Научно-исследовательский университет ИТМО
Факультет систем управления и робототехники

Курсовая работа по дисциплине
«Моделирование и управление движением роботов»

Четырёхколёсная платформа с роликонесущими колесами

Студент:

Зимирев Илья Александрович
гр. R4236c (283398)

Преподаватель:

Колюбин Сергей Алексеевич
д.т.н., профессор факультета СУиР

Санкт-Петербург, 2025

Содержание

Введение

Колесо Илона (Mecanum wheel) — роликонесущее колесо, разработанное в 1972 году, позволяющее транспорту осуществлять движение в любом направлении. Как утверждается в работе [1], колеса Илона широко применяются в различных областях, включая промышленность, медицину, образование, военное дело и научные исследования. В промышленности они используются, например, в вилочных погрузчиках, где требуется высокая маневренность для работы в ограниченных пространствах складов и заводов, а в медицинской сфере колеса Илона находят применение в инвалидных креслах, обеспечивая пациентам мобильность и удобство передвижения.

Рис. 1: Слева – погрузчик; справа – инвалидное кресло

Основное преимущество колеса Илона заключается в специальных роликах, установленных вдоль обода колеса под углом. Эти ролики способны вращаться независимо друг от друга и с помощью правильной конфигурации позволяют транспортному средству двигаться боковым ходом, диагонально, а также поворачиваться на месте без необходимости изменения направления движения. Данная особенность приводит к высокой манёвренности системы, недостижимой при использовании классических колёс.

Однако, следует отметить, что применение колеса Илона ограничено гладкими поверхностями и относительно невысокими скоростями. Несмотря на свою эффективность на таких условиях, на более неровных или непредсказуемых поверхностях и при высоких скоростях его производительность может снижаться. Тем не менее, в ситуациях, где требуется высокая степень маневренности и точности движения, колесо Илона остается одним из наиболее привлекательных вариантов.

Целью работы является моделирование работы мобильной платформы на роликонесущих колёсах. Задачи, необходимые для достижения поставленной цели, включают в себя проведение кинематического и динамического анализа системы, разработку планировщика траектории, синтез регулятора для управления движением и проведение численного моделирования.

1 Обзор существующих решений

В статье [2], авторы рассматривают геометрию и кинематику роликонесущего колеса. Это помогает им разработать модель, достаточно точную для дальнейших исследований. Однако они фокусируются на подвижной платформе с тремя колесами Илона, что не совсем соответствует теме данного исследования.

В другой работе [3] кинематический анализ дополняется динамическим анализом. Для четырехколесной платформы на роликонесущих колесах приводится уравнение Лагранжа, на основе которого решается прямая задача динамики, то есть находится зависимость координат платформы от приложенных моментов двигателей.

В статье [4] рассмотрена динамика более обобщенного варианта тележки на омниколесах, а именно свободная тележка с n роликонесущими колёсами на плоскости и сфере в неголономной постановке.

В еще одной работе [5] описывается математическая модель мобильной роботизированной платформы, которая проверяется с использованием данных с физической модели.

Авторы работы [6] разрабатывают глобальный планировщик пути и систему навигации для мобильного робота на роликонесущих колесах, проверяя эти методы как в симуляции, так и экспериментально.

Таким образом, рассмотрены научные работы, посвященные моделированию колеса Илона и его практическому применению. В этих исследованиях представлены различные решения для задач, связанных с моделированием и управлением движением мобильной роботизированной платформы с колесами Илона, включая анализ геометрии колеса, кинематики и динамики системы, верификацию математических моделей.

2 Построение модели

2.1 Описание системы

Колесо Илона состоит из набора k конгруэнтных роликов, расположенных симметрично вокруг корпуса колеса. Поверхность каждого ролика является частью поверхности вращения. Угол Y_i, образуемый между вектором линейной скорости ролика v_r и осью E_i, перпендикулярной оси колеса, обычно выбирается равным $\pm 45°$. Схему колеса Илона с введенными обозначениями можно видеть на рисунке 2 слева.

Особая конструкция колеса позволяет роботам совершать движения сложной механической природы. Стоит отметить, что существуют левостороннее и правостороннее колесо Илона, и принято их располагать таким образом, чтобы ось вращения верхнего ролика была направлена в центр платформы.

На рисунке 2 справа представлена кинематическая схема далее рассматриваемой в работе четырёхколёсной платформы с роликонесущими колёсами. Каждое из колёс приводится в движение отдельным двигателем, что обеспечивает платформе три степени свободы, необходимые для всенаправленного движения по ровной поверхности. Плоскость вращения колес всегда остается неподвижной относительно платформы. Центры масс колёс, как и центр масс платформы, совпадают с их геометрическими центрами.

На схеме использованы следующие обозначения:

- X, Y — оси системы координат, связанной с землёй
- X_R, Y_R — оси подвижной системы координат, связанной с её центром масс платформы
- θ — угол между системой координат земли и подвижной

системой координат робота

- v_x, v_y — линейные скорости платформы относительно системы координат земли

- ω_z — угловая скорость относительно вертикальной оси

- α_i — угол между осью X_R и отрезком OP_i, соединяющим центр масс платформы с центрои масс i-го колеса

- l_{iy}, l_{ix} — смещения оси колеса от центра платформы

- V_{ir} — линейная скорость ролика i-го колеса

- S_i, E_i — система координат i-го колеса, связанная с его центром масс

- Y_i — угол между вектором линейной скорости ролика и плоскостью колеса

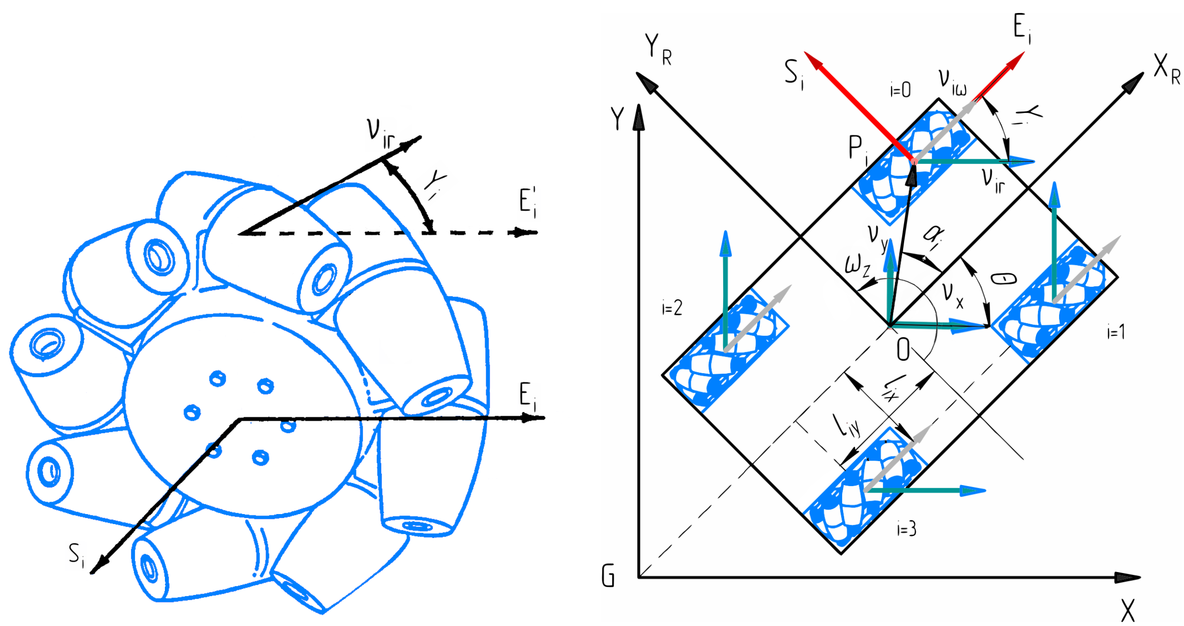


Рис. 2: Слева – схема колеса Илона; справа – схема мобильного робота

2.2 Кинематический анализ

Для проведения кинематического анализа платформы в первую очередь необходимо рассмотреть отдельно роликонесущее колесо. Для упрощения дальнейших расчётов было допущено отсутствие проскальзывания роликов относительно Земли. Так как каждый ролик зафиксирован непосредственно на корпусе, двигается вместе с ним и находится в контакте с землёй, угловая скорость колеса и ролика предполагаются равными. Пусть r_r — радиус ролика, тогда скорость движения ролика i-го колеса

$$V_{ir} = \frac{1}{\cos Y_i} \cdot r_r \cdot \omega_i, \quad i = 1...4,$$

где ω_i - угловая скорость i-го колеса.

Так как колесо двигается без проскальзывания, его скорость может быть определена как сумма проекций на оси E_i и S_i. Линейная скорость точки на ободе колеса (центра ролика) равна:

$$w_{E_i} = R_w \cdot \omega_i,$$

где R_w — радиус колеса.

Тогда проекции скорости на оси E_i и S_i соответственно равны:

$$V_{E_i} = w_{E_i} + V_{ir} \cdot \cos Y_i = R_w \cdot \omega_i + r_r \cdot \omega_i$$
$$V_{S_i} = w_{S_i} = V_{ir} \cdot \sin Y_i$$

Выражение для проекций линейной скорости в матричной форме может быть записано следующим образом:

$$\begin{bmatrix} V_{E_i} \\ V_{S_i} \end{bmatrix} = \begin{bmatrix} R_w & \cos Y_i \\ 0 & \sin Y_i \end{bmatrix} \cdot \begin{bmatrix} \omega_i \\ V_{ir} \end{bmatrix} = T_{p_i} \cdot \begin{bmatrix} \omega_i \\ V_{ir} \end{bmatrix}$$

Таким образом, матрица T_{p_i} является матрицей перехода в систему координат геометрического центра колеса. Так как система координат колеса является параллельным переносом системы координат мобильной платформы, линейная скорость i-го колеса проецируется без изменений.

$$\begin{bmatrix} V_{X_r i} \\ V_{Y_r i} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} V_{E_i} \\ V_{S_i} \end{bmatrix} = T_{p_i} \cdot \begin{bmatrix} \omega_i \\ V_{ir} \end{bmatrix}$$

Поскольку мобильная платформа не только движется поступательно, но и вращается, необходимо учитывать угловую скорость в проекции линейных скоростей:

$$\begin{bmatrix} V_{X_r i} \\ V_{Y_r i} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -l_{iy} \\ 0 & 1 & l_{ix} \end{bmatrix} \cdot \begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix} = T_0 \cdot \begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix},$$

где T_0 — матрица перехода от глобальной фиксированной системы координат к (X_R, Y_R).

Из двух предыдущих уравнений получим модель для решения обратной задачи кинематики:

$$\begin{bmatrix} R_w & \cos Y_i \\ 0 & \sin Y_i \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \omega_i \\ V_{ir} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -l_{iy} \\ 0 & 1 & l_{ix} \end{bmatrix} \cdot \begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix}$$

Учитывая, что $0 < |Y_i| < \frac{\pi}{2}$ и $\det T_{p_i} \neq 0$, можно получить зависимость между линейной скоростью платформы и угловой скоростью i-го колеса:

$$\begin{bmatrix} \omega_i \\ V_{ir} \end{bmatrix} = \begin{bmatrix} R_w & \cos Y_i \\ 0 & \sin Y_i \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & 0 & -l_{iy} \\ 0 & 1 & l_{ix} \end{bmatrix} \cdot \begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix}$$

Линейную и угловую скорости всей системы можно рассчитать, зная угловые скорости колес и соответствующие линейные скорости роликов:

$$\begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix} = T_{DK} \cdot \begin{bmatrix} \omega_i \\ V_{ir} \end{bmatrix}$$

По аналогии система для обратной кинематики:

$$\begin{bmatrix} \omega_i \\ V_{ir} \end{bmatrix} = T_{IK} \cdot \begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix},$$

где матрица преобразования T_{IK}, как было найдено выше, равна:

$$T_{IK} = T_{p_i}^{-1} T_0,$$

матрица обратного преобразования T_{DK} считается как псевдообратная от предыдущей, то есть

$$T_{DK} = (T_{IK}^T \cdot T_{IK})^{-1} \cdot T_{IK}^T$$

Тогда:

$$T_{IK} = \begin{bmatrix} R_w & \cos Y_i \\ 0 & \sin Y_i \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & 0 & -l_{iy} \\ 0 & 1 & l_{ix} \end{bmatrix}$$

Пусть li — расстояние от центра платформы до центра $i-$ колеса. Поскольку все колёса платформы одного размера, $l_{ix} = l_i \cdot \cos \alpha_i$ и $l_{iy} = l_i \cdot \sin \alpha_i$, то:

$$T_{IK} = \frac{1}{-R_w} \cdot \begin{bmatrix} \frac{\cos(-Y_i)}{\sin Y_i} & \frac{\cos(-Y_i)}{\sin Y_i} & \frac{l_i \cdot \sin(-\alpha_i - Y_i)}{\sin Y_i} \\ -\frac{r_{wheel}}{\sin Y_i} & -\frac{r_{wheel}}{\sin Y_i} & -\frac{l_i \cdot \sin(-\alpha_i) \cdot r_{wheel}}{\sin Y_i} \end{bmatrix}$$

Без учёта проскальзывания было получено окончательное выражение для обратной задачи кинематики:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{-1}{R_w} \cdot \begin{bmatrix} \frac{\cos(-Y_1)}{\sin Y_1} & \frac{\sin(-Y_1)}{\sin Y_1} & \frac{l_1 \cdot \sin(-Y_1 - \alpha_1)}{\sin Y_1} \\ \frac{\cos(-Y_2)}{\sin Y_2} & \frac{\sin(-Y_2)}{\sin Y_2} & \frac{l_2 \cdot \sin(-Y_2 - \alpha_2)}{\sin Y_2} \\ \frac{\cos(-Y_3)}{\sin Y_3} & \frac{\sin(-Y_3)}{\sin Y_3} & \frac{l_3 \cdot \sin(-Y_3 - \alpha_3)}{\sin Y_3} \\ \frac{\cos(-Y_4)}{\sin Y_4} & \frac{\sin(-Y_4)}{\sin Y_4} & \frac{l_4 \cdot \sin(-Y_4 - \alpha_4)}{\sin Y_4} \end{bmatrix} \cdot \begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix}$$

Путем взятия от T_{IK} псевдообратной матрицы была получена T_{DK}:

$$T_{DK} = \frac{1}{l_i^2 + 1} \times$$
$$\times \begin{bmatrix} -\frac{1}{2} \cdot (-l_i^2 \cdot \sin 2\alpha_i) \cdot R_w & \frac{1}{2} \cdot l_i^2 \cdot \sin(Y_i + 2\alpha_i) - \frac{1}{2} \cdot \sin(-Y_i) \cdot l_i^2 - \sin(-Y_i) \\ -\frac{1}{2} \cdot R_w \cdot (l_i^2 - l_i^2 \cdot \cos 2\alpha_i + 2) & -\frac{1}{2} \cdot l_i^2 \cdot \cos(Y_i + 2\alpha_i) + \frac{1}{2} \cdot \cos(-Y_i) \cdot l_i^2 + \cos(-Y_i) \\ \cos\alpha_i \cdot l_i \cdot R_w & \cos(\alpha_i + Y_i) \cdot l_i \end{bmatrix}$$

Подставив T_{DK} в систему, приведённую в начале параграфа, можно получить решение прямой задачи кинематики.

2.3 Динамический анализ

Динамический анализ включает в себя вычисление связи между приложенными к системе силами и ускорением системы. В данной работе это было сделано через второй закон Ньютона. Для платформы с массой m и моментом инерции I его можно записать в виде:

$$F_{X_r} = m\dot{V}_{X_r} \qquad F_{Y_r} = m\dot{V}_{Y_r} \qquad M_z = I\dot{\omega}_z$$

где F_{X_r}, F_{Y_r} – силы, действующие вдоль соответствующих осей, M_z – момент силы, поворачивающий тележку вокруг вертикальной оси Z.

Для нахождения момента инерции тележка была упрощена до однородного параллелепипеда размеров l_x, l_y, l_z с четырьмя однородными дисками радиуса R_w и толщиной d_w, закреплённых в нижних углах платформы.

Момент инерции платформы в таком случае равен:

$$I_p = \frac{1}{12}m_p(l_x^2 + l_y^2),$$

где m_p – масса платформы.

Момент инерции колёс относительно центра масс платформы по теореме Штейнера равен:

$$I_w = I_{w,c} + m_w l_w^2,$$

где $I_{w,c}$ – момент инерции колеса относительно оси, проходящей через центр масс колеса, m_w – масса колеса, l_w – расстояние от центра масс тележки до колеса.

$$I_{w,c} = \frac{1}{4}m_w R_w^2 + \frac{1}{12}m_w d_w^2,$$

$$l_w = \sqrt{l_x^2 + l_y^2},$$

$$I_w = \frac{1}{4} m_w R_w^2 + \frac{1}{12} m_w d_w^2 + m_w (l_x^2 + l_y^2),$$

Суммарный момент инерции платформы на омниколесах равен:

$$I = I_p + \sum_{i=1}^{n} I_w,$$

$$I = \frac{1}{12} m_p (l_x^2 + l_y^2) + n(\frac{1}{4} m_w R_w^2 + \frac{1}{12} m_w d_w^2 + m_w (l_x^2 + l_y^2)),$$

Каждое колесо обладает силой тяги F_i, как представлено на рисунке 3.

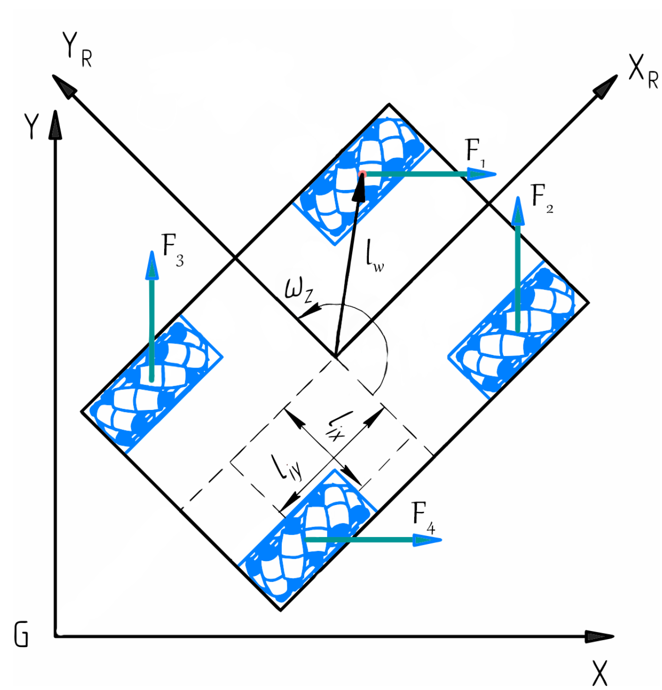


Рис. 3: Силы тяги

Силы F_{X_r}, F_{Y_r} и момент сил M_z складываются из сил тяги колёс следующим образом:

$$F_{X_r} = F_1 cos Y_1 + F_2 cos Y_2 + F_3 cos Y_3 + F_4 cos Y_4,$$

$$F_{Y_r} = F_1 sinY_1 + F_2 sinY_2 + F_3 sinY_3 + F_4 sinY_4,$$

$$M_z = F_1 l_w \sin(Y_1 + \beta_1) + F_2 l_w \sin(Y_2 + \beta_2) + F_3 l_w \sin(Y_3 + \beta_3) + F_4 l_w \sin(Y_4 + \beta_4),$$

либо, в матричном виде:

$$\begin{bmatrix} F_{X_r} \\ F_{Y_r} \\ M_z \end{bmatrix} = \begin{bmatrix} \cos Y_1 & \cos Y_2 & \cos Y_3 & \cos Y_4 \\ \sin Y_1 & \sin Y_2 & \sin Y_3 & \sin Y_4 \\ l_w \sin(Y_1 + \beta_1) & l_w \sin(Y_2 + \beta_2) & l_w \sin(Y_3 + \beta_3) & l_w \sin(Y_4 + \beta_4) \end{bmatrix} \cdot \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_3 \end{bmatrix},$$

где $\beta_i = \pi/2 - \alpha_i = \arcsin(l_{yi}/l_w)$

Подставив все полученные результаты во второй закон Ньютона, представленный выше, можно получить динамическую модель системы.

3 Планирование движения

Для создания маршрута движения был использован алгоритм A*, один из самых популярных алгоритмов для поиска пути. Это метод поиска кратчайшего пути в графе, который сочетает точную стоимость уже пройденного маршрута $g(n)$ с эвристической оценкой оставшейся дистанции до цели $h(n)$, суммируя их в функцию $f(n) = g(n) + h(n)$ для приоритизации узлов. После использования алгоритма A* мы получаем набор точек в виде (x, y) от начальной до конечной точки. Для получения угла поворота робота θ мы берём соседние точки маршрута и находим угол между ними, тем самым мобильная платформа всегда направлена вдоль движения. Стоит отметить, что рассматриваемая платформа с роликонесущими колёсами позволяет движение в любую сторону без смены ориентации, а значит вычисление угла не является обязательным.

Для планирования траектории (маршрута с временными метками) используется алгоритм, который вычисляет кумулятивное расстояние вдоль маршрута, затем, используя заданные параметры максимальной скорости v_{max} и максимального ускорения a_{max}, рассчитывает временные метки для каждой точки пути с учетом трех фаз движения: ускорения, равномерного движения и замедления. Также, для более точной траектории определяется кривизна пути и на её основе определяется локальное ограничение скорости

$$v_{lim}(t) = min(v_{max}, \sqrt{a_{max} * R(i)}),$$

где $R(i)$ – радиус кривизны в точке i.

Это позволяет траектории замедляться на поворотах. После расчета временных меток производится сплайн-интерполяция, которая позволяет получить плавную траекторию с временным шагом.

Для удобства использования на вход планировщику траектории подаётся карта, на которой чёрным $rgb(0, 0, 0)$ обозначены препятствия, зелёным $rgb(0, 255, 0)$ обозначена начальная точка, красным $rgb(255, 0, 0)$ обозначена конечная точка, а синим $rgb(0, 0, 255)$ обозначены промежуточные точки. Также, для учёта ширины платформы используется расширение препятствий на половину ширины робота.

4 Слежение за траекторией

Поскольку мы не используем оформленную математическую модель мобильной платформы, самым удобным способом управления является ПИД-регулятор. Этот алгоритм управления позволяет абстрагироваться от математической модели и настраивать только коэффициенты. При этом ПИД-регулятор позволяет добиться достаточно высокой точности и скорости слежения.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt},$$

где $u(t)$ – управляющее воздействие, K_p, K_i, K_d – коэффициенты пропорционального, интегрального и дифференциального усиления соответственно, $e(t)$ – ошибка регулирования

На вход регулятор получает ошибки по координатам (x, y, θ), а на выходе мы получаем u_x, u_y, u_{theta}, после чего мы используем матрицу, псевдообратную от указанной в разделе Динамический анализ для получения силы тяги каждого колеса. Стоит отметить, что целиком наш регулятор состоит из трёх ПИД-регуляторов, по одному на каждую компоненту u. В будущем, для большего приближения к реальности, будет возможность добавить математическую модель двигателя. Тогда u_x, u_y, u_{theta} будут перерасчитываться в напряжение на каждом отдельном двигателе, однако для данной работы это нам показалось излишним. Для базового поиска коэффициентов регуляторов использовался метод Циглера-Никольса, затем полученные коэффициенты уточнялись опытным путём. Итоговые полученные коэффициенты указаны в Приложении к работе.

5 Моделирование

Для выполнения численного моделирования были выбраны следующие числовые значения:

- размеры тележки $l_x = 0,1$ м, $l_y = 0,2$ м
- масса тележки $m_p = 0,5$ кг
- размеры колеса $R_w = 0,05$ м, $d_w = 0,02$ м
- масса колеса $m_w = 0,1$кг
- углы между векторами линейной скорости роликов и плоскостью колёс $Y_1 = Y_4 = 45°$, $Y_2 = Y_3 = -45°$

Моделирование слежения тележки за траекторией было выполнено в пакете Simulink программы MATLAB. Динамика системы и регулятор были созданы с помощью блоков Simulink, карты подавались в виде изображений. Схема Simulink и программный код MATLAB находятся в Приложении к работе.

Первая карта представляет собой случайно расставленные прямоугольники в качестве препятствий и случайно расставленные начальная, конечная и промежуточные точки.

Как мы видим по Рис. 6, отклонения составляют примерно 0.1 как для линейных осей, так и для угла поворота. Для нас это достаточно хороший результат, поэтому более сложные алгоритмы управления, например, линеаризация обратной связью, не применялись. Для лучшего понимания и отображения возможной статической ошибки время моделирования сделано на 10% больше времени, которое определил планировщик движения. Всё это время сверх необходимого на вход системы подаётся последняя точка маршрута.

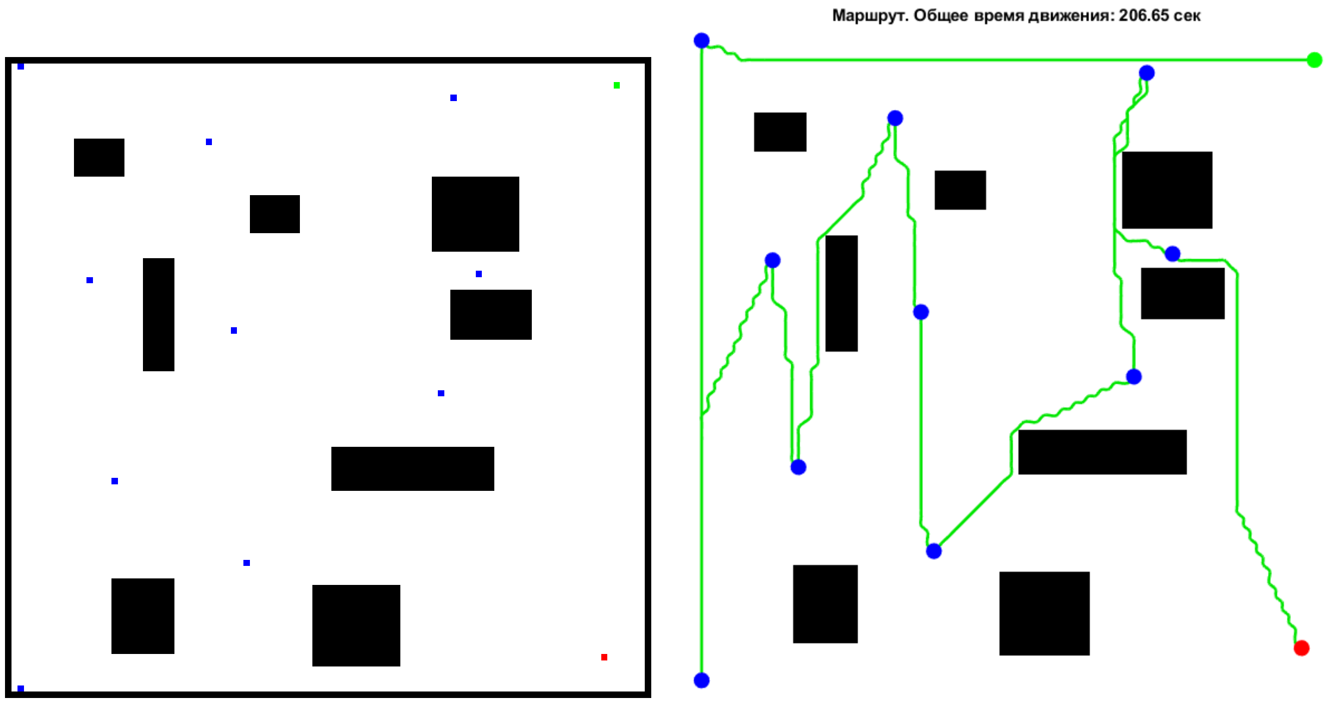


Рис. 4: Слева – карта; справа – полученный маршрут

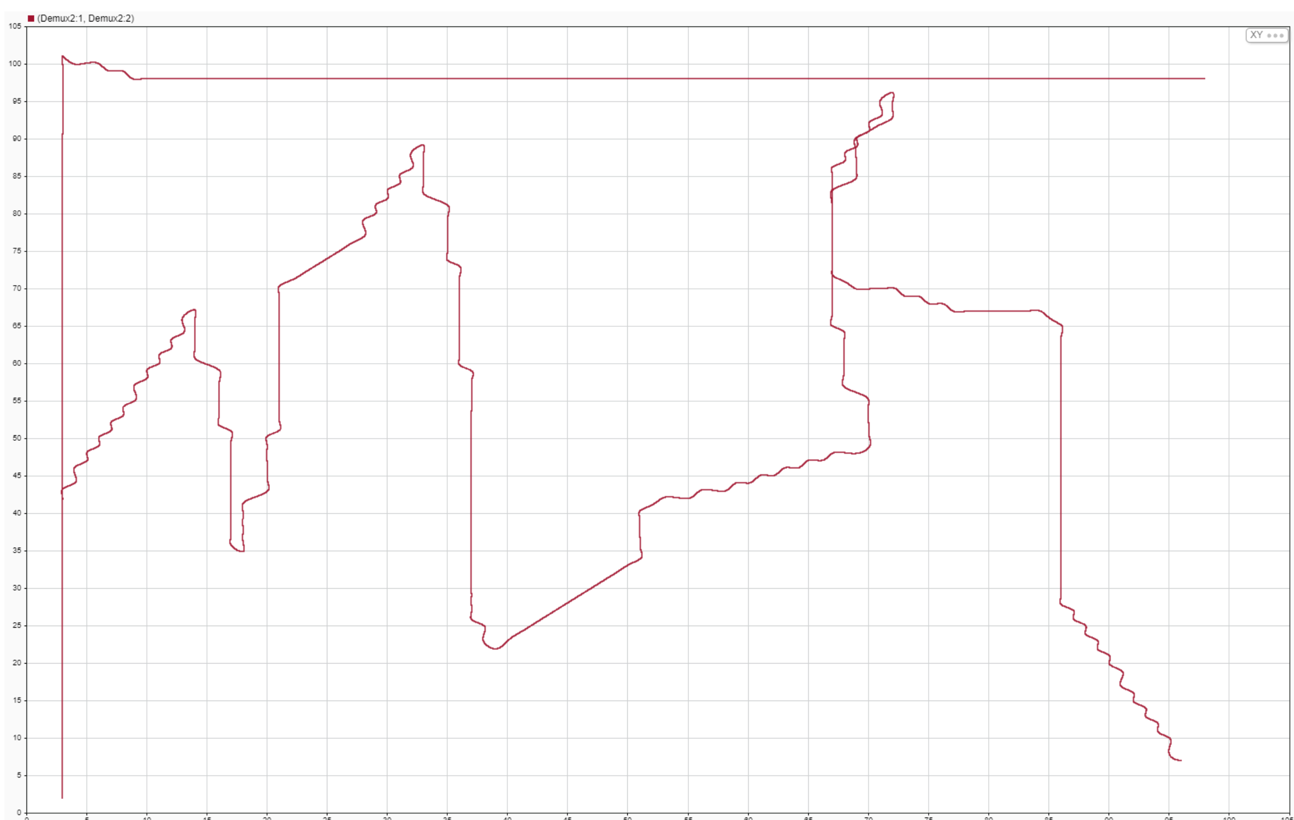


Рис. 5: Траектория движения робота в симуляции

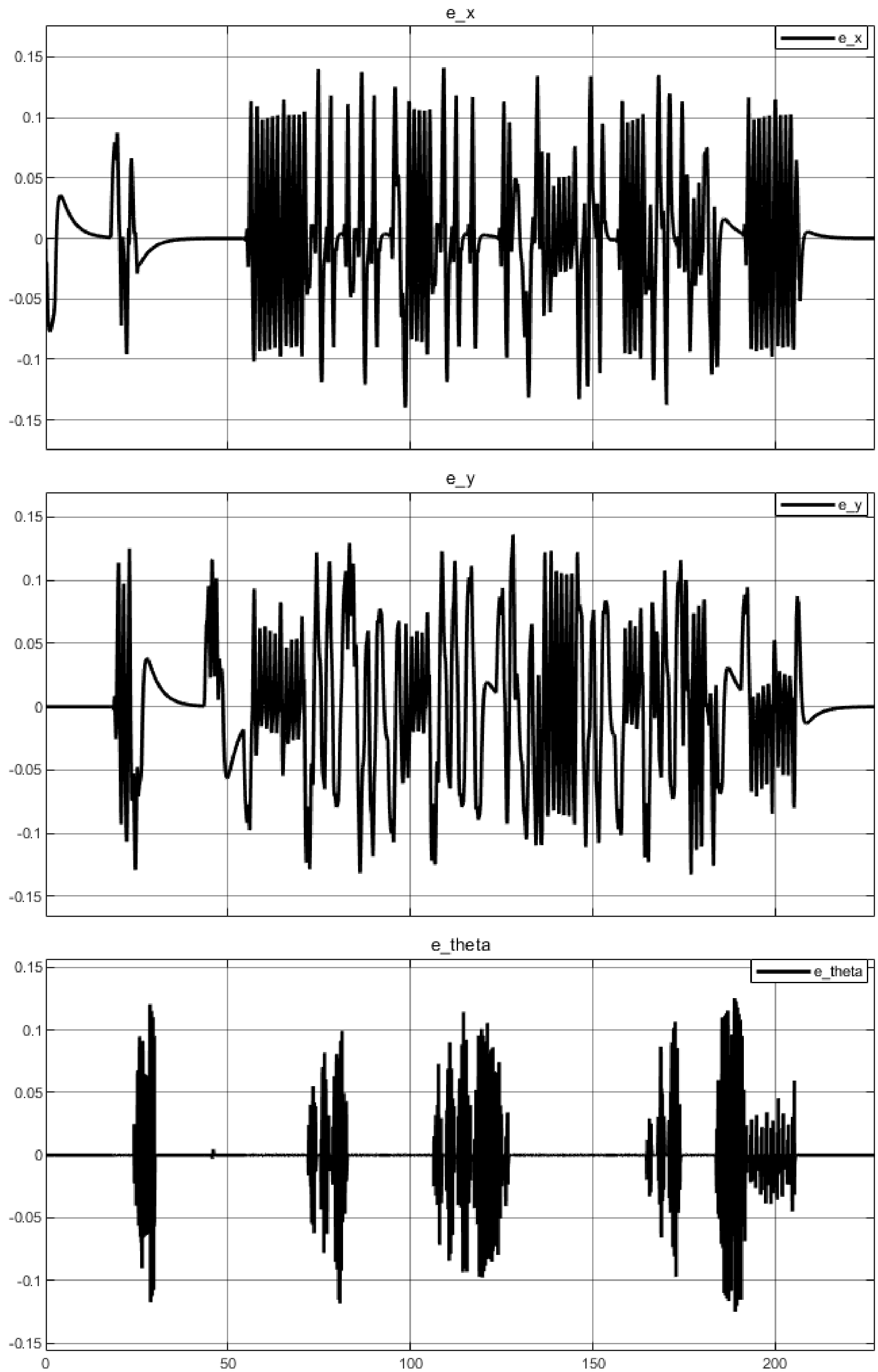
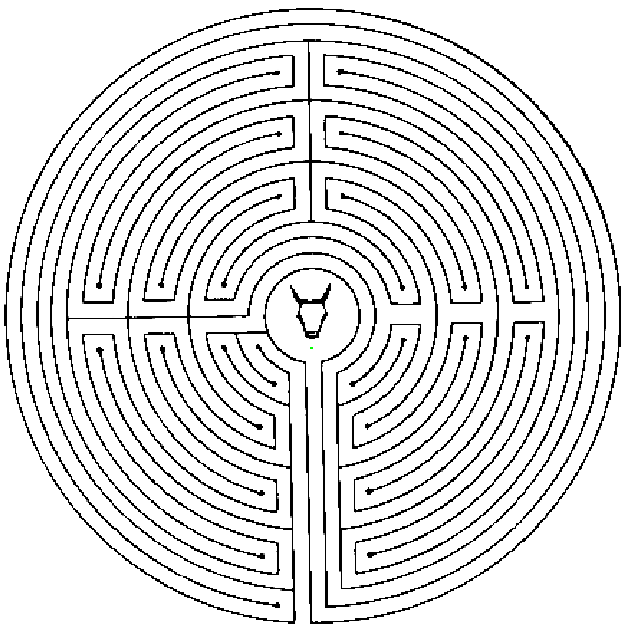


Рис. 6: Ошибки по x (сверху), y (по центру) и θ (снизу)

В качестве второй карты был выбран известный лабиринт Минотавра. В нём уже отсутствуют промежуточные точки, планировщику необходимо найти выход.



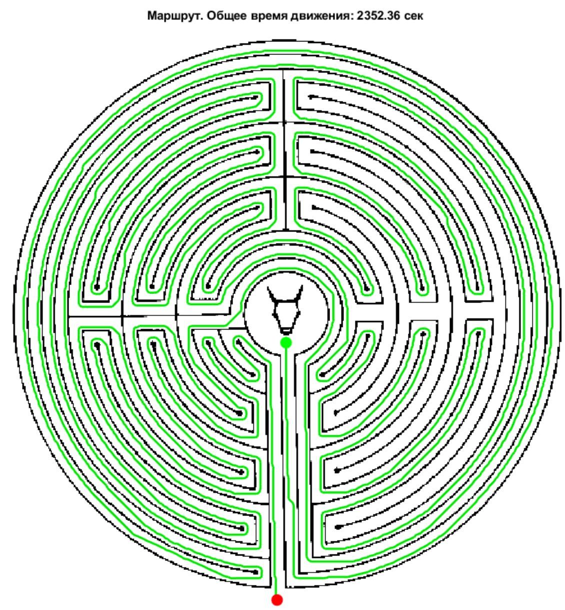


Рис. 7: Слева – лабиринт; справа – полученный маршрут

Сложность этой траектории заключается в постоянном угловом повороте, поэтому особо критично смотреть на ошибку по θ. Однако, как видно по Рис. 9 данная ошибка в пике составляет всего 0.01 радианты или $\approx 0.5\ ^\circ$. Также на Рис. 7 и Рис. 8 видно, что маршрут немного угловат, хотя сами стены лабиринта дуги. Это происходит из-за того, что алгоритм пытается найти самый короткий путь, и из-за этого старается максимально срезать углы.

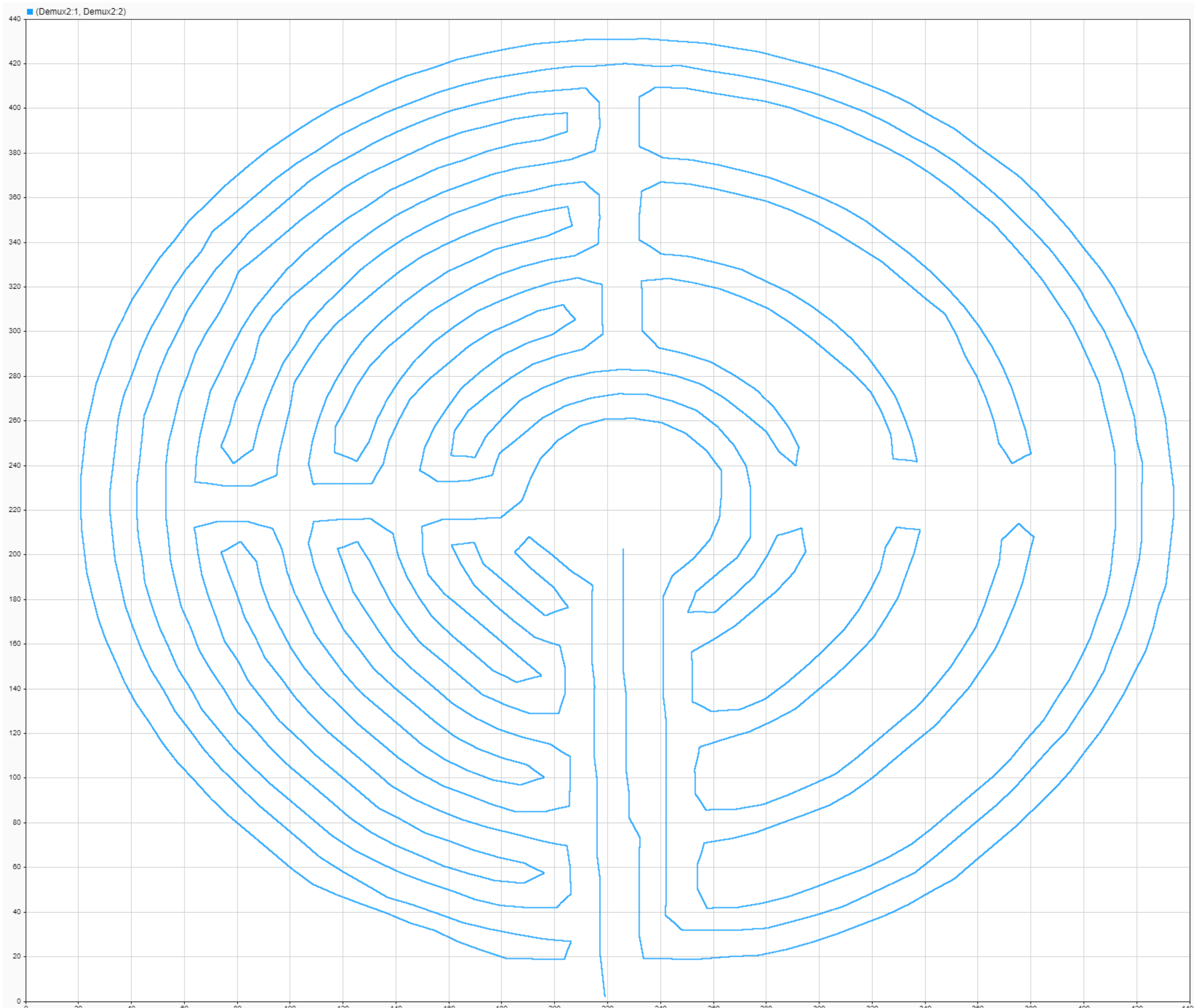


Рис. 8: Траектория движения робота в симуляции

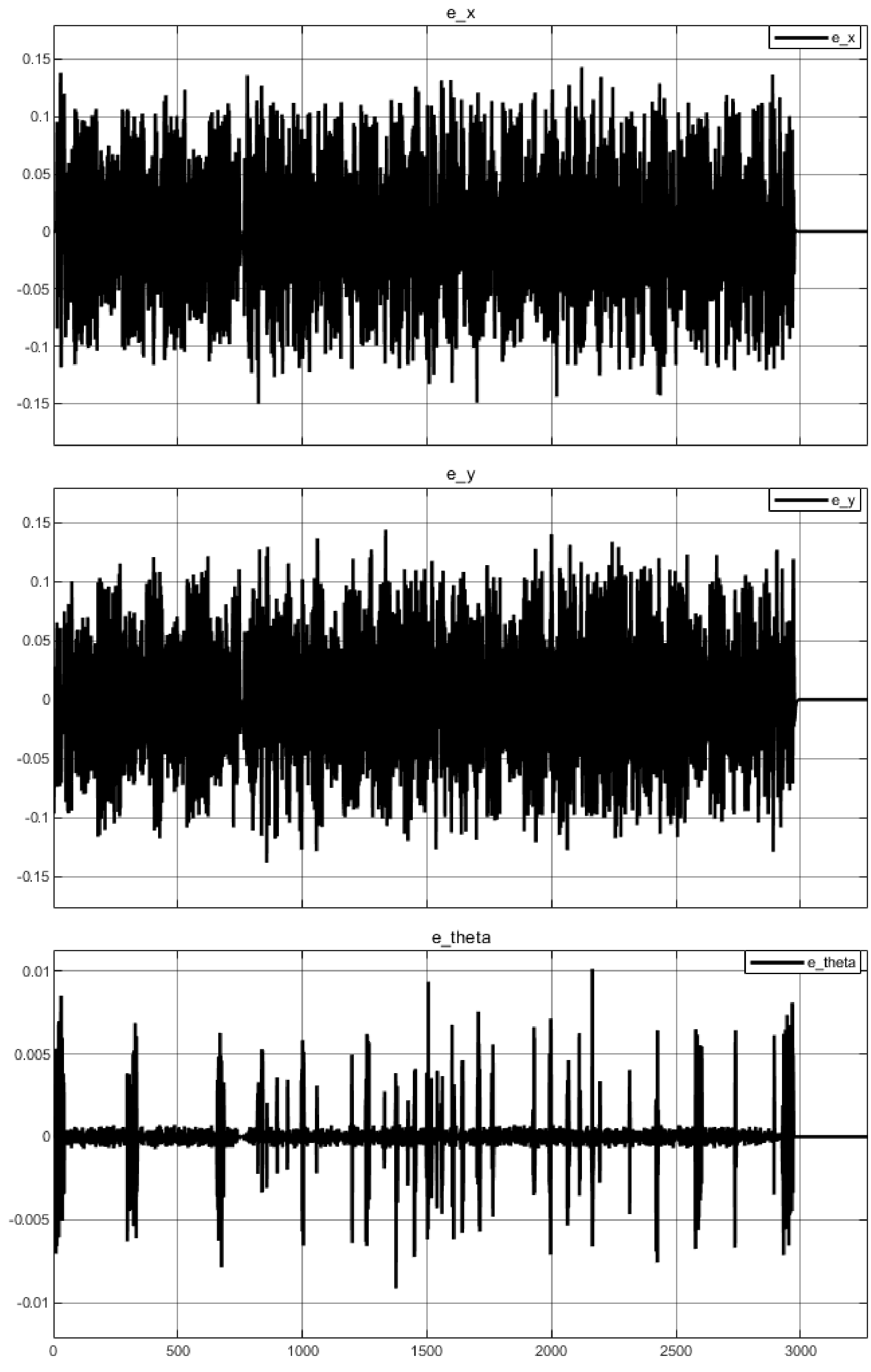


Рис. 9: Ошибки по x (сверху), y (по центру) и θ (снизу)

Заключение

В данной работе проведено комплексное исследование по моделированию и управлению движением мобильной платформы с роликонесущими колесами. В ходе исследования выполнен обзор существующих решений, подробно описана конструкция системы, проведён кинематический анализ с решением прямой и обратной задач кинематики, а также представлен динамический анализ на основе второго закона Ньютона. Разработаны алгоритмы планирования маршрута и слежения за траекторией с использованием метода A* и ПИД-регуляторов, что позволило добиться высокой точности позиционирования и малых ошибок по координатам и угловой ориентации. Результаты численного моделирования в MATLAB/Simulink подтвердили корректность выбранных подходов и параметров управления.

Проведённый анализ кинематики и динамики платформы показал, что разработанная модель способна адекватно описывать движение системы. Использование алгоритмов планирования пути и ПИД-регуляторов обеспечило стабильное и точное слежение за заданной траекторией, что подтверждается результатами симуляций. Для дальнейшего улучшения полученной математической модели и симуляции стоит учитывать двигатели, проскальзывания, трения в колёсах, детализировать строение и уточнять параметры, разрабатывать более продвинутые алгоритмы управления, например, оптимальные для экономии максимального количества заряда мобильной платформы.

Список литературы

[1] Florentina Adascalitei and Ioan Doroftei. Practical applications for mobile robots based on mecanum wheels - a systematic survey. *Romanian Review Precision Mechanics, Optics and Mechatronics*, pages 21–29, 01 2011.

[2] A. Gfrerrer. Geometry and kinematics of the mecanum wheel. *Computer Aided Geometric Design*, 25(9):784–791, 2008. Classical Techniques for Applied Geometry.

[3] Felix Becker, Olga Bondarev, Igor Zeidis, Klaus Zimmermann, Mohamed Abdelrahman, and Boris Adamov. An approach to the kinematics and dynamics of a four-wheeled mecanum vehicles. *Problems of Mechanics*, 2:27–37, 01 2014.

[4] И. С. Мамаев А. В. Борисов, А. А. Килин. Тележка с омниколёсами на плоскости и сфере. *Нелинейная динамика*, 7(4):785–801, 2011.

[5] Nkgatho Tlale and Mark de Villiers. Kinematics and dynamics modelling of a mecanum wheeled mobile platform. pages 657–662, 2008.

[6] Ching-Chih Tsai, Ching-Zu Kuo, Chun-Chieh Chan, and Xiao-Ci Wang. Global path planning and navigation of an omnidirectional mecanum mobile robot. pages 85–90, 2013.

Приложение

Коэффициенты ПИД-регуляторов, полученные для симуляции. Стоит отметить, что при изменении параметров модели, коэффициенты могут работать значительно хуже, вплоть до нестабильности системы.

$$K_p(x) = K_p(y) = 20, K_p(\theta) = 1000$$

$$K_i(x) = K_i(y) = 5, K_i(\theta) = 10$$

$$K_d(x) = K_d(y) = 10, K_d(\theta) = 100$$

На рисунке 10 представлена схема моделирования слежения платформы на омниколёсах за траекторией, на листинге 1 – алгоритм планировщика траектории, на листинге 2 – задание численных параметров и некоторые промежуточные подсчёты.

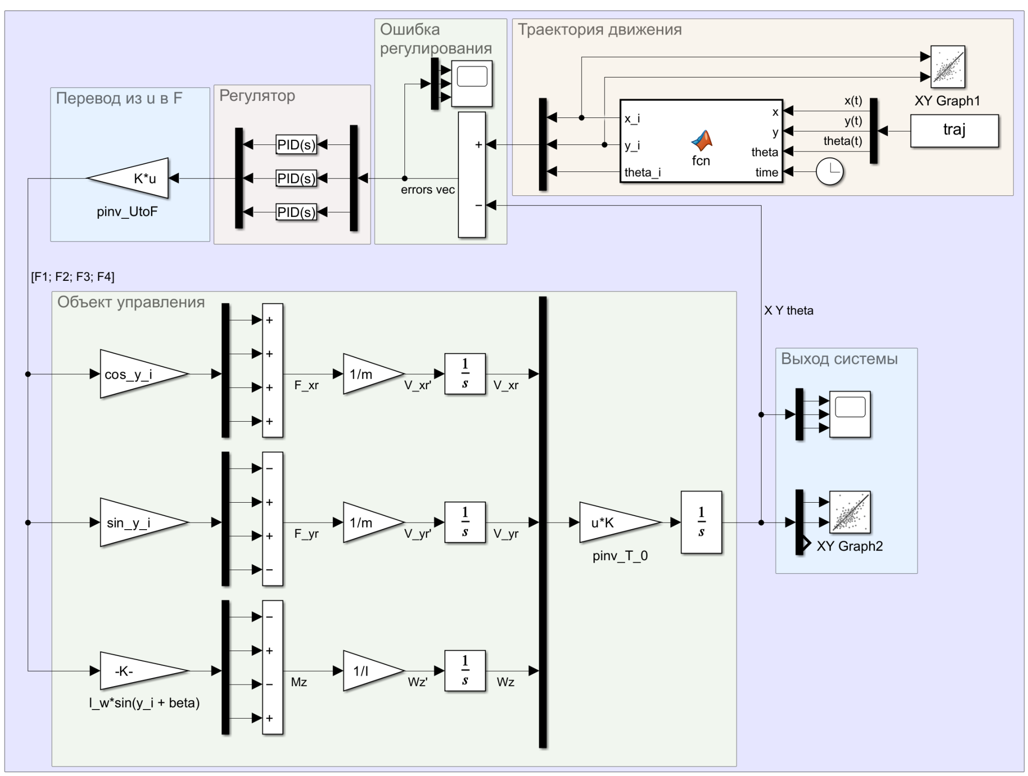


Рис. 10: Схема моделирования Simulink

Листинг 1: Алгоритм планирования траектории A*

```
function trajectory = planRoute(mapFileName, v_max, a_max, robotWidth)
    % Чтение изображения и определение маркерных точек
    I = imread(mapFileName);
    [imgHeight, imgWidth, ~] = size(I);

    % Стартовая точка зелёный( [0,255,0])
    startMask = (I(:,:,1)==0 & I(:,:,2)==255 & I(:,:,3)==0);
    [rowS, colS] = find(startMask, 1);
    if isempty(colS)
        error('Начальная точка не найдена');
    end
    startPt = [imgHeight - rowS + 1, colS];

    % Конечная точка красный( [255,0,0])
    goalMask = (I(:,:,1)==255 & I(:,:,2)==0 & I(:,:,3)==0);
    [rowG, colG] = find(goalMask, 1);
    if isempty(colG)
        error('Конечная точка не найдена');
    end
    goalPt = [imgHeight - rowG + 1, colG];

    % Промежуточные точки синий( [0,0,255])
    interMask = (I(:,:,1)==0 & I(:,:,2)==0 & I(:,:,3)==255);
    [rowI, colI] = find(interMask);
    interPts = [];
    if ~isempty(colI)
        interPts = [imgHeight - rowI + 1, colI];  % [x, y]
        interPts = sortrows(interPts, 2); % сортировка по y
    end

    % Список ключевых точек
    waypoints = [startPt; interPts; goalPt];

    %Создание occupancy grid с расширением препятствий
    obsMask = (I(:,:,1)==0 & I(:,:,2)==0 & I(:,:,3)==0);
```

```matlab
    occMat = double(obsMask);

    se = strel('disk', ceil(robotWidth/2));
    inflatedOccMat = imdilate(occMat, se);

    inflatedOccMatFlipped = flipud(inflatedOccMat);

    resolution = 1;
    mapObj = occupancyMap(inflatedOccMatFlipped, resolution);

    %Планирование пути между ключевыми точками с помощью A*
    fullPath = [];
    for i = 1:size(waypoints,1)-1
        planner = plannerAStarGrid(mapObj);
        [pathSegment, solnInfo] = plan(planner, waypoints(i,:), waypoints(i+1,:));
        if isempty(pathSegment)
            error('Путь не найден между точками %d и %d.', i, i+1);
        end
        if i > 1
            % Проверяем, совпадает ли конец предыдущего сегмента
            if isequal(fullPath(end,:), pathSegment(1,:))
            pathSegment(1,:) = [];
            end
        end
        fullPath = [fullPath; pathSegment];
    end

    %Интерполяция траектории сплайном по кумулятивному расстоянию
    diffPath = diff(fullPath);
    segLengths = sqrt(sum(diffPath.^2, 2));
    cumDist = [0; cumsum(segLengths)];

    numInterp = 5000;  % вообще это надо подбирать и от этого многое зависит. Изначально было 500 и этого было мало
    s_interp = linspace(0, cumDist(end), numInterp);
    x_interp = spline(cumDist, fullPath(:,1)', s_interp);
    y_interp = spline(cumDist, fullPath(:,2)', s_interp);
```

```
    pathInterp = [x_interp', y_interp'];

    % Расчёт кривизны и локального ограничения скорости
    % Для каждой внутренней точки аппроксимируем радиус кривизны R
    R = Inf(numInterp,1); % На краях считаем R = Inf (прямолинейное движение)
    for i = 2:numInterp-1
        p_prev = pathInterp(i-1,:);
        p_curr = pathInterp(i,:);
        p_next = pathInterp(i+1,:);
        v1 = p_curr - p_prev;
        v2 = p_next - p_curr;
        norm_v1 = norm(v1);
        norm_v2 = norm(v2);
        if norm_v1 == 0 || norm_v2 == 0
            continue;
        end
        cosTheta = dot(v1, v2) / (norm_v1 * norm_v2);
        cosTheta = min(1, max(-1, cosTheta)); % Ограничиваем для acos
        theta = acos(cosTheta);
        d = norm(p_next - p_prev);
        if theta > 0
            R(i) = d / theta;
        end
    end

    v_limit = min(v_max, sqrt(a_max * R));
    % Для точек с Inf (прямолинейное движение) v_limit будет v_max

    % Расчёт профиля скорости с учётом ускорения, торможения и ограничений на поворот
    % Вычисляем профили скорости через прямой и обратный проход
    v_profile = zeros(numInterp,1);
    v_profile(1) = 0;
    % Прямой проход
    for i = 2:numInterp
        ds = s_interp(i) - s_interp(i-1);
        v_possible = sqrt(v_profile(i-1)^2 + 2 * a_max * ds
```

```
);
        v_profile(i) = min(v_possible, v_limit(i));
    end
    % Обратный проход торможение( до 0 в конце)
    v_profile(numInterp) = 0;
    for i = numInterp-1:-1:1
        ds = s_interp(i+1) - s_interp(i);
        v_possible = sqrt(v_profile(i+1)^2 + 2 * a_max * ds);
        v_profile(i) = min(min(v_profile(i), v_possible), v_limit(i));
    end

    % Расчёт временных меток по профилю скорости
    t_interp = zeros(numInterp,1);
    for i = 1:numInterp-1
        if (v_profile(i) + v_profile(i+1)) > 0
            dt = 2*(s_interp(i+1)-s_interp(i)) / (v_profile(i) + v_profile(i+1));
        else
            dt = 0;
        end
        t_interp(i+1) = t_interp(i) + dt;
    end

    % Вычисление угла theta вдоль траектории
    theta_interp = zeros(numInterp, 1);
    for i = 1:numInterp-1
        delta = pathInterp(i+1,:) - pathInterp(i,:);
        theta_interp(i) = atan2(delta(2), delta(1));
    end
    theta_interp(end) = theta_interp(end-1);

    Path = [pathInterp(:,2), pathInterp(:,1)];
    trajectory = [t_interp, Path, theta_interp];

    % Рисуем маршрут на исходном изображении
    fullPath_img = [ fullPath(:,2), imgHeight - fullPath(:,1) + 1 ];
    smoothPath_img = [Path(:,1), imgHeight - Path(:,2) +
```

```
1];

  startPt_img = [ startPt(2), imgHeight - startPt(1) + 1 ];
  goalPt_img  = [ goalPt(2), imgHeight - goalPt(1) + 1 ];
  if ~isempty(interPts)
      interPts_img = [ interPts(:,2), imgHeight - interPts(:,1) + 1 ];
  end

  % Увеличиваем картинку для рисования
  % Изначально тестировал на маленьких картинках и так было удобнее смотреть
  scaleFactor = 3;
  I_scaled = imresize(I, scaleFactor, 'nearest');

  fullPath_scaled = fullPath_img * scaleFactor;
  smoothPath_scaled = smoothPath_img * scaleFactor;
  startPt_scaled = startPt_img * scaleFactor;
  goalPt_scaled = goalPt_img * scaleFactor;
  if ~isempty(interPts)
      interPts_scaled = interPts_img * scaleFactor;
  end

  figure;
  imshow(I_scaled);
  hold on;

  %plot(fullPath_scaled(:,1), fullPath_scaled(:,2), 'Color', [0.5 0.5 0.5], 'LineWidth', 1);

  plot(smoothPath_scaled(:,1), smoothPath_scaled(:,2), 'Color', [0 0.9 0], 'LineWidth', 2);
  plot(startPt_scaled(1), startPt_scaled(2), 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'g');
  plot(goalPt_scaled(1), goalPt_scaled(2), 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');
  if ~isempty(interPts)
      plot(interPts_scaled(:,1), interPts_scaled(:,2), 'bo', 'MarkerSize', 10, 'MarkerFaceColor', 'b');
```

```
    end
    title(sprintf('Маршрут. Общее время движения: %.2f сек',
  t_interp(end)));
    hold off;

    fprintf('Общее время движения: %.2f сек\n', t_interp(end)
  );
end
```

Листинг 2: Задание численных параметров

```
%Работа с траекторией:
traj = planRoute('img1.png', 5, 2, 2); % получение траектории

x_0 = traj(1, 2)  % Первое значение x
y_0 = traj(1, 3)  % Первое значение y
theta_0 = traj(1, 4) % Первое значение theta

x_last = traj(end, 2)  % Последнее значение x
y_last = traj(end, 3)  % Последнее значение y
theta_last = traj(end, 4) % Последнее значение theta
t_end = traj(end, 1);

%Задаём параметры модели:
wheel_r = 0.05  % радиус колёс

wheel_d = 0.02 % ширина колёс

wheel_m = 0.1  % масса колеса

platform_m = 0.5  % масса тележки

platform_lx = 0.10  % ширина  тележки

platform_ly = 0.20  % длина тележки

y_i = pi/4  % угол ролика

n = 4 % количество колёс

```

```
%Расчитываем необходимое для симуляции:
l_w = sqrt(platform_lx^2 + platform_ly^2)    % расстояние от цм.. колеса до цм.. платформы

beta = asin(platform_ly/l_w)   % угол, который понадобится позже в расчётах

I_w = wheel_m*(1/4*wheel_r^2 + 1/12*wheel_d^2 + l_w^2)   % момент инерции колеса относительно цм.. платформы

I_p = 1/12 * platform_m * l_w^2   % момент инерции платформы

I = I_p + n*I_w   % общий момент инерции

cos_y_i = cos(y_i) % нужно для симуляции, можно было оставить в симулинке, но так удобнее

sin_y_i = sin(y_i)

m = n*wheel_m + platform_m   % общая масса

T_0 = [1  0 -platform_ly;        % переход между глобальной системой координат и координатами робота
       0  1  platform_lx;
       0  0  1]

pinv_T_0 = pinv(T_0)   % переход между координатами робота и глобальной системой координат

UtoF = [cos_y_i   cos_y_i  cos_y_i   cos_y_i;
        -sin_y_i  sin_y_i  sin_y_i  -sin_y_i;
        -l_w      l_w      -l_w      l_w] % переход между u и F колёс

pinv_UtoF = pinv(UtoF) % переход между F колёс и u
```

Листинг 3: Функция внутри MATLAB Function на схеме моделирования

```
function [x_i, y_i, theta_i] = fcn(x, y, theta, time,
    y_last, t_end, x_last, theta_last)

if time>t_end
    x_i = x_last;
    y_i = y_last;
    theta_i = theta_last;
else
    x_i = x;
    y_i = y;
    theta_i = theta;
end
```