saveas_window(self, checked)** – функция, создающая новый файл сохранения с выбором его имени и директории хранения.

Функции вкладки **«Режим работы»**

- **def show2_new_window(self, checked)** – функция, сбрасывающая весь прогресс и возвращающая систему к настройкам по умолчанию.
- **def show2_load_window(self, checked)** – функция, загружающая существующие файлы сохранения в формате **json**.
- **def show2_save_window(self, checked)** – функция, сохраняющая все произведённые пользователем изменения в уже существующий файл сохранения, если такового нет, создаёт его.
- **def show2_saveas_window(self, checked)** – функция, создающая

новый файл сохранения с выбором его имени и директории хранения.

Функции вкладки **«Настройка параметров»**

- **def show4_select_param_dec(self, checked)** – вызов окна с виджетами для указания конструктивных параметров робота-**Декарта**.
- **def show4_change_param_dec(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.
- **def show4_select_coord_dec(self, checked)** – вызов окна с виджетами для указания ограничений по координатам рабочей области робота-**Декарта**.
- **def show4_change_coord_dec(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.

- **def show4_select_param_cil(self, checked)** – вызов окна с виджетами для указания конструктивных параметров робота-**Цилиндра**.
- **def show4_change_param_cil(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.
- **def show4_select_coord_cil(self, checked)** – вызов окна с виджетами для указания ограничений по координатам рабочей области робота-**Цилиндра**.
- **def show4_change_coord_cil(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.

- **def show4_select_param_scr(self, checked)** – вызов окна с виджетами для указания конструктивных параметров робота-**Скары**.
- **def show4_change_param_scr(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.
- **def show4_select_coord_scr(self, checked)** – вызов окна с виджетами для указания ограничений по координатам рабочей области робота-

Скары.

- **def show4_change_coord_scr(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.

- **def show4_select_param_col(self, checked)** – вызов окна с виджетами для указания конструктивных параметров робота-**Колера**.
- **def show4_change_param_col(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.
- **def show4_select_coord_col(self, checked)** – вызов окна с виджетами для указания ограничений по координатам рабочей области робота-**Колера**.
- **def show4_change_coord_col(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.

- **def show4_select_engine(self, checked)** – вызов окна с виджетами для указания параметров двигателей (**J, n, Umax, Ku, Kq**).
- **def show4_change_eng(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.
- **def show4_select_reg(self, checked)** – вызов окна с виджетами для указания коэффициентов ПИД-регуляторов.
- **def show4_change_reg(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.
- **def show4_select_calc(self, checked)** – вызов окна с виджетами для параметров вычислителя.
- **def show4_change_calc(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.

- **def show4_select_pos_move(self, checked)** – вызов окна виджетами для заполнения циклограммы позиционного движения.
- **def show4_get_pos_values(self, change)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.
- **def show4_select_cont_line(self, checked)** – вызов окна виджетами для выбора и настройки траектории-прямой при контурном типе управления, содержит кнопку с вызовом окна с рабочей областью для размещения траектории внутри её границ.
- **def show4_change_line(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.
- **def show4_select_cont_circle(self, checked)** – вызов окна виджетами для выбора и настройки траектории-окружности при контурном типе управления, содержит кнопку с вызовом окна с рабочей областью для размещения траектории внутри её границ.
- **def show4_change_circle(self, change, key)** – вспомогательная к предыдущей функция для перезаписи параметров в соответствии с индексом виджета.

Функции вкладки **«Расчёт»**

- **def show5_graph(self, with_graph=False)** – масштабная функция, отвечающая за вызов окна с рабочей областью заданного типа робота (при **with_graph=False**) и построение пути перемещения рабочего органа (при **with_graph=True**). Внутри неё происходит расчёт размерности рабочих областей, реализация позиционного и контурного управления, выбор регулируется условными операторами **if-elif** в соответствии с системными параметрами, хранящимися в соответствующем словаре.
- **def show5_anim_pos(self, i)** – вспомогательная функция, добавляющая на окно с рабочей областью анимацию отрисовки контура при позиционном управлении в режиме реального времени.
- **def show5_anim_cont(self, i)** – вспомогательная функция, добавляющая на окно с рабочей областью анимацию отрисовки контура при контурном управлении в режиме реального времени, работает одинаково для прямой и окружности.

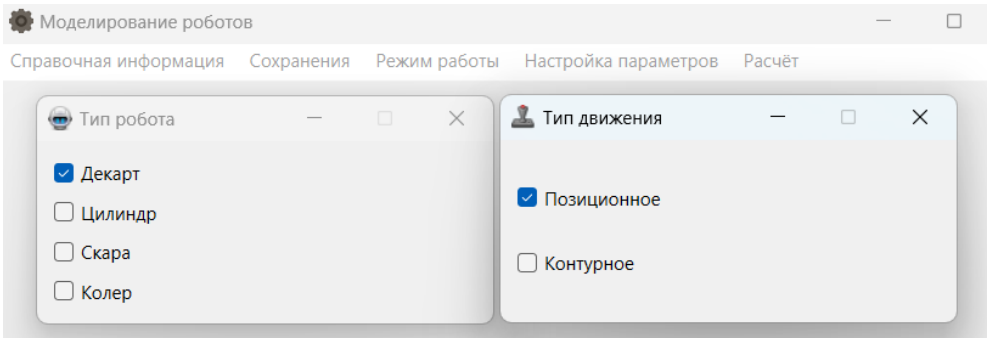
Пример выполнения работы на программном комплексе

Для изучения особенностей и процесса моделирования при **позиционном** управлении требуется провести ряд экспериментов для каждого доступного типа роботов. Для каждого робота (Декарт, Цилиндр, Колер, Скара) необходимо будет подобрать такое соотношение параметров, чтобы получить:

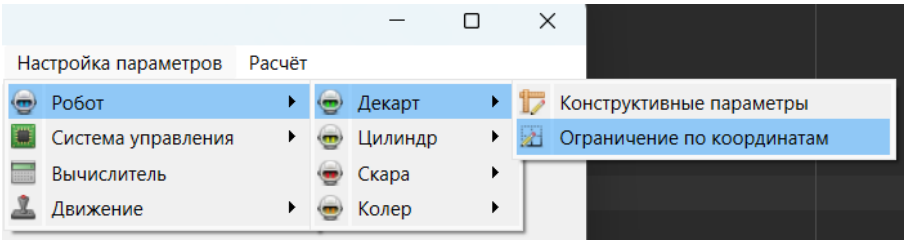
- ❏ Наиболее быстрый вариант отрисовки контура, где точностью можно пренебречь.
- ❏ Наиболее точный вариант отрисовки контура, где скоростью можно пренебречь.
- ❏ Оптимальный вариант отрисовки контура, где конечный результат будет оптимизирован по времени и точности.

Последовательность действий при этом следующая:

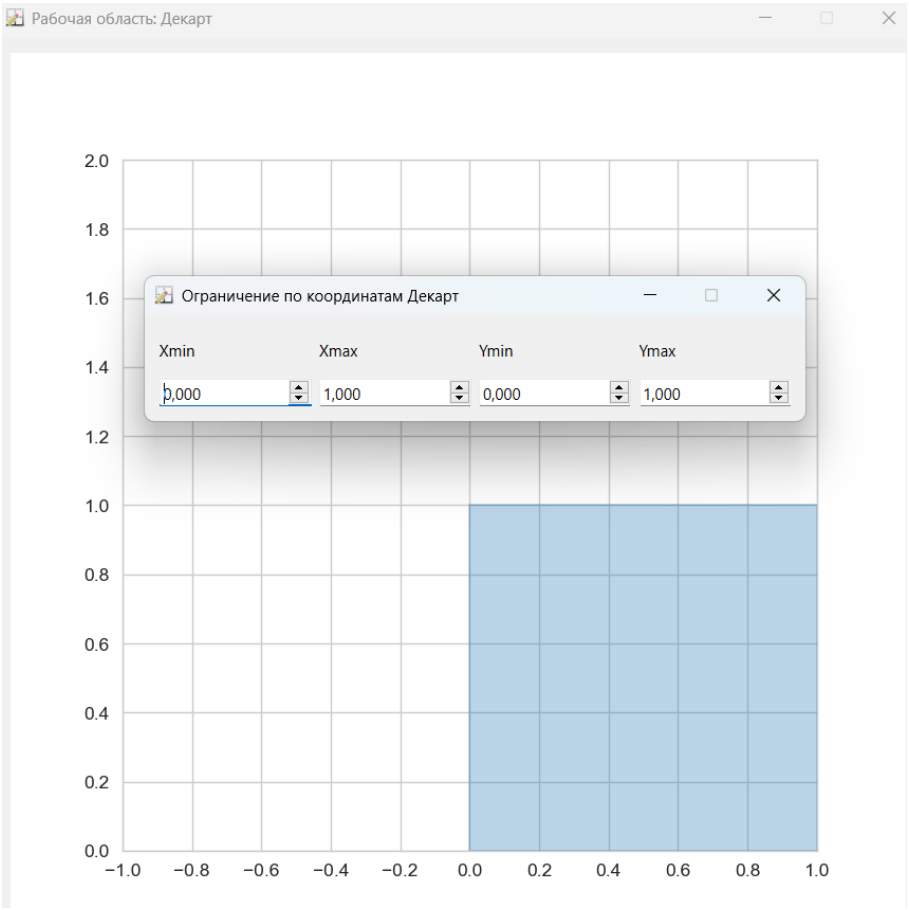
- ❏ Во вкладке **«Режим работы»** необходимо задать нужный тип робота и его движения.



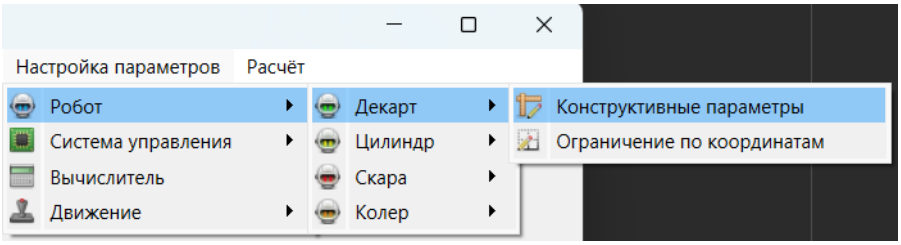
- ❏ Затем перейти к вкладке **«Настройка параметров»** и для выбранного типа робота задать **ограничения по координатам**.



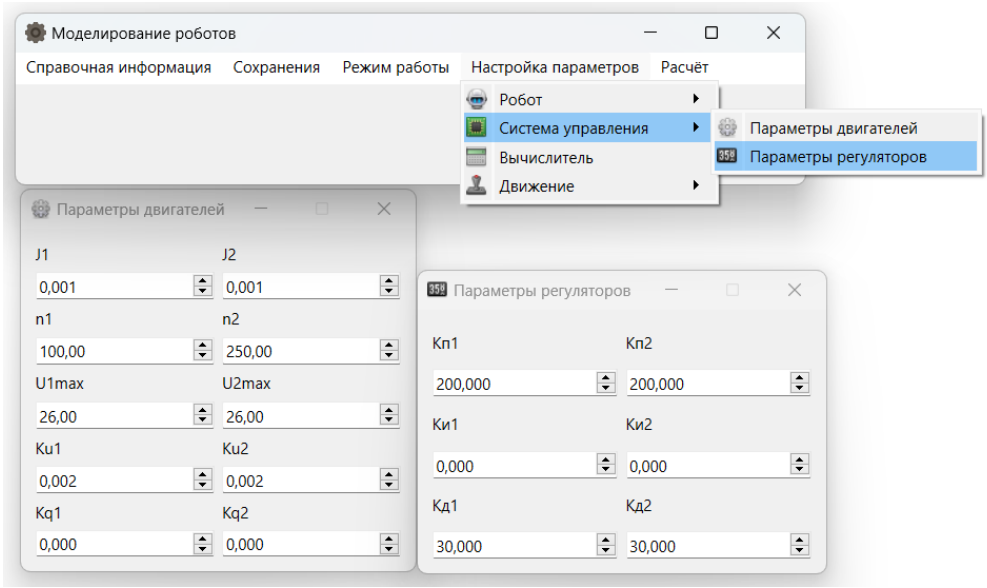
- Предварительно правильность построения рабочей области можно проверить во вкладках **«Расчёт» → «Рабочая область»**.



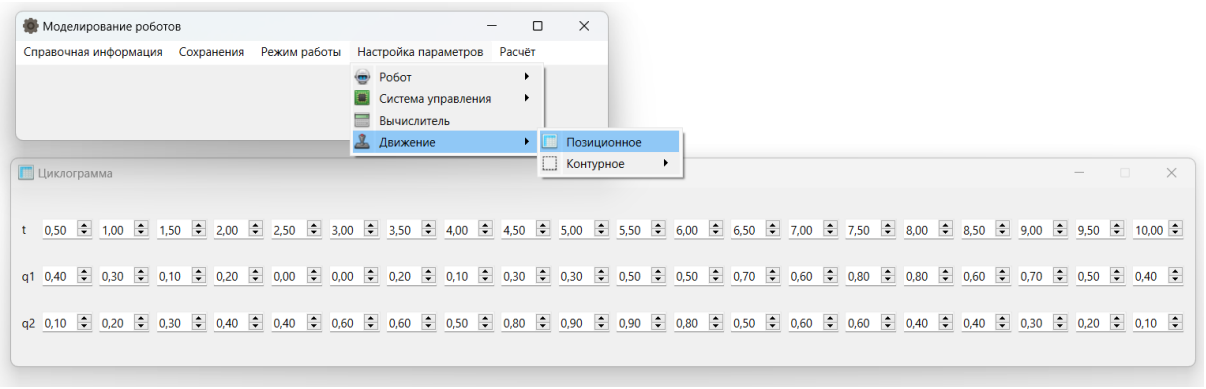
- Далее идёт настройка **конструктивных параметров** во вкладке с соответствующим названием, здесь указываются динамические характеристики заданного робота, при этом следует отметить, что длины звеньев влияют на построение границ рабочих областей некоторых роботов.



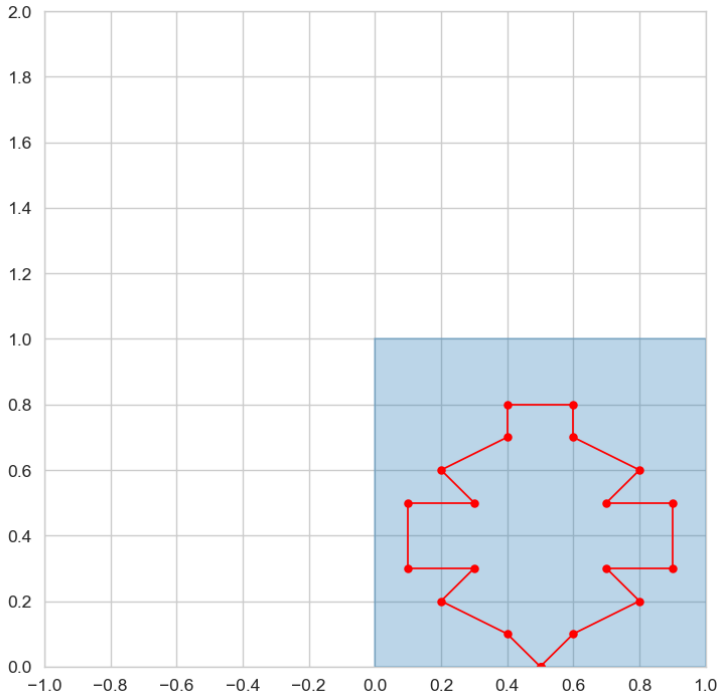
- Также необходимо обратить внимание на настройку параметров **системы управления**. Для этого в одноимённой вкладке нужно определиться с параметрами **регуляторов** и **двигателей**. Параметры регуляторов определяются экспериментальным путём. Необходимо подобрать пропорциональный, интегральный и дифференцирующий коэффициенты таким образом, чтобы контур движения схвата вышел наиболее точным без растяжений и астатизма.



- После первичного определения всех необходимых параметров следует перейти к заполнению **циклограммы** для реализации позиционного управления. В случае, если результат построения контура получится неудовлетворительным, всегда можно будет вернуться к соответствующей вкладке и перенастроить любой из присутствующих компонентов до тех пор, пока контур не будет соответствовать желаемому результату. Первый ряд предполагает указание моментов времени для соответствующих обобщенных координат, второй и третий – указание самих обобщенных координат.



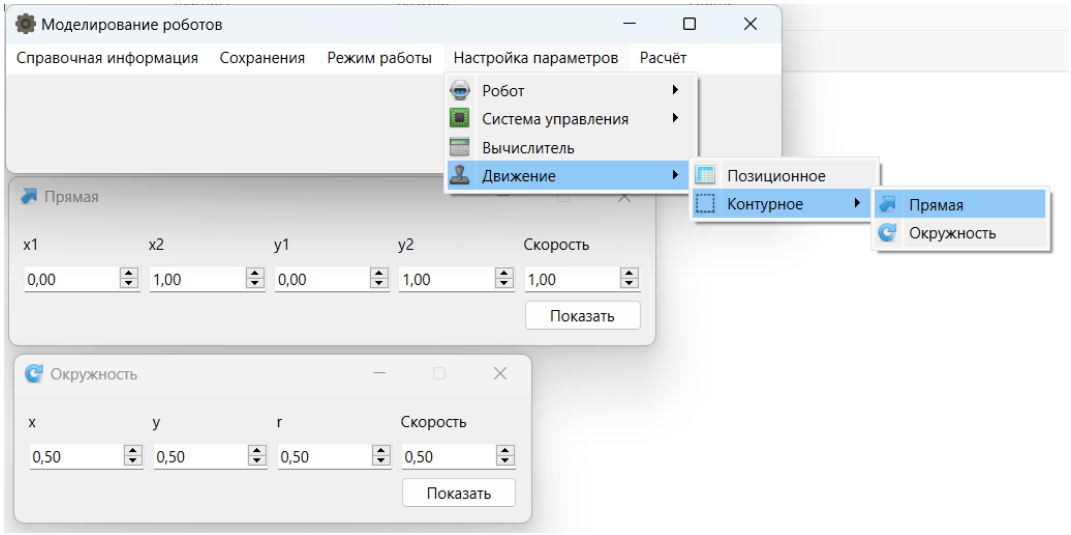
- По мере завершения этапа с подготовкой можно от вкладки **«Настройка параметров»** перейти к вкладке **«Расчёт»** и запустить процесс моделирования. При корректной настройке контур движения должен находиться внутри рабочей области



- Значения **циклограммы**, а конкретно моменты времени, необходимо скорректировать таким образом, чтобы получить три различных результата построения контура перемещения рабочего органа. То есть **быстрый**, **точный** и **оптимизированный** по времени и точности.

В качестве дополнительного задания можно попробовать провести оптимизацию моделирования каждого из роботов в режиме **контурного** управления. В этом случае при построении контура согласно заданной траектории необходимо будет соблюсти баланс между скоростью и параметрами самого робота.

Настройка всех необходимых параметров Декарта, Цилиндра, Колера или Скары, а также в отдельности их звеньев выполняется во вкладке с соответствующим названием, а для выбора типа траектории необходимо будет обратиться к подвкладке **«Движение»**, затем выбрать подпункт **«Контурное»**, где пользователю будет предоставлен выбор между прямой и окружностью. В каждом из всплывающих окон содержатся параметры самого контура и задаётся скорость отрисовки. Важно учесть, что для работы с данным режимом заранее нужно изменить тип управления роботом. Для этого следует обратиться к вкладке **«Режим работы»** и изменить движение с позиционного на контурное.



Перед началом моделирования необходимо настроить траекторию таким образом, чтобы её пределы не выходили за границы рабочей области. Для прямой пользователь может задавать стартовую и конечную точку, а для окружности – её центр и радиус. Для большего удобства программный инструмент предоставляет возможность предпросмотра через нажатие кнопки «Показать» прямо внутри окна с параметрами. Результаты изменений будут выведены в отдельное окно, что будет полезно для последующих доработок.

Когда все предварительные настройки будут завершены, можно приступать к процессу моделирования через вкладку «Расчёт». Как в случае с позиционным управлением, отрисовка установленного контура будет происходить в режиме реального времени в соответствующем окне. В случае контурного управления результат будет считаться удовлетворительным, если контур будет наиболее близок к условленной траектории, не будет растягиваться и выходить за пределы рабочей области и будет оптимизирован по скорости. Ряд подобных экспериментов должен быть произведён для каждого из доступных типов роботов.