Федеральное государственное учреждение
Федеральный исследовательский центр
«Информатика и управление»
Российской академии наук

На правах рукописи

Дарьина Анна Николаевна

Методы решения задач оптимального управления с фазовыми ограничениями, возникающими в робототехнике

Специальность 1.2.3 —
«Теоретическая информатика, кибернетика»

Диссертация
на соискание ученой степени
доктора физико-математических наук

Научный консультант:
д.ф.-м.н. Д.Ю. Карамзин

Москва, 2025

Оглавление

Введение

Актуальность темы исследования. В настоящее время большое внимание уделяется решению задач оптимизации управления мобильными роботами. Робот может как иметь связь с оператором, получая от него команды (ручное управление), так и действовать автономно, в соответствии с заложенной программой (автоматическое управление). Автоматизация колесных мобильных роботов является предметом многих исследований. Но, несмотря на фундаментальные и прикладные исследования, задачи по управлению ими до конца не решены. Задачи изучения траекторий мобильного робота и расчета управления актуальны в связи с возрастающими требованиями к точности таких систем, необходимостью синтеза оптимального управления, обеспечения плавных динамических перемещений. Расчет оптимальной траектории в сложных средах с фазовыми ограничениями требует оценки всего пространства возможных состояний и поиска наилучшего решения.

Развитие современных робототехнических систем связано с их всё более широким применением в промышленности, здравоохранении, логистике, сельском хозяйстве, оборонной сфере и бытовой технике. В связи с этим возрастают требования к функциональным возможностям таких систем: автономности, точности выполнения задач, устойчивости к внешним воздействиям, энергоэффективности, а также безопасности взаимодействия как с человеком, так и с окружающей средой. Одной из ключевых проблем при проектировании и эксплуатации робототехнических систем является обеспечение эффективного управления при наличии ограничений на физические параметры объектов – такие ограничения могут касаться силовых характеристик исполнительных механизмов, скоростей движения, диапазонов изменения обобщённых координат, потребляемой мощности, тепловых режимов и т. д.

Современная робототехника стремительно развивается, охватывая широкий спектр приложений – от промышленной автоматизации и транспортных систем до медицинских и бытовых устройств. Управление мобильными и манипуляционными роботами становится всё более сложным, поскольку предъявляются

высокие требования к точности, скорости реакции, энергоэффективности и безопасности. В этой связи особую важность приобретают методы оптимального управления, позволяющие находить наилучшие в том или ином смысле траектории движения и управляющие воздействия при наличии разнообразных ограничений. Это требует как глубокого теоретического анализа, так и разработки устойчивых вычислительных алгоритмов, применимых к широкому классу моделей.

Автоматизация процесса управления представляет собой переход от ручного или частично ручного регулирования к использованию технических средств, алгоритмов и программных комплексов, способных принимать решения и управлять системой без непосредственного участия человека. Этот процесс лежит в основе современных интеллектуальных систем и является ключевым элементом цифровой трансформации промышленности, транспорта, энергетики, здравоохранения и других сфер.

С развитием вычислительной техники, сенсорики и алгоритмов искусственного интеллекта автоматизация управления вышла на качественно новый уровень. Современные системы способны не только отслеживать текущее состояние объекта, но и прогнозировать его поведение, адаптироваться к изменениям внешней среды и находить оптимальные управляющие воздействия в реальном времени. Это особенно важно в условиях сложных, быстро меняющихся или опасных для человека сред – например, в космических миссиях, на атомных станциях или в автономных транспортных средствах.

Одной из важнейших задач при автоматизации является разработка алгоритмов, обеспечивающих выполнение поставленных целей при соблюдении фазовых и управляющих ограничений. Здесь на первый план выходят методы оптимального и адаптивного управления, а также модели предиктивного управления, которые позволяют строить траектории с учётом будущего поведения системы и ограничений на её параметры.

Автоматизация также способствует повышению точности, скорости и надежности управления. Системы, оснащённые датчиками, микроконтроллерами и программным обеспечением, могут работать непрерывно, без усталости, минимизируя человеческий фактор и снижая вероятность ошибок. Кроме того, автоматизированные системы позволяют эффективно интегрировать большие объёмы данных, используя методы машинного обучения и анализа для улучшения управленческих решений.

Важно отметить, что автоматизация не исключает, а трансформирует роль человека: вместо непосредственного управления оператор переходит к функциям мониторинга, настройки параметров, анализа и принятия стратегических ре-

шений. Это требует новых подходов к обучению специалистов, проектированию человеко-машинных интерфейсов и обеспечению кибербезопасности.

Таким образом, автоматизация процесса управления – это не просто замена человека техникой, а создание интеллектуальных, самонастраивающихся систем, способных эффективно и безопасно функционировать в сложных и неопределённых условиях. Она является движущей силой научно-технического прогресса и тесно связана с развитием математики, информатики и инженерных наук, обеспечивая реализацию приоритетных направлений цифровой экономики и технологического суверенитета. Особую значимость указанные проблемы имеют в контексте развития интеллектуальных робототехнических систем, в которых требуется интеграция методов управления с ограничениями с технологиями машинного обучения, планирования траекторий, восприятия окружающей среды и принятия решений. Такая интеграция позволяет создавать более гибкие, адаптивные и надежные системы, способные работать в неструктурированных и изменяющихся условиях.

С развитием автономных систем, таких как мобильные роботы, манипуляторы, беспилотные летательные аппараты и транспортные средства, возрастает потребность в эффективных методах управления, которые обеспечивают не только достижение цели, но и соблюдение различных ограничений на состояния и управляющие воздействия. Задачи управления с ограничениями представляют собой важное направление теории управления, имеющее прямое прикладное значение для робототехники. Они заключаются в синтезе алгоритмов управления, которые обеспечивают достижение заданных целей функционирования системы с одновременным соблюдением жёстких и мягких ограничений на состояния, входные сигналы и выходные переменные. Эти задачи особенно актуальны при разработке мобильных и антропоморфных роботов, систем совместной работы, автономных транспортных средств и дронов, где управление должно осуществляться в условиях неопределённости, ограниченных вычислительных ресурсов и необходимости обеспечения безопасного поведения в реальном времени.

Одним из ключевых классов задач оптимального управления, возникающих в робототехнике, являются задачи с фазовыми ограничениями, которые задают допустимые области изменения состояния системы в каждый момент времени (например, положения, скорости, ориентация). Такие ограничения могут быть связаны с необходимостью избегать столкновений, сохранять устойчивость движения, соблюдать технологические режимы работы исполнительных механизмов, ограничивать диапазоны координат и скоростей, а также обеспечивать безопасность взаимодействия робота с человеком и окружающей средой, а также удовлетворять технологическим требованиям к функционированию отдельных

компонентов робота. Фазовые ограничения отражают физические, технические или экологические границы, в которых может работать система. Например, температура в реакторе не должна превышать определённого уровня, траектория аппарата – выходить за безопасную зону, а уровень запасов в логистической системе – опускаться ниже критического минимума. Нарушение этих условий может привести к авариям, снижению эффективности или полному отказу системы. Наличие фазовых ограничений существенно усложняет как теоретический анализ задач, так и построение эффективных численных методов их решения.

В отличие от задач без ограничений, учет фазовых условий значительно усложняет анализ и поиск оптимального решения. Традиционные подходы к решению задач оптимального управления, такие как принцип максимума Понтрягина и динамическое программирование, сталкиваются с рядом трудностей при учете фазовых ограничений, особенно в многомерных и нелинейных системах и требуют модификации для корректного учета таких ограничений. Это обусловлено тем, что фазовые ограничения изменяют структуру экстремалей, вызывают необходимость анализа участков активных ограничений, а также требуют корректного определения множителей Лагранжа и сопряжённых переменных на границе допустимой области. Следовательно, актуальной задачей является разработка и обоснование новых методов и алгоритмов, способных эффективно учитывать фазовые ограничения в реальных робототехнических системах.

Особую сложность представляют задачи оптимального управления, в которых траектория системы многократно касается границы фазового ограничения, что приводит к возникновению скользящих режимов, разрывов в сопряжённых переменных и потере гладкости оптимального управления. Существующие аналитические методы, такие как принцип максимума Понтрягина, требуют сложной интерпретации в таких случаях, а численные подходы сталкиваются с проблемами устойчивости, сходимости и адекватного отражения структуры множителей Лагранжа. В свою очередь, численные методы часто испытывают проблемы со сходимостью, особенно при высокой размерности задачи и наличии множественных активных ограничений.

Разработка эффективных алгоритмов численного анализа и синтеза оптимальных траекторий в условиях фазовых ограничений, включая корректную обработку входа на границу и выхода с неё, представляет собой актуальную научную проблему, лежащую на стыке теоретической кибернетики, вычислительной математики и информационных технологий.

В последние годы наблюдается значительный прогресс в области методов управления с ограничениями, включая модели предиктивного управления, оптимальное управление, методы штрафных функций, управление на основе ба-

рьерных функций и др. Однако остаются открытыми вопросы практической реализации этих подходов в сложных нелинейных системах, особенно в условиях частичной наблюдаемости состояния, временных задержек, дискретизации сигналов и аппаратных ограничений вычислительных платформ.

Актуальность решения задач оптимального управления с фазовыми ограничениями особенно возрастает в контексте современных приоритетных направлений научно-технического развития, определенных как на национальном, так и на международном уровне. К таким направлениям относятся цифровизация промышленности, развитие автономных и интеллектуальных систем, робототехника, космические технологии, «зелёная» энергетика и персонализированная медицина – все они напрямую связаны с необходимостью точного и безопасного управления сложными динамическими процессами в условиях жестких ограничений.

Например, в робототехнике и автономном транспорте фазовые ограничения обеспечивают соблюдение границ рабочей зоны, ограничение скорости, ускорения или температурных режимов, что критично для безопасности и надёжности. В космической отрасли при маневрировании аппаратов требуется строгое соблюдение ограничений по траектории, расходу топлива и ориентации, чтобы избежать столкновений и перегрузок. В энергетических системах, особенно с использованием возобновляемых источников, важно удерживать параметры в допустимых диапазонах для стабильной работы сети. В биомедицинских приложениях, таких как управление дозировкой лекарств или функционирование имплантируемых устройств, фазовые ограничения защищают организм пациента от чрезмерного воздействия.

Государственные программы, такие как национальные проекты в области науки, технологий и цифровой экономики, а также международные инициативы в сфере искусственного интеллекта и автономных систем, выделяют разработку методов оптимального управления с учетом ограничений в качестве стратегического направления. Это связано с растущей потребностью в системах, способных принимать оптимальные решения в реальном времени при неполной информации и высокой степени неопределённости.

Актуальность задач оптимального управления с фазовыми ограничениями тесно связана с развитием современной прикладной математики, которая выступает фундаментом для построения строгих моделей и эффективных алгоритмов в этой области. Математика обеспечивает теоретическую базу для анализа динамических систем, формулировки условий оптимальности и разработки методов решения сложных задач, в которых наряду с критерием качества необходимо учитывать ограничения на состояние системы.

Фазовые ограничения приводят к необходимости расширения классических подходов, таких как принцип максимума Понтрягина или метод динамического программирования. Их учет требует привлечения аппарата нелинейного анализа, теории дифференциальных включений, методов вариационного исчисления и выпуклой оптимизации. Особую роль играет теория множителей Лагранжа в бесконечномерных пространствах, а также современные подходы к построению двойственных оценок и условий трансверсальности.

Кроме того, развитие численных методов решения краевых задач, проектирование эффективных схем дискретизации и использование методов оптимизации в функциональных пространствах – все это является предметом активных математических исследований. Современные вычислительные алгоритмы, такие как методы последовательного квадратичного программирования, стрельбы, коллокации или псевдоспектральные методы, основаны на глубоких математических результатах и непрерывно совершенствуются благодаря междисциплинарному взаимодействию математики и инженерных наук.

Таким образом, задачи оптимального управления с фазовыми ограничениями стимулируют развитие новых разделов прикладной математики, способствуя появлению более точных моделей, устойчивых алгоритмов и обобщённых теорем существования и регулярности решений. Это делает их не только практически значимыми, но и теоретически глубокими, что подчёркивает центральную роль математики в преодолении вызовов современной науки и технологий. В условиях роста сложности систем и повышения требований к их безопасности и эффективности, математика становится ключевым инструментом для построения надёжных и оптимальных решений в реальном мире.

Оптимальное управление в режиме реального времени – это важнейшее направление в современной теории и практике управления, направленное на реализацию оптимальных решений непосредственно в ходе функционирования динамической системы, с учётом текущих данных и изменяющихся условий. Такой подход особенно актуален для систем, где задержки в принятии решений могут привести к снижению эффективности, нарушению безопасности или полному отказу.

В отличие от классических задач оптимального управления, решаемых заранее и в offline-режиме, управление в реальном времени требует, чтобы алгоритмы находили и применяли управляющие воздействия за строго ограниченный промежуток времени между измерением состояния системы и следующим шагом управления. Это ставит высокие требования к скорости вычислений, устойчивости алгоритмов и точности моделей.

К ключевым особенностям оптимального управления в реальном времени

относятся: быстродействие алгоритмов – необходимость решения сложных оптимизационных задач за доли или миллисекунды; учет ограничений – возможность явно включать в расчет фазовые (например, температура, скорость, положение) и управляющие (максимальная сила, мощность) ограничения; адаптивность – способность корректировать управление при изменении условий или появлении внешних возмущений; интеграция с сенсорами и системами связи – тесное взаимодействие с измерительными устройствами и каналами передачи данных.

Такие системы находят широкое применение в автономном транспорте (управление траекторией, торможением, обходом препятствий); в промышленной автоматизации (оптимизация технологических процессов в режиме онлайн); в энергетике (балансировка нагрузки в умных сетях); в робототехнике (точное позиционирование и взаимодействие с динамической средой); в медицинских устройствах (например, в инсулиновых помпах с адаптивным дозированием).

Развитие вычислительных технологий, включая использование встраиваемых процессоров, специализированных ускорителей и эффективных численных методов, делает возможным реализацию сложных алгоритмов оптимального управления даже на бортовых системах с ограниченными ресурсами.

Таким образом, исследование методов решения задач оптимального управления с фазовыми ограничениями, возникающих в робототехнике, является актуальной научной проблемой, имеющей важное значение для проектирования и реализации современных автономных систем, а оптимальное управление в режиме реального времени представляет собой синергию математической теории, вычислительных алгоритмов и инженерной практики, играет ключевую роль в создании интеллектуальных, безопасных и высокопроизводительных систем, отвечающих вызовам современного мира, и остаётся одной из приоритетных областей в развитии автоматики, робототехники и цифровых технологий.

Диссертационная работа направлена на решение актуальной **научной проблемы** – получение точного решения задачи оптимального управления с фазовыми ограничениями, применительно к мобильному роботу, для последующего использования этого решения при управлении роботом в режиме реального времени.

Цель диссертационного исследования – разработка методов решения задачи оптимального управления с ограничениями применительно к робототехническим устройствам с возможностью использования в режиме реального времени.

Для достижения поставленной цели решаются следующие **задачи**:

1. Аналитический обзор методов решения задач оптимального управления

мобильными роботами и существующих подходов к их практической реализации.

2. Теоретическое обоснование метода регуляризации для задачи оптимального управления с фазовыми ограничениями для мобильных роботов.

3. Разработка алгоритма и численная реализация метода регуляризации через построение аппроксимирующей последовательности регулярных ε-задач для решения задачи оптимального движения мобильного робота с передним приводом при наличии фазовых ограничений.

4. Теоретический подход к решениию задачи о перемещении мобильного робота при фазовых ограничениях глубины 2.

5. Разработка алгоритма и численное решение задачи о перемещении мобильного робота в рамках непрямого подхода на основе модифицированного принципа максимума Понтрягина при фазовых ограничениях глубины 2.

6. Математическая формализация принципа разделения траекторий для задач оптимального управления с фазовыми ограничениями при ненулевой начальной скорости, теоретическое обоснование принципа.

7. Разработка алгоритма, обоснование его сходимости и численная реализация принципа разделения траекторий с фазовыми ограничениями при ненулевой начальной скорости.

8. Построение математической модели для управления мобильным роботом в реальных условиях.

9. Разработка алгоритмов определения параметров математической модели, генерации траекторий движения мобольного робота, определения функции выбора на основе сгенерированной траектории с помощью адаптивного функционала.

10. Разработка метода решения задачи оптимального управления с фазовыми ограничениями для реального мобильного робота с учетом навигационной информации.

11. Разработка программных комплексов для нахождения оптимальной траектории и управления робототехническими устройствами и проведение математического и имитационного моделирования движения робототехнических объектов.

Объектом исследования является оптимальное движение мобильного робота в плоскости с препятствиями.

Предметом исследования служат численные методы для разработки методов решения задач оптимального управления с фазовыми ограничениями, применительно к робототехническим устройствам.

Методы исследования. В диссертационной работе использовались методы теории оптимального управления, математического анализа, дифференциальных уравнений, линейной алгебры, методы численного моделирования, методы машинного обучения и численные методы оптимизации.

Научная новизна.

Научная новизна настоящего исследования заключается в разработке и теоретическом обосновании новых методов управления робототехническими системами при наличии фазовых ограничений, а также в создании эффективных алгоритмических и программных средств их реализации.

1. Проведен теоретический анализ метода регуляризации для задачи оптимального управления с фазовыми ограничениями для мобильных роботов. На основе анализа получены аналитические выражения для меры множителя Лагранжа и разработан алгоритм решения задачи управления мобильным роботом. Данный подход позволил получать точные решения задачи оптимального управления мобильным роботом при наличии фазовых ограничений. Точные решения использовались при построении управления в режиме реального времени как «ядро» управления, а методы машинного обучения – как слой адаптации и устойчивости к неопределённостям.

2. Разработан, теоретически обоснован и численно проверен метод стрельбы для нахождения решения задачи о перемещении мобильного робота при фазовых ограничениях глубины 2. Метод применялся к задаче с динамикой управления ньютоновского типа. Предложенный метод позволил находить точные решения, которые впоследствии использовались для улучшения структуры пропорционально-интегрально-дифференцирующего регулятора, применяемого при нахождении траекторий мобильного робота в режиме реального времени.

3. Разработан метод поиска касательных, которые используются в принципе разделения траекторий на прямолинейные участки и движение по границе фазового ограничения для поиска моментов переключения между разными видами траекторий. Идея заключалась в построении графа вершинами, которого были точки переключения, а ребрами – траектории. Метод позволил находить оптимальную траекторию при наличии большого количества

препятствий. Проведено теоретическое исследование и обоснование метода.

4. Разработаны подходы к решению задачи оптимального управления с фазовыми ограничениями для нелинейных робототехнических систем, включающих новые методы формирования допустимых траекторий движения, учитывающих ограничения на обобщенные координаты, скорости, ускорения и управляющие воздействия. В отличие от существующих решений, предложенные методы обеспечивают высокую точность управления и устойчивость при снижении вычислительной сложности за счёт использования адаптивных стратегий прогнозирования и коррекции траектории.

5. Разработаны новые методы интеграции управления с технологиями машинного обучения и восприятия окружающей среды, что позволило реализовать адаптивное поведение робота в условиях частичной информации о состоянии объекта и внешней среды: новый метод идентификации модели мобильного робота, метод генерации траекторий с помощью нейронных сетей и метод определения функции выбора интегрального управления на основе сгенерированной траектории с помощью модели с использованием адаптивного функционала. Это дало возможность повысить автономность и устойчивость робототехнических систем при выполнении задач в неструктурированных условиях.

6. Разработан гибридный алгоритм, который с помощью модифицированого метода модели предиктивного управления, предлагает новую структуру функции выбора для учета оптимального управления, а с помощью метода одновременной локализации и картографии использует навигационную информацию. Этот метод ориентирован на реальное время и ограниченные вычислительные ресурсы бортовых контроллеров роботов, но позволяет учитывать фазовые ограничения, обеспечивая при этом гладкость переходных процессов и минимизацию энергетических затрат.

Достоверность научных результатов диссертационной работы подтверждается совпадением теоретических выводов и экспериментальных данных, полученных на математических моделях и опытных образцах в Робототехническом центре ФИЦ ИУ РАН, а также подтверждается экспертизой научных статей, опубликованных в ведущих научных российских и международных изданиях, апробацией и обсуждением результатов на международных и российских научных конференциях и семинарах.

Теоретическая и практическая значимость работы. Представленные в диссертационной работе результаты имеют как теоретическое, так и практическое значение. Полученные в работе новые методы, изложенные в главах 1, 2, 3, теоретически обоснованы, их можно применять к различным моделям мобильных роботов. Кроме того, метод регуляризации, представленный в главе 1, может быть модифицирован многими новыми способами. Полученное методом регуляризации точное решение используется как «эталон» для методов машинного обучения. Рассмотренный в главе 2 метод стрельбы также дает точное решение задачи ньютоновского типа, которое служит «эталоном» для ПИД-регулятора. Эти исследования применяются в образовательном процессе. Методы, разработанные в главе 4, могут быть использованы на реальных роботах для объезда роботом указанной местности по заранее спланированной территории с целью, например, составления карты местности. Все алгоритмы реализованы в виде программных модулей и используются в исследованиях, проводимых в Робототехническом центре ФИЦ ИУ РАН по государственному заданию № FFNG-2024-0010.

Соответствие диссертации паспорту научной специальности. В соответствии с формулой специальности 1.2.3 «Теоретическая информатика, кибернетика» (физико-математические науки) диссертация посвящена разработке и исследованию методов решения задач оптимального управления с фазовыми ограничениями, возникающими в робототехнике, что соответствует следующим пунктам паспорта специальности: п. 6 «Математическая теория оптимального управления, включая оптимального управления в условиях конфликта» и п. 9 «Математическая теория исследования операций».

Основные положения, выносимые на защиту

1. Теоретическое обоснование метода регуляризации для задачи оптимального управления с фазовыми ограничениями для мобильных роботов, метода стрельбы для нахождения решения задачи о перемещении мобильного робота при фазовых ограничениях глубины 2 и метода поиска касательных в задаче быстродействия для колесного мобильного робота.

2. Разработка метода регуляризации, основанного на принципе максимума Понтрягина и регулярных приближениях исходной задачи оптимального управления с фазовыми ограничениями для мобильных роботов.

3. Разработка метода стрельбы для нахождения решения задачи о перемещении мобильного робота с динамикой управления ньютоновского типа при фазовых ограничениях глубины 2.

4. Разработка метода поиска касательных для нахождения моментов переключения между прямолинейными участками траекторий и участками, идущими по границе фазовых ограничений в задаче быстродействия для колесного мобильного робота.

5. Разработка метода поиска управления мобильным роботом в режиме реального времени, включающего идентификацию модели мобильного робота, генерацию траекторий и определение функции выбора управления на основе сгенерированной траектории с помощью модели робота.

6. Численные методы реализации гибридного алгоритма, который предлагает новую структуру функции выбора для учета оптимального управления и навигационной информации.

7. Комплексы программ, реализующие представленные в диссертационном исследовании методы решения задач оптимального управления с фазовыми ограничениями, возникающими в робототехнике.

Апробация результатов. Результаты диссертационного исследования докладывались и обсуждались на международных и всероссийских конференциях, семинарах и симпозиумах:

- Международный симпозиум «Надежность и качество». Пенза, 2016
- II Международная научно-практическая конференция, посвящённая 105-летию со дня рождения адмирала флота СССР дважды героя Советского Союза Сергея Георгиевича Горшкова. Елец, 2018
- Академия управления МВД России. 2018
- III Международная научно-практическая конференция, посвящённая 110-летию со дня рождения академика Н.А. Пилюгина. Елец, 2019
- European Control Conference 2020, ECC 2020
- 7th International Conference on Control, Decision and Information Technologies, CoDIT 2020
- 14-th International conference “Intellectual Systems” (INTELS’20), 2020
- IV Международная научно-практическая конференция «Фундаментально-прикладные проблемы безопасности, живучести, надёжности, устойчивости и эффективности систем». Елец, 2020

- 18th IFAC Workshop on Control Applications of Optimization, CAO 2022
- Международный симпозиум «Надежность и качество». Пенза, 2022
- 15-th International conference «Intellectual Systems» (INTELS'22), 2022
- IX Международной научно-практической конференции «Системы управления, сложные системы: моделирование, устойчивость, стабилизация, интеллектуальные технологии». Елец, 2023
- Всероссийская конференция с международным участием «Прикладные проблемы системной безопасности». Елец, 2023
- Всероссийская научно-практическая конференция «Прикладная физика и инжиниринг: актуальные проблемы». Елецкий государственный университет им. И.А. Бунина, 2024
- 16-th International conference «Intellectual Systems» (INTELS'24), 2024
- 11th International Conference on Control, Decision and Information Technologies, CoDIT 2025

Грантовая поддержка. Научные исследования в рамках диссертационной работы поддержаны следующими грантами: РФФИ 18-29-03061-мк «Исследование методов синтеза обучающихся систем интеллектуального управления робототехническими устройствами при фазовых и структурных ограничениях в технологии цифровой экономики», 2018-2020 (исполнитель, рук. Прокопьев И.В.), РНФ 19-11-00258 «Теоретическое и численное исследование задач оптимального управления с фазовыми ограничениями», 2019-2021 (исполнитель, рук. Карамзин Д.Ю.). Результаты, представленные диссертации в главе 3 в части обоснования и исследования принципа разделения траекторий, исследования разрабатываемого численного метода касательных и проведения вычислительных экспериментов на модели мобильного робота были получены в рамках проекта Министерства науки и высшего образования Российской Федерации 075-15-2024-544 «Математические модели и численные методы как основа для разработки робототехнических комплексов, интеллектуальных технологий конструирования».

Публикации по теме диссертации. Основные результаты диссертационного исследования опубликованы в 29 научных публикациях, из них 7 публикаций в изданиях, включенных в перечень рецензируемых журналов, рекомендованных ВАК, включая 6 публикаций в изданиях, отнесенных к категориям К-1 или К-2 из Перечня ВАК, 10 – в научных изданиях, индексируемых Web

of Science и Scopus, 12 – в трудах конференций и изданиях, включенных в базу РИНЦ.

Личный вклад. Все положения, выносимые на защиту, изложенные в диссертации, принадлежат лично автору. В совместных работах автор принимал непосредственное участие в формировании направления исследования, теоретических обоснованиях, в выводе формул для вычислений, в программной реализации, и преобладающее участии в выборе экспериментальных задач и проведении вычислительных экспериментов.

Структура и объем диссертации. Диссертация состоит из введения, четырех глав, заключения, библиографического списка и приложений. Содержание работы изложено на 354 страницах, включает 11 таблиц, 122 рисунка и 3 приложения.

Обзор методов решения задач оптимального управления мобильными роботами и подходов к их реализации

Область управления динамическими системами с фазовыми ограничениями представляет собой активно развивающееся и востребованное направление современной математической теории управления, обусловленное широким спектром приложений, особенно в робототехнике.

Исторически задачи поиска управляющих воздействий относились к классу задач вариационного исчисления. Многие задачи оптимального управления динамическими процессами успешно решаются с помощью классического вариационного исчисления, особенно в ситуациях, когда множество допустимых управлений открыто, а требуемые управляющие функции предполагаются гладкими [1–3]. Развитие и обобщение методов вариационного исчисления применительно к задачам оптимального управления освещены в трудах [4–8]. В становление и развитие этой теории значительный вклад внесли многие ученые, в числе которых выделяются работы Л. Эйлера, Ж.-Л. Лагранжа, К. Вейерштрасса, Д. Гильберта, А. Лебега, А.М. Лежандра, Б. Римана, К. Якоби, Л. Тонелли, Н.Н. Боголюбова, С.Н. Бернштейна, М.А. Лаврентьева, А.Г. Сигалова, В. Ритца, Дж. Варги, Э. Макшейна, Р.В. Гамкрелидзе, А.Д. Иоффе, В.Ф. Кротова, А.М. Размадзе, В.М. Тихомирова [9] и других. Однако на практике применение классических методов вариационного исчисления часто затруднено из-за наличия ограничений на управляющие параметры, а также возможных разрывов первого рода как у самих управляющих функций, так и у их производных. Учитывая

важность таких ограничений в реальных задачах, возникает необходимость в создании подходов, способных корректно учитывать данные особенности.

В 1960-х годах был сформулирован принцип максимума Понтрягина [10], который требует лишь слабых предположений о дифференцируемости и оказывается особенно эффективным для задач с ограничениями.

Появление принципа максимума стало поворотным моментом в развитии теории оптимального управления и стимулировало рост числа исследований и публикаций. Эти работы были направлены как на изучение связи между оптимальным управлением и классическим вариационным исчислением [11, 12], так и на расширение принципа максимума на более сложные классы задач, включая задачи с фазовыми и смешанными ограничениями. Первые необходимые условия оптимальности для задач с фазовыми ограничениями были получены Р.В. Гамкрелидзе [13] и представлены в фундаментальной монографии [10]. Позднее А.Я. Дубовицкий и А.А. Милютин предложили новую формулировку принципа максимума, применимую к задачам с ограничениями [14]. В отличие от подхода Гамкрелидзе, их метод не нуждается в предварительных предположениях о регулярности траектории. Тем не менее, в ряде практически значимых случаев принцип максимума в интерпретации Дубовицкого–Милютина может оказываться вырожденным. Явление вырождения было впервые обнаружено и детально исследовано А.В. Арутюновым и Н.Т. Тынянским [15], что послужило толчком к созданию модифицированных, невырожденных версий принципа.

С 1960-х годов данное направление теории активно развивается, о чём свидетельствует большое количество публикаций, посвящённых различным аспектам исследования [16–48].

Значительный вклад в развитие теории задач оптимального управления с фазовыми ограничениями внесли такие исследователи, как А.В. Арутюнов [49–52], С.М. Асеев [53], Р.В. Гамкрелидзе [13], А.В. Дмитрук, А.Я. Дубовицкий [14], В.А. Дубовицкий [54], М.И. Зеликин, Д.Ю. Карамзин [55–59], А.Б. Куржанский [60], А.С. Матвеев [61], А.А. Милютин [62], Н.П. Осмоловский, Е.С. Половинкин, Г.В. Смирнов, Н.Т. Тынянский, Х. Халкин, M.M. Ferreira [63, 64], H. Halkin, F.L. Pereira [65–71], R.B. Vinter [72, 73] и другие.

Принцип максимума справедливо считается ключевым инструментом для определения оптимальных управлений и соответствующих траекторий в различных динамических системах. Однако его использование в классической форме сопряжено со значительными аналитическими и вычислительными трудностями. В робототехнике крайне мало примеров решения новых прикладных задач с помощью принципа максимума. Большинство работ, связанных с этим подходом, остаются теоретическими, хотя изначально Л.С. Понтрягин подчёркивал

его практическую значимость.

Сложность задач оптимального управления в робототехнике обусловлена, в частности, нелинейностью динамики роботизированных систем, необходимостью учёта разнообразных фазовых ограничений, а также сложными динамическими взаимодействиями при управлении группами роботов. Эти факторы приводят к тому, что классические математические методы оптимального управления всё реже применяются при проектировании современных робототехнических решений.

Ещё одна серьёзная проблема заключается в том, что в стандартной постановке задачи оптимальное управление ищется как функция времени, что соответствует разомкнутой системе управления — управляющие воздействия заранее задаются по временному графику, а не по текущему состоянию объекта. Даже в случаях, когда удаётся осуществить переход к управлению по состоянию (синтезу замкнутой системы), анализ качественных характеристик движения остаётся затруднительным. В реальных системах управления, как правило, используются алгоритмы с обратной связью, что делает классический принцип максимума малопригодным для практической робототехники без существенной его адаптации.

В задачах с фазовыми ограничениями часто возникает явление вырождения принципа максимума, связанное с поведением системы на границе допустимой области состояний. Для преодоления этой трудности были разработаны модифицированные формы принципа максимума, учитывающие специфику изменения гамильтониана вблизи границы ограничений [15, 52, 54, 63, 64, 74–76, 78]. Отдельно стоит отметить исследования, посвящённые непрерывности множителя в виде меры, которые играют важную роль в дальнейшем анализе [27, 79–83]. Также существует обширная литература по численным методам решения задач оптимального управления с фазовыми ограничениями и смежным вопросам, см., например, [65–67, 84–93].

Методы решения задач с фазовыми ограничениями высокого порядка представлены в работах [36, 39, 94, 95].

Значительный вклад в развитие теории оптимального управления и её практического применения внесли такие выдающиеся учёные, как Н.Н. Красовский, Ю.С. Осипов, А.В. Кряжимский, А.Б. Куржанский, Ф.П. Васильев, В.И. Благодатских, С.М. Асеев, М.И. Зеликин, В.М. Тихомиров, Ф.Л. Черноусько, Р. Габасов, Ф.М. Кириллова, М. Атанс, П. Фалб, Р. Беллман, Ф. Кларк, А. Брайсон, Хо Ю-ши, Э. Ли, Л. Маркус и многие другие.

Что касается применения методов оптимального управления в робототехнике, то примеры таких исследований можно найти в работах [96–99].

Большинство практических задач оптимального управления характеризуются неунимодальной структурой целевого функционала, особенно в условиях наличия фазовых ограничений. Такие ограничения являются типичными для задач управления роботами и особенно актуальны при координации групп роботов, где каждый агент выступает одновременно как объект управления и как динамическое ограничение для других участников системы. В таких условиях множество допустимых траекторий становится невыпуклым, а поиск оптимального решения — существенно более сложной задачей.

Универсальных алгоритмов для решения подобных задач нелинейного программирования не существует. Выбор подходящего метода зависит от ряда факторов: формы целевой функции, сложности её вычисления, типа и количества ограничений, требуемой точности и возможностей вычислительной платформы. Среди прямых численных методов решения экстремальных задач особое место занимают градиентные подходы [100, 101]. Однако в условиях неунимодальности и невыпуклости функционала эти методы обладают серьёзными ограничениями — в частности, они склонны к попаданию в локальные экстремумы и не обеспечивают сходимость к глобальному оптимуму [102].

На практике даже приближённая оценка глобального минимума или максимума зачастую недоступна, что делает особенно актуальным использование современных методов глобальной оптимизации. Особый интерес представляют как детерминированные алгоритмы, разработанные в трудах Ю.Г. Евтушенко, Р.Г. Стронгина, М.А. Посыпкина, И.Х. Сигала, Х. Туя и других исследователей [103–107], так и, в особенности, эвристические методы, основанные на популяционных и эволюционных принципах поиска. К последним относятся подходы, предложенные Дж. Холландом, Л. Фогелем, Д. Гольдбергом, а также развитые в работах Е.С. Семенкина, А.П. Карпенко и других авторов [108–112, 114, 162]. Эти методы находят широкое применение при параметрическом поиске управляющих функций в задачах с фазовыми ограничениями [115, 116].

В настоящее время разработано множество как прямых, так и непрямых методов решения задач оптимального управления. Прямой подход состоит в аппроксимации исходной задачи путём дискретизации управляющих и фазовых переменных на временной сетке, что позволяет свести её к задаче нелинейного программирования [117]. Напротив, так называемый непрямой подход традиционно считается классическим методом и основан на использовании принципа максимума Понтрягина. На сегодняшний день этот принцип остаётся одним из наиболее важных и фундаментальных результатов в теории оптимального управления.

Методы и алгоритмы, применяемые для решения задач быстродействия, мож-

но условно разделить на несколько групп: аналитические (включая принцип максимума Понтрягина [10, 16]), численные (такие как различные методы оптимизации [31, 118–120], метод Дейкстры [121], A* [122], D* [123]), эволюционные и генетические алгоритмы [116]), метод сетевого оператора [124–126]), алгоритмы оптимизации по принципу роя частиц (PSO) [111, 112, 114, 127, 162]), вероятностные методы (например, марковские процессы принятия решений [128]), методы машинного обучения (в частности, обучение с подкреплением [129]) и методы, основанные на дискретизации конфигурационного пространства, такие как RRT (Rapidly-exploring Random Tree) [130, 131] и PRM (Probabilistic Roadmap Method) [132] и др.

Методы оптимизации формулируют задачу построения траектории как задачу математической оптимизации и используют численные процедуры для её решения. Их основное преимущество заключается в способности находить траектории, оптимальные по заданному критерию, с учётом сложных динамических и фазовых ограничений, в том числе при невыпуклости множества допустимых решений. Однако, несмотря на активное развитие численных методов оптимизации, не все из них применимы для решения задач оптимального управления в реальных условиях [116]. Многие из этих методов требуют высококачественного начального приближения и могут быть чрезмерно ресурсоёмкими с точки зрения вычислительных затрат.

Среди подходов, эффективно работающих на дискретизированных фазовых пространствах, одним из наиболее распространённых алгоритмов поиска пути является метод A* (A-star). Он использует эвристическую оценку стоимости оставшегося пути, что позволяет находить оптимальное решение с учётом препятствий, сочетая преимущества алгоритма Дейкстры и жадного поиска. Метод D* (Dynamic A*) представляет собой развитие A*, ориентированное на динамические среды — он способен эффективно перестраивать траекторию при изменении условий, например, при появлении новых препятствий. Однако, как и A*, он требует предварительной дискретизации пространства. Алгоритм Дейкстры находит кратчайший путь от начальной вершины до всех остальных в графе, что делает его полезным для задач поиска оптимального маршрута. Тем не менее, по сравнению с A*, он менее эффективен, поскольку не использует эвристическую информацию и исследует значительно большее число узлов.

Генетические алгоритмы, которые также требуют дискретизации временного интервала, сталкиваются с проблемой медленной сходимости при увеличении числа временных узлов. Это связано с тем, что целевая функция становится сложной и многоэкстремальной, а начальные значения управляющих параметров оказывают значительно большее влияние на качество решения, чем пара-

метры в конечной части интервала. Метод символьной регрессии, позволяющий приближённо строить управляющую функцию в аналитической форме, демонстрирует меньшую чувствительность к числу временных узлов, что делает его одним из наиболее перспективных подходов. Однако, как и другие методы численной оптимизации, включая генетические алгоритмы, он характеризуется высокой вычислительной сложностью.

Вопросы управления реальными роботами рассматривались в ряде исследований [133–149]. Работа [150] посвящена применению обучения с подкреплением, а в [151, 152] рассматриваются методы глубокого обучения. Задачи планирования траекторий изучались в публикациях [153–159], тогда как вопросы навигации освещены в работах [160–173].

Основные обозначения

В данном разделе преведены основные обозначения, используемые в диссертационной работе.

$\mathbb{R}$	множество вещественных чисел
$\mathbb{R}^n$	n-мерное арифметическое пространство
$\mathbb{R}^{m\times n}$	пространство вещественных матриц размера $m \times n$
$e^1, e^2, \ldots, e^n$	векторы стандартного базиса в $\mathbb{R}^n$
E	единичная матрица, размерность которой всегда ясна из контекста
$\langle x, y\rangle$	скалярное произведение векторов x и y
$\|\|x\|\|$	евклидова норма вектора x, равная $\langle x, y\rangle^{1/2}$
$\|\|x\|\|_{L_2}$	норма вектора x в пространстве L_2
A^T	транспонированная матрица A
$\|I\|$	мощность конечного множества I
x_I	вектор с компонентами x_i, $i \in I$, где $x \in \mathbb{R}^n$ и $I \subset \{1, \ldots, n\}$
$F : \mathbb{R}^n \to \mathbb{R}^m$	отображение, действующее из пространства $\mathbb{R}^n$ в пространство $\mathbb{R}^m$
$\frac{\partial F}{\partial x_I}$	матрица частных производных F по переменным x_i, $i \in I$
$\Psi'(x; d)$	производная отображения $\Psi : \mathbb{R}^n \to \mathbb{R}^m$ в точке $x \in \mathbb{R}^n$ по направлению $d \in \mathbb{R}^n$
$\operatorname{im}\Lambda$	образ линейного оператора Λ
$\ker\Lambda$	ядро линейного оператора Λ
$X \times Y$	декартово произведение множеств X и Y
$\varphi^i(u)$	i-ая компонента функции φ

$\nabla\varphi^i(u)$	градиент i-ой компоненты функции φ
∇_u	частичный градиент относительно u
Γ'_u	производная функции Γ
lin	линейная оболочка множества
$x_\varepsilon \rightrightarrows x_0$	x_ε слабо сходится к x_0
$\operatorname{Clos} u$	замыкание по мере функции u

Сокращения и абберевиатуры

В данном разделе преведены сокращения и абберевиатуры, используемые в диссертационной работе.

РТС	робототехнические системы
MPC	модель прогностического управления
MPPI	метод прогностического интегрального управления (Model Predictive Path Integral)
ПИД	пропорционально-интегрально-дифференциальный (регулятор)
SLAM	метод одновременной локализация и картографирования (Simultaneous Localization and Mapping)
ASLAM	активный метод одновременной локализация и картографирования
МБ	мобильный робот
CNN	сверточная нейронная сеть (Convolutional Neural Network)
ИНС	инерциальная навигационная система
AUV	беспилотный подводный аппарат (Autonomous Underwater Vehicle)
ОДУ	обыкновенное дифференциальное уравнение
ЧПУ	числовое программное управление
LQR	линейно-квадратичный регулятор (Linear Quadratic Regulator)
ABS	антиблокировочная система
GPS	глобальная система позиционирования (Global Positioning System)
RRT	быстрый неравномерный дерево-поиск (Rapidly-exploring Random Tree)
БТС	беспилотное транспортное средство
IMU	блок инерциальных измерений (Inertial Measurement Unit)

ROS	операционная система для роботов (Robot Operating System)
DWA	динамическое окно подхода (Dynamic Window Approach)
Teb	Timed Elastic Band — метод планирования траектории с учётом времени и динамики
КПД	коэффициент полезного действия
LSTM	долгая краткосрочная память (Long Short-Term Memory) – тип рекуррентной нейронной сети
GRU	Gated Recurrent Unit — модифицированная рекуррентная сеть с воротами
RNN	рекуррентная нейронная сеть (Recurrent Neural Network)
PSO	оптимизация роем частиц (Particle Swarm Optimization)

Глава 1

Математические основы робототехники

В данной главе исследуется общая задача оптимального управления при наличии фазовых ограничений. С прикладной точки зрения актуальность данной статьи заключается в необходимости иметь дело с экстремалями, которые не удовлетворяют требуемым условиям регулярности, рассмотренным в разделе 1.2. В этом случае некоторые полезные свойства множителей Лагранжа могут оказаться недоступными, если попытаться решить эту задачу численно в рамках непрямого подхода, то есть с использованием принципа максимума. Чтобы преодолеть эту трудность, в разделе 1.3 приведен и обоснован метод регуляризации, основанный на регулярных приближениях исходной задачи. Регулярная задача решена численно, алгоритм представлен в разделе 1.4. Таким образом, решение ε-аппроксимации можно рассматривать как приближенный расчет исходного решения. В разделе 1.5 приведен пример применения предложенного метода к задаче оптимального управления мобильным роботом. Другие примеры регуляризованных задач управления мобильным роботом рассмотрены в разделе 1.6.

При решении задач оптимального управления часто возникает вопрос: почему используется принцип максимума Понтрягина, а не используются «современные» методы, например, обучение робота с подкреплением, или похожие методы? Ответ на этот вопрос изучен и представлен в разделе 1.7. Выводы по первой главе приведены в разделе 1.8.

1.1 Введение

В диссертационной работе изучается движение мобильного робота при ограничениях, накладываемых на фазовые координаты. В качестве критерия оптимальности рассматривается быстродействие, т.е. время, затрачиваемое на движение из заданной точки A в точку B. Задача о наибыстрейшем движении мобильного робота при фазовых ограничениях, или задача о наибыстрейшем обходе роботом препятствия, в упрощенном неинерциальном кинематическом случае может быть описано следующей задачей оптимального управления:

$$\begin{cases} J(p,u,T) := T \to \min, \\ \dot{x}_1 = p \cdot \cos\alpha, \\ \dot{x}_2 = p \cdot \sin\alpha, \\ \dot{\alpha} = u, \\ x(0) = A, \quad x(T) = B, \\ \alpha(0) = \alpha_0, \\ p \in [-1,1], \\ u \in [-1,1], \\ x_1^2 + x_2^2 \geq 1. \end{cases} \tag{1.1}$$

Здесь $x = (x_1, x_2) \in \mathbb{R}^2$ – основная фазовая переменная, $p, u \in \mathbb{R}$ – скалярные параметры управления, $A = (a_1, a_2)$, $B = (b_1, b_2)$ – заданные точки на плоскости, лежащие вне единичного круга. Задача рассматривается на отрезке времени $[0, T]$. Конечный момент времени T не фиксирован, и более того, необходимо найти наименьшее возможное такое время T, за которое можно, начиная движение из точки A, попасть в точку B с помощью каких-либо допустимых управлений $p(t)$, $u(t)$. При этом робот не должен заезжать на единичный круг на плоскости, который считается препятствием. Переменная α – это дополнительная фазовая переменная, отвечающая углу поворота мобильного робота. Угловая скорость поворота робота управляется функцией $u(t)$, а продольная скорость – функцией $p(t)$. Обе функции считаются измеримыми. Робот оснащен обратным ходом, что соответствует отрицательным значениям p. Начальное значение угла поворота α_0 задано. Его конечное значение свободно.

Здесь p - продольная скорость, а α - угол поворота, который определяет линию приложенного тягового усилия. Две переменные u и p являются управляющими параметрами, которые считаются независимыми, поскольку управляющая пара (p, u) принимает значения из квадрата $[-1, 1] \times [-1, 1]$.

В задаче (1.1) реализуется так называемая простейшая униципиклическая модель движения, которая дополняется возможностью разворота робота на месте,

т.е. движением при $p = 0$. На практике она может приближенно соответствовать упрощенной модели трехколесного робота с передним приводом. Если добавить ускорение в задаче (1.1), то эта модель при определенных поправочных коэффициентах дает реальное движение такого робота (тогда продольная и угловая скорости становятся фазовыми переменными, а управление – крутящим моментом, который обеспечивают двигатели, вращающие и разворачивающие переднее колесо).

Отметим, что продольная и угловая скорости в этой модели являются независимыми, что не является характерным для некоторых других мобильных роботов (например, для гусеничного робота). В то же время подход, предложенный в данной работе, может быть применен и ко многим другим моделям, в том числе и к упомянутому выше случаю управляемых крутящих моментов.

Динамика такого типа возникает, например, в неинерциальной модели движения трехколесного мобильного робота с передним приводом. Отличительной особенностью этой модели является возможность поворота робота на месте. Это известное упрощение общей модели. Для получения более сложных и реалистичных моделей смотрите, например, [174] или [175], посвященные моделям AUV. В то же время кинематических уравнений достаточно, чтобы проиллюстрировать вычислительный основанный на принципе максимума, предложенном в этой заметке.

Анализ движения усложняется наличием препятствия, которое робот должен обойти. Это препятствие определяется единичной окружностью на плоскости $\mathbb{R}^2$. Для этой ограниченной системы управления оптимальное по времени движение изучается с помощью применения принципа максимума. Однако прямое применение этого инструмента не приводит к значительному прогрессу в поиске решения. Во-первых, по той причине, что модель одноколесного велосипеда нерегулярна по отношению к каким-либо ограничениям состояния. Во-вторых, возникает особый режим управления в отношении угловой скорости. Поэтому для приближенного вычисления экстремалей рассматривается некоторое регуляризующее ε-возмущение исходной задачи.

Важно отметить существование решения в задаче (1.1) в классе измеримых управлений p и u. Действительно, очевидно, что в этой задаче всегда найдется допустимый процесс, и значит, существование решения в классе измеримых управлений p, u следует из известной теоремы Филиппова, [48] (поскольку правая часть динамической системы линейна по управлению, а множество значений управления – выпуклый компакт). Обозначим решение этой задачи (которых может быть много) через $(x^*, \alpha^*, p^*, u^*, T^*)$.

Итак, задача состоит в оптимальном (по быстродействию) обходе роботом

препятствия, которое задано в виде единичного круга на плоскости, имеется в виду все время, затраченное на движение, согласно (1.1). В работе предлагается некоторый численный метод нахождения решения $(x^*, \alpha^*, p^*, u^*, T^*)$, основанный на принципе максимума Понтрягина, [10].

Отметим, что решение задачи (1.1) с помощью принципа максимума осложнено рядом трудностей. Во-первых, любая допустимая траектория, соприкасающаяся с границей единичного круга, не будет регулярной по Р.В. Гамкрелидзе (см. работу [13] и Главу 6 в [10]). Действительно, в любой точке границы имеет место

$$x_1 \cos\alpha + x_2 \sin\alpha = 0,$$

а значит, градиент функции $\Gamma(x, p, u) := -2p(x_1 \cos\alpha + x_2 \sin\alpha)$ – скалярного произведения правой части динамической системы на градиент фазоограничения – по (p, u) равен нулю. Более того, по этой же причине в задаче (1.1) заведомо не будут выполняться даже и более слабые, чем условия регулярности, известные условия управляемости относительно фазовых ограничений, [50]. (Такая ситуация весьма распространена в технических системах, где обычно количество фазовых переменных превышает количество переменных управления.) Поэтому борелевская мера-множитель Лагранжа, отвечающая за фазовое ограничение, может иметь сингулярную составляющую. В связи с этим, т.е. в связи с отсутствием регулярности и наличием сингулярной составляющей меры, не представляется возможным свести условия принципа максимума к соответствующей краевой задаче, решение которой привело бы к решению задачи управления. Здесь, в частности, возникает проблема нахождения точек выхода на границу и схода с границы фазового ограничения.

Кроме того, применение принципа максимума затруднено наличием особого режима по угловой скорости u. Робот не всегда будет следовать с максимальной по модулю угловой скоростью, т.е. при $|u| = 1$. Однако на участках, где $|u| < 1$, условие максимума, очевидно, не информативно. Возникает проблема нахождения точек выхода на особый режим управления и схода с него.

Оба явления, т.е. отсутствие регулярности относительно фазового ограничения и особый режим управления по угловой скорости, имеют совершенно разную природу. Тем не менее, одним из возможных решений и в первом и во втором случае является подходящая регуляризация исходной задачи. В работе предлагается с помощью малого параметра $\varepsilon > 0$ определенным образом возмутить исходную задачу (1.1), так чтобы возмущенная задача стала регулярной в упомянутом выше смысле. Для этого будут использованы дополнительные переменные управления.

1.2 Общая теория

В этом разделе приведем некоторые факты из теории принципа максимума для задач быстродействия с фазовыми ограничениями.

1.2.1 Постановка задачи

Рассмотрим задачу быстродействия с фазовыми ограничениями в общем виде

$$\begin{aligned} &T \to \min, \\ &\dot{x} = f(x,u), \\ &x(0) = A, \;\; x(T) = B, \\ &u(t) \in U \;\; \text{для п.в.}\, t \in [0,T], \\ &g(x(t)) \le 0 \;\; \forall\, t \in [0,T]. \end{aligned} \tag{1.2}$$

Здесь, $t \in [0,T]$ – время, x – фазовая переменная в $\mathbb{R}^n$, u – переменная управления в $\mathbb{R}^m$. Управляющая функция $u(\cdot)$ измерима.

Множество U имеет вид

$$U := \{u \in \mathbb{R}^m : \varphi(u) \le 0\},$$

где φ – заданная вектор-функция. Точки A и B определяют начальную и конечную позиции (краевые условия). Допустимая траектория $x(\cdot)$ является абсолютно непрерывной функцией, удовлетворяет дифференциальному ограничению $\dot{x}(t) = f(x(t),u(t))$ для п.в. $t \in [0,T]$, начальному и конечному краевому условию, и также неравенству $g(x(t)) \le 0 \;\forall\, t \in [0,T]$, которое задает фазовые ограничения. Число T определяет конечный момент времени движения – оно не фиксировано, но подлежит минимизации.

Отображения

$$f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n, \;\; \varphi : \mathbb{R}^m \to \mathbb{R}^r, \;\; \text{и} \;\; g : \mathbb{R}^n \to \mathbb{R}$$

предполагаются достаточно гладкими (точнее, $f, \varphi \in C^1$, $g \in C^2$).

Пусть $u(\cdot)$ – управление, а $x(\cdot)$ – отвечающая ему траектория на отрезке времени $[0,T]$.

Определение 1 *Тройка (x,u,T) называется допустимым процессом, если выполнены концевые и фазовые ограничения.*

Определение 2 *Допустимый процесс* (x^*, u^*, T^*) *называется оптимальным, если конечное время* T *в* (1.2) *принимает наименьшее возможное значение на множестве всех допустимых процессов (так называемый глобальный минимум).*

Ниже предположим, что в Задаче (1.2) существует по крайней мере один оптимальный процесс (x^*, u^*, T^*).

Для формулировки принципа максимума Понтрягина потребуются некоторые определения и обозначения.

Пусть задана измеримая функция $u(\cdot) : \mathbb{R} \to \mathbb{R}$.

Определение 3 *Замыканием по мере функции* $u(\cdot)$ *в точке* τ *называется множество векторов* $v \in \mathbb{R}^m$ *таких, что*

$$\ell\Big(\{t \in [\tau - \varepsilon, \tau + \varepsilon] : \ |u(t) - v| \le \varepsilon\}\Big) > 0 \ \ \forall \varepsilon > 0,$$

где ℓ *– мера Лебега на* $\mathbb{R}$.

1.2.2 Условия регулярности и невырожденности

Многозначное отображение, ставящее в соответствие точке на прямой замыкание по мере функции в этой точке, обозначим как $\operatorname{Clos} u$. В контексте изучения задач оптимального управления понятие замыкания по мере измеримой функции было предложено в [62]. В работе [73] элементы множества $\operatorname{Clos} u(\tau)$ называются существенными значениями функции u в точке τ. Некоторые свойства замыкания по мере функции описаны также в [58].

Определение 4 *Точка* $u \in U$ *называется регулярной относительно ограничений на управление, если градиенты* $\nabla\varphi^i(u)$, $i \in I(u)$, *линейно независимы.*

Здесь, $I(u) := \{i : \varphi^i(u) = 0\}$ — множество активных индексов; ∇ обозначает классический градиент скалярной функции.

Определение 5 *Фазовые ограничения называются регулярными, если для любого* $x \in \mathbb{R}^n$ *существует вектор* $d = d(x) \in \mathbb{R}^k$ *такой, что*

$$\Big\langle \nabla g^j(x), d \Big\rangle > 0 \quad \forall\, j : g^j(x) = 0.$$

Рассмотрим функцию

$$\Gamma(x,u) := \big\langle \nabla g(x), f(x,u) \big\rangle.$$

Сформулируем условие регулярности.

Определение 6 *Процесс (x^*, u^*, T^*) называется регулярным относительно фазовых ограничений, если для любых $t \in [0, T^*]$ и $u \in \operatorname{Clos} u^*(t)$ точка u регулярна, в то время как существует вектор $d = d(u,t) \in \ker \nabla\varphi^i(u)$, $i \in I(u)$, такой что*

$$\Big\langle \nabla_u \Gamma^j(x^*(t), u), d \Big\rangle > 0 \quad \forall\, j : g^j(x^*(t)) = 0,$$

где ∇_u — частичный градиент относительно u.

Такое понятие регулярности было впервые предложено в [13] и позднее получило дальнейшее развитие в ряде работ, см., например, [70]. Заметим, что в ряде случаев и применительно к принципу максимума приведенное выше условие регулярности может быть значительно ослаблено путем сведения анализа к множеству точек $u \in U$: $\Gamma^j(x^*(t), u) = 0$.

1.2.3 Принцип максимума Понтрягина с фазовыми ограничениями: классический подход

Теория оптимального управления занимается изучением динамических систем, поведение которых можно регулировать с помощью управляющих воздействий, с целью минимизировать (или максимизировать) некоторый функционал качества. Одним из наиболее мощных инструментов анализа таких задач является *принцип максимума Понтрягина*, сформулированный Л.С. Понтрягиным и его учениками в 1950-х годах [10, 16]. Принцип максимума Понтрягина является мощным инструментом для анализа задач оптимального управления.

Первоначально принцип максимума был сформулирован для задач без фазовых ограничений. Однако в реальных приложениях часто возникают ограничения на состояния системы (например, температура не должна превышать определённого значения, или траектория должна оставаться в заданной области как в данной работе). Такие ограничения называются *фазовыми* и существенно усложняют анализ задачи, требуя введения сингулярных мер в сопряжённую систему и дополнительных условий дополняющей нежёсткости. Тем не менее, даже в таких случаях принцип максимума позволяет получить необходимые условия оптимальности, которые могут быть использованы для построения оптимальных траекторий.

Современные обобщения принципа максимума включают случаи разрывных управлений, импульсных систем, задач с запаздыванием и стохастических систем [12, 30, 85]. Однако классическая формулировка с фазовыми ограничениями остается важной в теории и приложениях, включая робототехнику, экономику и аэрокосмические задачи.

Рассмотрим расширенную функцию Гамильтона-Понтрягина

$$\bar{H}(x,u,\psi,\mu) := \big\langle \psi, f(x,u) \big\rangle - \mu\Gamma(x,u),$$

где ψ, μ – переменные из $\mathbb{R}^n$ и $\mathbb{R}$ соответственно.

Теорема 1 *Предположим, что регулярный относительно фазовых ограничений процесс (x^*, u^*, T^*) является оптимальным в Задаче (1.2). Тогда существуют множители Лагранжа: число $\lambda \in [0,1]$, абсолютно непрерывная векторная функция $\psi \in \mathbb{W}_{1,\infty}([0,T^*];\mathbb{R}^n)$, и непрерывная скалярная функция $\mu \in \mathbb{C}([0,T^*];\mathbb{R})$ такие, что*

- *Нетривиальность*

$$\lambda + |\psi(t) - \mu(t)\nabla g(x^*(t))| > 0 \ \ \forall\, t \in [0,T^*];$$

- *Сопряженное уравнение*

$$\dot{\psi}(t) = -\bar{H}'_x(x^*(t), u^*(t), \psi(t), \mu(t)) \ \ \textit{для п.в.}\ t \in [0,T^*];$$

- *Условие максимума*

$$\max_{u\in U} \bar{H}(x^*(t), u, \psi(t), \mu(t)) = \bar{H}(x^*(t), u^*(t), \psi(t), \mu(t))$$

$$\textit{для п.в.}\ t \in [0,T^*],$$

- *Условие трансверсальности*

$$\max_{u\in U} \bar{H}(x^*(T^*), u, \psi(T^*), \mu(T^*)) = \lambda.$$

Кроме того, функция $\mu(\cdot)$ не убывает, и

$$\int_0^{T^*} g(x^*(t))d\mu(t) = 0.$$

Теорема 1 была получена в [58] при некоторых дополнительных предположениях гладкости, а в работах [55] и [68] была доказана в полной общности. Отметим также, что Теорема 1 является уточнением соответствующего результата из [10], Глава 6. Уточнение касается непрерывности меры-множителя $\mu(\cdot)$.

Приведем доказательство теоремы 1. Напомним следующее понятие, [50].

Определение 7 *В концевых точках выполняются условия управляемости относительно фазовых ограничений, если найдутся векторы $u_A, u_B \in U$ такие, что*

$$\langle \nabla g(A), f(A, u_A) \rangle < 0, \quad при \;\; g(A) = 0,$$
$$\langle \nabla g(B), f(B, u_B) \rangle > 0, \quad при \;\; g(B) = 0.$$

Относительно этих условий и условия регулярности имеет место следующее простое утверждение.

Предложение 1 *Предположим, что допустимый процесс (x, u, T) регулярен относительно фазовых ограничений. Тогда в концевых точках выполнены условия управляемости относительно фазовых ограничений.*

Доказательство очевидно следует из определений. Перейдем к доказательству теоремы.

Доказательство Теоремы 1. Рассмотрим заданный оптимальный процесс управления (x^*, u^*, T^*) регулярный относительно фазовых ограничений. В силу Предложения 1 в концевых точках выполнены условия управляемости относительно фазовых ограничений. Тогда в виду результатов, полученных в [50, 58, 70], а также уточнений сделанных в [55] (см. Лемму 4.1), существует невырождающий набор множителей Лагранжа (λ, ψ, μ) такой, что функция $\mu(t)$ непрерывна на $(0, T^*)$, и который удовлетворяет всем условиям принципа максимума кроме условия нетривиальности, которое слабее, чем в теореме:

$$\lambda + \ell\left(\{t \in [0, T^*] : \; \psi(t) - \mu(t)\nabla g(x^*(t)) \neq 0\}\right) > 0. \tag{1.3}$$

Ясно, что в виду (1.3) и Замечания 1 можно считать, что функция μ непрерывна на всем отрезке времени $[0, T^*]$.

Необходимо показать, что в условиях регулярности, сформулированных в этой работе, условие (1.3) влечет

$$\lambda + |\psi(t) - \mu(t)\nabla g(x^*(t))| > 0 \;\; \forall\, t \in [0, T^*]. \tag{1.4}$$

Для простоты изложения предположим, что $u^*(t) \in \operatorname{int} U$ для п.в. t. (Общий случай рассматривается в полной аналогии.) При этом предположении, из условия максимума следует, что

$$H'_u(x^*(t), u, \psi(t), \lambda) = \mu(t)\Gamma'_u(x^*(t), u) \ \ \forall\, u \in \mathcal{U}(t), \ \ t \in [0, T^*], \tag{1.5}$$

где H означает обычную функцию Гамильтона-Понтрягина:

$$H(x, u, \psi, \lambda) := \langle \psi, f(x, u) \rangle - \lambda f_0(x, u).$$

Рассмотрим множество точек

$$\mathcal{T} := \{t \in [0, T^*] : g(x^*(t)) = 0, \, \exists\, u_0 \in \mathcal{U}(t) : \, \Gamma(x^*(t), u_0) = 0\}.$$

Отметим, что множество $\mathcal{T}$ замкнуто.

Рассмотрим произвольную точку $t \in \mathcal{T}$. В силу регулярности, $\Gamma'_u(x^*(t), u_0) \neq 0$ для некоторого $u_0 = u_0(t) \in \mathcal{U}(t)$. Поэтому взяв скалярное произведение (1.5) с $\Gamma'_u(x^*(t), u_0)$, учитывая, что $|\Gamma'_u(x^*(t), u_0)|^2 > 0$, получаем, что

$$\mu(t) = \frac{\Big\langle H'_u(x^*(t), u_0(t), \psi(t), \lambda), \Gamma'_u(x^*(t), u_0(t)) \Big\rangle}{|\Gamma'_u(x^*(t), u_0)|^2} \ \ \forall\, t \in \mathcal{T}. \tag{1.6}$$

Используя стандартные рассуждения, основанные на соображении компактности, и условие регулярности, можно заключить, что $|\Gamma'_u(x^*(t), u_0)| \geq \varepsilon \ \forall\, t \in \mathcal{T}$ для некоторого $\varepsilon > 0$. Тогда из (1.6) имеет место следующая оценка

$$\mu(t) \leq \operatorname{const}(\lambda + |\psi(t)|) \ \ \forall\, t \in \mathcal{T}. \tag{1.7}$$

Здесь константа const не зависит от t.

Предположим, что условие (1.4) нарушено. Тогда $\lambda = 0$, и существует точка $t_0 \in [0, T^*]$ такая, что $\psi(t_0) = \mu(t_0)\nabla g(x^*(t_0))$. Для $t \in [0, T^*]$ положим

$$\widetilde{\mu}(t) := \mu(t) - \mu(t_0), \ \ \widetilde{\psi}(t) := \psi(t) - \mu(t_0)\nabla g(x^*(t)).$$

В силу Замечания 4, новый набор множителей Лагранжа $(\lambda, \widetilde{\psi}, \widetilde{\mu})$ снова удовлетворяет принципу максимума и (1.3), причем $\widetilde{\mu}(t_0) = 0$, $\widetilde{\psi}(t_0) = 0$. Для удобства переобозначим $\widetilde{\mu}$, $\widetilde{\psi}$ снова через μ, и ψ соответственно.

Вначале рассмотрим случай $t_0 \in \mathcal{T}$. После переобозначений оценка (1.7) принимает следующий вид

$$|\mu(t)| \leq C_1 |\psi(t)| \ \ \forall\, t \in \mathcal{T}. \tag{1.8}$$

Здесь $C_1 > 0$ – некоторая константа.

В силу свойства Липшиц-непрерывности сопряженной функции ψ, имеем

$$|\psi(t)| \le K|t - t_0| \ \ \forall\, t \in [0, T^*], \tag{1.9}$$

при некотором $K > 0$.

Докажем оценку

$$|\mu(t)| \le C_1 K|t - t_0| \ \ \forall\, t \in [0, T^*]. \tag{1.10}$$

Действительно, при $t \in \mathcal{T}$, эта оценка вытекает из (1.8) и (1.9). Рассмотрим точку $t \in [0, T^*]$, $t > t_0$, такую, что $t \notin \mathcal{T}$. Поскольку множество $\mathcal{T}$ замкнуто, а $t_0 \in \mathcal{T}$, существует точка $t_* \in \mathcal{T}$, $t_0 \le t_* < t$, такая, что $\mathcal{T} \cap (t_*, t] = \varnothing$. Для точки t_* оценка (1.10) уже доказана. В то же время из предложения 4.8 в [58] вытекает постоянство функции μ на $[t_*, t]$. Эти рассуждения доказывают (1.10) справа от t_0. Точно такие же рассуждения проводятся и слева от точки t_0. Оценка (1.10) установлена.

Используя полученную оценку и (1.9), из сопряженного уравнения имеем

$$|\psi(t)| \le KC_2(C_1 + 1) \int_{t_0}^{t} (s - t_0) ds, \ \ t \in [0, T^*],$$

где $C_2 > 0$.

Отсюда

$$|\psi(t)| \le KC_2(C_1 + 1) \frac{(t - t_0)^2}{2} \ \ \forall\, t \in [0, T^*]. \tag{1.11}$$

Применяя аналогичную (1.10) оценку, но используя (1.11) вместо (1.9),

$$|\mu(t)| \le KC_2(C_1 + 1) C_1 \frac{(t - t_0)^2}{2} \ \ \forall\, t \in [0, T^*].$$

Из этой оценки и из (1.11), используя сопряженное уравнение, получаем

$$|\psi(t)| \le KC_2^2(C_1 + 1)^2 \frac{|t - t_0|^3}{3!} \ \ \forall\, t \in [0, T^*].$$

Продолжая такой итеративный процесс, на n-ом шаге, имеем

$$|\psi(t)| \le \frac{KC_2^n(C_1 + 1)^n}{(n + 1)!} \cdot |t - t_0|^{n+1} \ \ \forall\, t \in [0, T^*].$$

При этом

$$|\mu(t)| \le \frac{KC_2^n(C_1+1)^nC_1}{(n+1)!} \cdot |t-t_0|^{n+1} \ \ \forall\, t \in [0,T^*].$$

Переход к пределу при $n \to \infty$ показывает, что $\psi = 0$, $\mu = 0$. Значит, все множители равны нулю, что противоречит условию нетривиальности (1.3).

Случай $t_0 \notin \mathcal{T}$ легко свести к уже рассмотренному случаю с помощью предложения 4.8 из [58]. Условие (1.3) доказано. $\square$

Ниже приведены несколько замечаний, которые будут использованы далее.

Замечание 1 *Из условий принципа максимума вытекает известный закон сохранения вдоль экстремали, [10, 57],*

$$\max_{u\in U} \bar{H}(x^*(t), u, \psi(t), \mu(t)) = \lambda \ \ \forall\, t \in [0,T^*].$$

Замечание 2 *Из условий принципа максимума следует, что $\mu(\cdot)$ постоянна на любом отрезке времени $[a,b]$, на котором оптимальная траектория лежит во внутренности фазовых ограничений, т.е. когда $g(x^*(t)) < 0$ $\forall\, t \in [a,b]$.*

Замечание 3 *Ввиду правила множителей Лагранжа, условие максимума влечет*

$$\bar{H}'_u(x^*(t), u^*(t), \psi(t), \mu(t)) \in N_U(u^*(t)), \quad \text{для п.в. } t \in [0,T^*],$$

где N_U означает предельный нормальный конус ко множеству, [8]. Предполагая, что все точки множества U регулярны, можно утверждать существование дополнительного множителя Лагранжа – векторной функции $\xi : [0,T^*] \to \mathbb{R}^r$ с неотрицательными компонентами такой, что $\langle \xi(t), \varphi(u^*(t)) \rangle = 0$,

$$\bar{H}'_u(x^*(t), u^*(t), \psi(t), \mu(t)) = \xi(t)\varphi'(u^*(t)), \quad \text{для п.в. } t \in [0,T^*].$$

Замечание 4 *Пусть набор множителей (λ, ψ, μ) удовлетворяет принципу максимума, и $a \in \mathbb{R}$ – произвольное число. Тогда набор множителей*

$$\Big(\lambda, \psi(t) + a\nabla g(x^*(t)), \mu(t) + a\Big)$$

снова удовлетворяет всем условиям принципа максимума, см. [10, 77]. Поэтому можно считать, что $\mu(T^*) = 0$ и тогда $\mu(t) \ge 0$, или, что эквивалентно, $\mu(0) = 0$ и тогда $\mu(t) \le 0$.

Принцип максимума из Теоремы 1 предоставляет так называемую полную систему уравнений для нахождения оптимального процесса, в которой количество уравнений формально равно количеству неизвестных. Действительно, рассмотрим для простоты $k = 1$ и, таким образом, μ является скаляром. При условии регулярности, указанном в определении 6 на границе фазового ограничения замечание 3 позволяет выразить μ через другие экстремальные значения. Вне границы фазового ограничения, то есть внутри области, функция μ постоянна согласно замечанию 2. Поэтому, используя свойство непрерывности, мерумножитель μ можно выразить в терминах состояния, совместного состояния и функции управления, то есть через x^*, ψ, u^*. В то же время коэффициент непрерывности удобен для определения точек соединения. Если одновременно с этим с помощью условия максимума получается выразить и экстремальное управление, то условия принципа максимума сводятся к решению соответствующей краевой задачи, которая может быть решена стандартными методами. Опираясь на эти соображения, можно предложить алгоритм вычисления экстремалей. Такой алгоритм описан разделе 1.4 и применен для нахождения приближенного решения задачи о наибыстрейшем обходе препятствия мобильным роботом (1.1).

Отметим, что условие регулярности, сформулированное в Определении 6, нельзя назвать удобным для предварительного анализа задачи, поскольку оно оперирует с заданным оптимальным процессом, который, вообще говоря, заранее неизвестен (его нам и следует вычислить). Также понятие регулярности, указанное в определении 6, и принцип максимума, сформулированный в Теореме 1, не всегда применимы в реальных задачах и особенно применительно к робототехнике, где число фазовых переменных обычно превышает количество управлений. В связи с этим представляется более практичным иметь некоторое априорное условие регулярности в терминах функций f, φ и g, которое бы гарантировало регулярность любого допустимого процесса, а значит, и заведомую применимость метода, основанного на Теореме 1. Такое априорное условие регулярности несложно привести.

Условие Р) Для всех $x \in \mathbb{R}^n$, $u \in U$ таких, что $g(x) = \Gamma(x, u) = 0$, имеет место, что точка u регулярна, и

$$\Gamma'_u(x, u) \notin \operatorname{lin}\Big\{\nabla\varphi^i(u),\ i \in I(u)\Big\}.$$

Это априорное условие легко проверить для конкретной задачи. В контексте задач быстродействия, его естественно дополнить следующим замечанием.

Замечание 5 *Пусть* $0 \in \operatorname{int} f(x,U)\ \forall\, x$. *Тогда Условие P) достаточно проверять на множестве*

$$\Big\{x \in \mathbb{R}^n, u \in U : g(x) = \Gamma(x,u) = 0, f(x,u) \neq 0\Big\}.$$

Тогда Условие P) гарантирует регулярность любого оптимального процесса задачи. Это наблюдение связано с тем, что в необходимых условиях оптимальности условие P) является достаточным для проверки только оптимального процесса, а также с тем, что выполняется следующее простое утверждение.

Лемма 1 *Пусть* (x^*, u^*, T^*) *– оптимальный процесс в Задаче* (1.2). *Предположим, что выполнено условие:* $0 \in \operatorname{int} f(x,U)\ \forall\, x \in \mathbb{R}^n$. *Тогда* $\exists\, \delta > 0$:

$$\ell\Big(t \in [0,T^*] : \Big|f(x^*(t),u^*(t))\Big| \leq \delta\Big) = 0.$$

Доказательство проводится стандартными рассуждениями от противного.

Это модифицированное Условие P) используется при анализе движения мобильного робота.

Уточненный принцип максимума дает полную систему уравнений для нахождения оптимального процесса в следующем смысле. При условии R) можно выразить μ в терминах других экстремальных значений, используя условие максимума, когда дуга экстремума лежит на границе множества, ограниченного состоянием. За пределами границы, то есть внутри ограничения состояния, функция μ постоянна в соответствии с замечанием 2. В то же время непрерывность показателя-множителя служит для определения точек пересечения экстремальной дуги. Следовательно, μ может быть выражено в терминах состояния, совместного состояния и функции управления на всем временном интервале $[0,T^*]$, если только максимальное условие позволяет выразить экстремальное управление. Затем условия принципа максимума сводятся к соответствующей краевой задаче, которая может быть решена численно.

Тем не менее, условие R) может не выполняться в различных приложениях, особенно в отношении робототехники: например, данные задачи (1.1) не удовлетворяют условию R). Поэтому в данной работе предлагается метод возмущений, позволяющий преодолеть явление нерегулярности. Между тем, идея состоит в том, чтобы применить теорему 1 к возмущенной задаче.

1.3 Принцип максимума Понтрягина с фазовыми ограничениями: подход на основе регуляризации

В данном разделе диссертационной работы рассмотрен метод регуляризации исходной задачи. Сначала разобран и обоснован метод регуляризации в общем слечае. Далее приведена явная формула для вычисления множителя μ. Также описан алгоритм метода регуляризации.

1.3.1 Регуляризация в общем случае

Обратимся к регуляризации общей задачи быстродействия с фазовыми ограничениями. Существует, очевидно, множество различных подходов к регуляризации.

Рассмотрим возмущенную задачу

$$\begin{aligned} &J(u,v,T) := T + \int_0^T |v|^2 dt \to \min, \\ &\dot{x} = f(x,u) + \varepsilon v, \\ &x(0) = A, \\ &x(T) = B, \\ &\varphi(u) \le 0, \\ &g(x) \le 0. \end{aligned} \tag{1.12}$$

Здесь, $v = (v_1, v_2, .., v_n)$ – дополнительное n-мерное управление.

Здесь фазовые ограничения предполагаются регулярными в смысле определения 5, в то время как регулярность оптимального процесса в задаче (1.2), указанном в определении 6 может не выполняться. Ограничения на управление $\varphi(u) \le 0$ регулярны, если регулярны все точки множества U. Эти предположения (которые естественны для ряда прикладных задач) ниже считаются *a priori* выполненными. В этих предположениях проверим для данных задачи (1.12) выполнение условия регулярности в смысле Определения 6.

Имеем

$$\Gamma^i(x,u,v) = \big\langle \nabla g^i(x), f(x,u) \big\rangle + \varepsilon \big\langle \nabla g^i(x), v \big\rangle,$$

for i от 1 до k.

Градиент Γ^i относительно (u,v) равен

$$(\nabla g^i(x) f'_u(x,u), \varepsilon \nabla g^i(x)),$$

в то время как градиенты активных ограничений управления относительно (u, v) имеют вид:

$$(\nabla\varphi^j(u), 0), \quad j \in I(u).$$

Возьмем некоторый вектор $z \in \cap_{j\in I(u)} \ker \nabla\varphi^j(u)$. Рассмотрим вектор d из определения 5 и достаточно большое число $a > 0$. Тогда вектор $(z, 0) + a(0, d)$ является таким, что выполняется определение 6.

Таким образом, благодаря регулярности фазовых ограничений и регулярности ограничений управления любой допустимый процесс в задаче (1.12) является регулярным относительно фазовых ограничений. Тогда получается следующее простое утверждение.

Предложение 2 *Предположим, что фазовые ограничения регулярны, и регулярны все точки $u \in U$. Тогда любой допустимый процесс задачи (1.12) регулярен относительно фазовых ограничений для всех $\varepsilon > 0$.*

Исследуем вопрос о существовании решения в возмущенной задаче. Заметим, что никаких геометрических ограничений на управление v не имеется, и следовательно, исходя из вида функционала, можно считать, что управление v ограничено лишь по норме $\|\cdot\|_{L_2}$. Однако, при известных предположениях можно гарантировать, что решение задачи (1.12) принадлежит пространству $\mathbb{W}_{1,2}([0, T])$.

Теорема 2 *Предположим, что:*

(a) *множество U компактно;*

(b) *множество $f(x, U)$ выпукло при всех x;*

(c) *имеет место оценка $|f(x, u)| \le \mathrm{const}(1 + |x|)\ \forall\, u \in U$;*

(d) *в исходной задаче (1.2) существует допустимый процесс.*

Тогда существует решение $(x_\varepsilon, u_\varepsilon, v_\varepsilon, T_\varepsilon)$ задачи (1.12) в классе траекторий $x \in \mathbb{W}_{1,2}([0, T])$ и управлений $u \in \mathbb{L}_\infty([0, T])$, $v \in \mathbb{L}_2([0, T])$ такое, что $x_\varepsilon \rightrightarrows x_0$, $\|v_\varepsilon\|_{\mathbb{L}_2} \to 0$ при $\varepsilon \to 0$, где x_0 – одно из решений исходной задачи (1.2).

Доказательство. Прежде всего отметим существование решения в исходной задаче (1.2). Действительно, в силу сделанных предположений (a)–(d) это следует из теоремы Филиппова [48]. Обозначим это решение через (x^*, u^*, T^*).

Возьмем натуральное число N и рассмотрим ε, N-задачу, которая получается из задачи (1.12) путем добавления одного ограничения на управления $|v| \leq N$. Процесс $(x^*, u^*, 0, T^*)$ является допустимым в ε, N-задаче, и значит, в силу той же теоремы у нее существует решение, которое мы обозначим через $(x_{\varepsilon,N}, u_{\varepsilon,N}, v_{\varepsilon,N}, T_{\varepsilon,N})$. Устремим $N \to \infty$ и найдем искомый процесс $(x_\varepsilon, u_\varepsilon, v_\varepsilon, T_\varepsilon)$.

Имеем

$$T_{\varepsilon,N} + \|v_{\varepsilon,N}\|^2_{L_2} = J(x_{\varepsilon,N}, u_{\varepsilon,N}, v_{\varepsilon,N}, T_{\varepsilon,N}) \leq J(x^*, u^*, 0, T^*) = T^*. \tag{1.13}$$

Отсюда поскольку $T_{\varepsilon,N} > 0$ выводим, что последовательность $\{T_{\varepsilon,N}\}$ ограничена. Переходя к подпоследовательности $T_{\varepsilon,N} \to T_\varepsilon$ при $N \to \infty$. Аналогично, переходя к подпоследовательности, $v_{\varepsilon,N} \xrightarrow{w} v_\varepsilon$ слабо в $\mathbb{L}_2([0, T_\varepsilon])$.

Из условия (c) и из (1.13) с помощью стандартных оценок, включающих неравенство Гронуола, получаем, что последовательность функций $x_{\varepsilon,N}$ ограничена в равномерной метрике $\|x_{\varepsilon,N}\|_C \leq \text{const}$. Поэтому семейство функций

$$y_{\varepsilon,N}(t) := A + \int_0^t f(x_{\varepsilon,N}(s), u_{\varepsilon,N}(s))ds$$

равномерно ограничено и равностепенно непрерывно. Поэтому, переходя к подпоследовательности, можно считать, что $y_{\varepsilon,N} \rightrightarrows y_\varepsilon$ равномерно на $[0, T_\varepsilon]$. При этом $\dot{y}_{\varepsilon,N} \xrightarrow{w} \dot{y}_\varepsilon$ слабо в L_2.

Семейство функций

$$z_{\varepsilon,N}(t) = \int_0^t v_{\varepsilon,N}(s)ds$$

также равномерно ограничено и равностепенно непрерывно. Это очевидно следует из оценки (1.13), так как по неравенству Коши-Буняковского

$$\left| \int_{t_1}^{t_2} v^j_{\varepsilon,N}(t)dt \right| \leq \sqrt{|t_1 - t_2|} \cdot \|v^j_{\varepsilon,N}\|_{L_2}, \;\; j = 1, .., n.$$

Поэтому, переходя к подпоследовательности $z_{\varepsilon,N} \rightrightarrows z_\varepsilon$, и значит, имеем $x_{\varepsilon,N} \rightrightarrows x_\varepsilon := y_\varepsilon + z_\varepsilon$. Из условий (a), (b), в силу равномерной сходимости $x_{\varepsilon,N}$, из леммы Мазура следует, что $\dot{y}_\varepsilon(t) \in f(x_\varepsilon(t), U)$ для п.в. t. По лемме об измеримом селекторе существует управление $u_\varepsilon : [0, T_\varepsilon] \to U$ такое, что $\dot{y}_\varepsilon(t) = f(x_\varepsilon(t), u_\varepsilon(t))$ для п.в. t. Таким образом,

$$x_\varepsilon(t) = A + \int_0^t f(x_\varepsilon(s), u_\varepsilon(s))ds + \varepsilon \int_0^t v_\varepsilon(s)ds,$$

и x_ε удовлетворяет фазовым и граничным условиям. Значит, $(x_\varepsilon, u_\varepsilon, v_\varepsilon, T_\varepsilon)$ – допустимый процесс в задаче (1.12). Из оценки (1.13) в силу слабой полунепрерывности снизу функционала J имеем

$$T_\varepsilon + \|v_\varepsilon\|^2_{L_2} \le T^*.$$

Перейдем к пределу при $\varepsilon \to 0$. Рассуждая аналогично, переходя к подпоследовательности, $T_\varepsilon \to T_0$, $x_\varepsilon \rightrightarrows x_0 \in \mathbb{W}_{1,2}([0,T_0])$, $v_\varepsilon \overset{w}{\to} v_0 \in \mathbb{L}_2([0,T_0])$, $\varepsilon \to 0$. Аналогично, найдется управление u_0 такое, что процесс (x_0, u_0, T_0) является допустимым в задаче (1.2). Однако,

$$T_0 + \|v_0\|^2_{L_2} \le T^* \quad \Rightarrow \quad T_0 = T^*,\ v_0 = 0,$$

и $v_\varepsilon \to 0$ сильно в силу оптимальности. Таким образом, процесс (x_0, u_0, T_0) также является решением задачи (1.2). $\square$

В классе траекторий из $\mathbb{W}_{1,2}([0,T])$ также справедлив принцип максимума. Его несложно вывести по аналогии с доказательством Теоремы 2, применив к ε, N-задаче принцип максимума для траекторий класса $\mathbb{W}_{1,\infty}([0,T])$ и перейдя в полученных условиях к пределу при $N \to \infty$.

Для простоты изложения предположим, что $g(A) + g(B) < 0$, т.е. что концы траектории лежат строго во внутренности фазового ограничения. С одной стороны, такое предположение позволяет избежать в пределе эффекта вырождающегося принципа максимума, см. [50, 77], – случая, который требует отдельного рассмотрения. С другой стороны, оно не является существенным для задачи об обходе препятствия мобильным роботом. Тогда, переходя стандартным образом к пределу при $N \to \infty$ в условиях принципа максимума, можно утверждать существование нетривиального набора множителей Лагранжа $(\lambda_\varepsilon, \psi_\varepsilon, \mu_\varepsilon)$, удовлетворяющего сопряженной системе, условию максимума и условию трансверсальности по времени. (Условие максимума выводится с помощью интегральной формы и слабой сходимости.) Условие нетривиальности

$$\lambda_\varepsilon + |\psi_\varepsilon(t) - \mu_\varepsilon(t)\nabla g(x_\varepsilon(t))| > 0 \ \ \forall\, t \in [0, T_\varepsilon] \tag{1.14}$$

выводится с помощью условия регулярности, которое выполняется в ε-задаче, и рассуждений полностью аналогичных приведенным в [70] при доказательстве Теоремы 3.1 (отличие заключается лишь в использовании расширенной версии неравенства Гронуолла, см., например, Упражнение 3.4 в [52]).

Рассмотрим ε-условие максимума по v. Имеем

$$v_\varepsilon(t) \in \operatorname*{argmax}_{v \in \mathbb{R}^n} \Big(\varepsilon \big\langle v, \psi_\varepsilon(t) - \mu_\varepsilon(t)\nabla g(x_\varepsilon(t)) \big\rangle - \lambda_\varepsilon |v|^2 \Big).$$

Отсюда в виду (1.14) получаем, что $\lambda_\varepsilon > 0$, и значит,

$$v_\varepsilon(t) = \frac{\varepsilon}{2\lambda_\varepsilon}\Big(\psi_\varepsilon(t) - \mu_\varepsilon(t)\nabla g(x_\varepsilon(t))\Big). \tag{1.15}$$

Таким образом, показано, что функция v_ε ограничена, и значит, поскольку выполнены условия регулярности, можно применить Теорему 1. Поэтому функция μ_ε непрерывна, и, следовательно, v_ε также непрерывна. Отметим, что сопряженная система задачи (1.12) отличается от исходной сопряженной системы задачи (1.2) наличием малого регуляризирующего члена $\varepsilon\mu_\varepsilon(t)g''(x_\varepsilon(t))v_\varepsilon(t)$.

Здесь функция v_ε определяется выражением (1.15). При каждом $\varepsilon > 0$ эта функция ограничена и даже, как было отмечено, непрерывна. Однако, неограниченные значения не исключаются при $\varepsilon \to 0$.

Заметим, что добавив в функционал J малый член $\varepsilon\|u\|_{L_2}$, наряду с регулярностью относительно фазовых ограничений, можно добиться и выполнения усиленного условия Лежандра. Тогда в виду результатов, полученных в [58], можно утверждать, что множитель μ_ε является даже липшицевой функцией (однако, с константой Липшица зависящей от ε).

1.3.2 Регуляризация задачи управления трехколесным мобильным роботом

Соответствующее возмущение (1.1) требуется по следующим двум причинам: а) невыполнение условия P) и б) наличие сингулярного режима управления относительно u. В этом разделе предлагается ε-регуляризация данных (1.1), цель которой такое возмущение исходной задачи, чтобы в возмущенной задаче условие P) выполнялось, в то время как условие максимума становится информативным над ε-допустимым множеством управления. Последнее свойство решает задачу сингулярного управления.

Любой допустимый процесс Задачи (1.1) нерегулярен относительно фазовых ограничений, как только траектория касается границы допустимой области. Действительно, в любой точке границы области имеем $x_1\cos\alpha + x_2\sin\alpha = 0$, и значит, градиент функции $\Gamma(x,p,u) = -2p(x_1\cos\alpha + x_2\sin\alpha)$ по (p,u) равен нулю. Поэтому Теорему 1 не удается применить для численного решения поставленной задачи, как это было сделано, например, в [68]. Однако, предлагается рассмотреть некоторое возмущение исходной задачи такое, что в возмущенной задаче выполняется Условие P). Теорему 1 будем применять к возмущенной задаче.

Итак, зафиксируем малое число $\varepsilon > 0$ и рассмотрим ε-задачу

$$\begin{cases} J(p,u,v,w,T) := T \to \min, \\ \dot{x}_1 = p \cdot \cos\alpha + \varepsilon w_1 + \varepsilon v_1, \\ \dot{x}_2 = p \cdot \sin\alpha + \varepsilon w_2 + \varepsilon v_2, \\ \dot{\alpha} = u, \\ x(0) = A, \;\; x(T) = B, \\ \alpha(0) = \alpha_0, \\ p^2 + w_1^2 + w_2^2 \le 1, \\ u^2 + v_1^2 + v_2^2 \le 1, \\ x_1^2 + x_2^2 \ge 1. \end{cases} \tag{1.16}$$

Здесь $w = (w_1, w_2) \in \mathbb{R}^2$ и $v = (v_1, v_2) \in \mathbb{R}^2$ – дополнительные переменные управления, отвечающие за регуляризацию задачи. Ясно, что задача (1.16) является релаксацией задачи (1.1), так как любой допустимый процесс задачи (1.1) будет допустимым и в задаче (1.16), если положить $v = w = 0$. Кроме того, решение в задаче (1.16) существует. Обозначим ее решение (какое-либо) через $\wp_\varepsilon := (x_\varepsilon, \alpha_\varepsilon, p_\varepsilon, u_\varepsilon, v_\varepsilon, w_\varepsilon, T_\varepsilon)$.

Покажем, что ε-задача регулярна. Имеет место следующее утверждение.

Предложение 3 *Любое решение $\wp_\varepsilon$ задачи (1.16) является регулярным относительно фазовых ограничений.*

Доказательство. Имеем,

$$\Gamma(x,\alpha,p,u,v,w) = -2x_1(p \cdot \cos\alpha + \varepsilon w_1 + \varepsilon v_1) - 2x_2(p \cdot \sin\alpha + \varepsilon w_2 + \varepsilon v_2).$$

В силу Замечания 5 условие регулярности будет заведомо выполнено для любого оптимального процесса, если на множестве

$$x, \alpha, p, u, v, w: \;\; x_1^2 + x_2^2 = 1, \; \Gamma(x,\alpha,p,u,v,w) = 0, \; f(\alpha,p,u,v,w) \ne 0$$

градиент функции Γ по (p,u,v,w) и градиенты активных ограничений на управление линейно независимы, т.е. линейно независима следующая система из трех векторов (множитель 2 сокращен):

$$a_1 := \begin{bmatrix} -x_1 \cdot \cos\alpha - x_2 \cdot \sin\alpha \\ 0 \\ -\varepsilon x_1 \\ -\varepsilon x_2 \\ -\varepsilon x_1 \\ -\varepsilon x_2 \end{bmatrix}, \quad a_2 := \begin{bmatrix} p \\ 0 \\ 0 \\ 0 \\ w_1 \\ w_2 \end{bmatrix}, \quad a_3 := \begin{bmatrix} 0 \\ u \\ v_1 \\ v_2 \\ 0 \\ 0 \end{bmatrix}.$$

Здесь функция f – правая часть динамической системы в (1.16), т.е.

$$f := \begin{bmatrix} p \cdot \cos\alpha + \varepsilon w_1 + \varepsilon v_1 \\ p \cdot \sin\alpha + \varepsilon w_2 + \varepsilon v_2 \\ u \end{bmatrix}.$$

Векторы a_1, a_2, a_3, очевидно, попарно линейно независимы при активных ограничениях на управление, так как $|x| = 1$. Поэтому, как легко видеть, если три вектора линейно зависимы, то первый вектор (с точностью до знака) является линейной комбинацией двух других: $a_1 = \alpha a_2 + \beta a_3$, причем $\alpha \neq 0$ и $\beta \neq 0$. Тем самым $u = 0$, и

$$\begin{aligned} &-x_1 \cdot \cos\alpha - x_2 \cdot \sin\alpha = \alpha p, \\ &-\varepsilon x_1 = \beta v_1,\ -\varepsilon x_2 = \beta v_2, \\ &-\varepsilon x_1 = \alpha w_1,\ -\varepsilon x_2 = \alpha w_2. \end{aligned}$$

Тогда, подставляя в выражения для Γ, имеем

$$\Gamma(x, \alpha, p, u, v, w)/2 = \alpha(p^2 + |w|^2) + \beta|v|^2 = \alpha + \beta = 0.$$

Таким образом, поскольку рассматривается случай, когда ограничения на управление активны, $\alpha = -\beta$. Учитывая, что $|x| = 1$, имеем $|v| = 1 \Rightarrow |\alpha| = |\beta| = \varepsilon \Rightarrow |w| = 1 \Rightarrow p = 0$. Следовательно, получили $p = u = 0$ и $w = -v \Rightarrow f = 0$. Это означает, что Условие P) выполнено с учетом Замечания 5. Тем самым установлено, что ε-задача регулярна. $\square$

Задача (1.16) является ослаблением задачи (1.1), поэтому любой допустимый процесс задачи (1.1) будет допустимым для задачи (1.16), если положить $v = w = 0$. Кроме того, решение задачи (1.16) существует. Рассмотрим любое решение и обозначим его как $(x_\varepsilon, \alpha_\varepsilon, p_\varepsilon, u_\varepsilon, v_\varepsilon, w_\varepsilon, T_\varepsilon)$. Нетрудно заметить, что решения ε-задачи приближают решения задачи (1.1), так как верно следующее прямое утверждение.

Лемма 2 *Для любой числовой последовательности $\varepsilon_k \to 0$ существует решение x^*, α^*, T^* задачи (1.1) и подпоследовательность ε_{k_i} такие, что $x_{\varepsilon_{k_i}} \rightrightarrows x^*$, $\alpha_{\varepsilon_{k_i}} \rightrightarrows \alpha^*$, и $T_{\varepsilon_{k_i}} \to T^*$. Если решение задачи (1.1) единственно, то $\varepsilon_{k_i} = \varepsilon_k$.*

Утверждение является простым следствием слабой секвенциальной компактности шара в гильбертовом пространстве и того, что ε-задача является релаксацией линейно-выпуклой задачи (1.1).

Таким образом, установлено, что если требуется найти численно хотя бы одно (какое-либо) решение задачи (1.1), то при малом $\varepsilon > 0$ таким решением может служить решение ε-задачи (1.16), которое в свое очередь можно найти приближенно с помощью решения условий принципа максимума Понтрягина, поскольку ε-задача регулярна ввиду предложения 3 и тем самым условия принципа ε-максимума могут быть сведены к соответствующей краевой задаче.

Перейдем к условиям принципа ε-максимума для задачи (1.16). Фазовое ограничение $|x|^2 \le 1$ заменим на эквивалентное ему $\frac{1}{2}|x|^2 \le \frac{1}{2}$, что, очевидно, не отразится на свойстве регулярности ε-задачи. Имеем

$$\begin{aligned}\bar{H}(x,\alpha,p,u,v,w,\psi,\mu) :=\\ &= \psi_1(p\cos\alpha + \varepsilon w_1 + \varepsilon v_1) + \psi_2(p\sin\alpha + \varepsilon w_2 + \varepsilon v_2) + \psi_3 u - \mu\Gamma(x,\alpha,p,u,v,w)\\ &= (\psi_1 + \mu x_1)(p\cos\alpha + \varepsilon w_1 + \varepsilon v_1) + (\psi_2 + \mu x_2)(p\sin\alpha + \varepsilon w_2 + \varepsilon v_2) + \psi_3 u,\end{aligned}$$

где $\psi = (\psi_1, \psi_2, \psi_3) \in \mathbb{R}^3$, $\mu \in \mathbb{R}$ – сопряженные переменные.

Благодаря теореме 1, которая применима в силу предложения 3, существуют (зависящие от ε) число $\lambda \ge 0$, абсолютно непрерывная вектор-функция $\psi(t)$ и непрерывная убывающая скалярная функция $\mu(t)$, все зависящие от ε, такие, что выполняются следующие условия.

Выполнено условие нетривиальности:

$$\lambda + |\psi_1(t) + \mu(t)x_{\varepsilon 1}(t)| + |\psi_2(t) + \mu(t)x_{\varepsilon 2}(t)| + |\psi_3(t)| > 0 \ \ \forall\, t \in [0, T_\varepsilon]. \tag{1.17}$$

Уравнение для сопряженной переменной принимает вид

$$\begin{cases} \dot{\psi}_1(t) = -\mu(t)(p_\varepsilon(t)\cos\alpha_\varepsilon(t) + \varepsilon w_{\varepsilon 1}(t) + \varepsilon v_{\varepsilon 1}(t)),\\ \dot{\psi}_2(t) = -\mu(t)(p_\varepsilon(t)\sin\alpha_\varepsilon(t) + \varepsilon w_{\varepsilon 2}(t) + \varepsilon v_{\varepsilon 2}(t)),\\ \dot{\psi}_3(t) = p_\varepsilon(t)\sin\alpha_\varepsilon(t)(\psi_1(t) + \mu(t)x_{\varepsilon 1}(t)) - p_\varepsilon(t)\cos\alpha_\varepsilon(t)(\psi_2(t) + \mu(t)x_{\varepsilon 2}(t)), \end{cases} \tag{1.18}$$

причем $\psi_3(T_\varepsilon) = 0$.

Условие максимума распадается на два независимых условия

$$\begin{aligned}\max_{p^2+|w|^2\le 1}\Big((\psi_1(t) + \mu(t)x_{\varepsilon 1}(t))(p\cos\alpha_\varepsilon(t) + \varepsilon w_1) +\\ (\psi_2(t) + \mu(t)x_{\varepsilon 2}(t))(p\sin\alpha_\varepsilon(t) + \varepsilon w_2)\Big) =\\ (\psi_1(t) + \mu(t)x_{\varepsilon 1}(t))(p_\varepsilon(t)\cos\alpha_\varepsilon(t) + \varepsilon w_{\varepsilon 1}(t)) +\\ (\psi_2(t) + \mu(t)x_{\varepsilon 2}(t))(p_\varepsilon(t)\sin\alpha_\varepsilon(t) + \varepsilon w_{\varepsilon 2}(t)),\end{aligned} \tag{1.19}$$

$$\begin{aligned}&\max_{u^2+|v|^2\le 1}\Big((\psi_1(t) + \mu(t)x_{\varepsilon 1}(t))\varepsilon v_1 + (\psi_2(t) + \mu(t)x_{\varepsilon 2}(t))\varepsilon v_2 + \psi_3(t)u\Big) =\\ &(\psi_1(t) + \mu(t)x_{\varepsilon 1}(t))\varepsilon v_{\varepsilon 1}(t) + (\psi_2(t) + \mu(t)x_{\varepsilon 2}(t))\varepsilon v_{\varepsilon 2}(t) + \psi_3(t)u_\varepsilon(t)\end{aligned} \tag{1.20}$$

для п.в. $t \in [0, T_\varepsilon]$.

Закон сохранения и условие трансверсальности по времени дают

$$\max_{p^2+|w|^2\le 1,\, u^2+|v|^2\le 1} \bar{H}(x_\varepsilon(t), \alpha_\varepsilon(t), p, u, v, w, \psi(t), \mu(t)) = \lambda \ \ \forall\, t \in [0, T_\varepsilon]. \tag{1.21}$$

Кроме этого, функция $\mu(t)$ постоянна на тех отрезках времени, где оптимальная траектория целиком лежит во внутренности фазового ограничения: $|x_\varepsilon(t)| > 1$. Также, не ограничивая общности, согласно Замечанию 4 можно полагать, что $\mu(T_\varepsilon) = 0$. Таким образом, $\mu(t) \ge 0$.

Утверждение 1. Имеет место более сильное, чем (1.17), условие нетривиальности

$$|\psi_1(t) + \mu(t)x_{\varepsilon 1}(t)| + |\psi_2(t) + \mu(t)x_{\varepsilon 2}(t)| + |\psi_3(t)| > 0 \ \ \forall\, t \in [0, T_\varepsilon]. \tag{1.22}$$

Действительно, если (1.22) нарушается, то из (1.21) имеем $\lambda = 0$, и следовательно, нарушается и (1.17).

Таким образом, множитель λ далее не играет существенной роли, поскольку в выражениях ниже он не участвует и можно считать после нормировки, что

$$(\psi_1(T_\varepsilon))^2 + (\psi_2(T_\varepsilon))^2 = 1.$$

Теперь найдем выражения для вычисления управлений $p_\varepsilon, u_\varepsilon, v_\varepsilon, w_\varepsilon$ и меры-множителя Лагранжа μ.

Из условия максимума (1.20) имеем

$$u_\varepsilon(t) = \frac{\psi_3(t)}{\sqrt{A_\varepsilon}}, \ \ v_{\varepsilon 1}(t) = \frac{\varepsilon(\psi_1(t) + \mu(t)x_{\varepsilon 1}(t))}{\sqrt{A_\varepsilon}}, \ \ v_{\varepsilon 2}(t) = \frac{\varepsilon(\psi_2(t) + \mu(t)x_{\varepsilon 2}(t))}{\sqrt{A_\varepsilon}},$$

где

$$A_\varepsilon := \varepsilon^2(\psi_1(t) + \mu(t)x_{\varepsilon 1}(t))^2 + \varepsilon^2(\psi_2(t) + \mu(t)x_{\varepsilon 2}(t))^2 + (\psi_3(t))^2 > 0.$$

Утверждение 2. Имеет место более сильное, чем условие (1.22), условие нетривиальности

$$|\psi_1(t) + \mu(t)x_{\varepsilon 1}(t)| + |\psi_2(t) + \mu(t)x_{\varepsilon 2}(t)| > 0 \ \ \forall\, t \in [0, T_\varepsilon]. \tag{1.23}$$

Действительно, вблизи границы области, используя Замечание 3, из Условия P) можно вывести оценку

$$|\mu(t)| \le \mathrm{const}(|\psi_1(t)| + |\psi_2(t)|),$$

где const – некоторая постоянная (зависящая от ε, но не от t). Поэтому, если (1.23) нарушается в какой-то точке $\tau \in [0, T_\varepsilon]$, то используя неравенство Гронуолла, считая при этом, что $\mu(\tau) = \psi_1(\tau) = \psi_2(\tau) = 0$ (что возможно в силу Замечания 4), из сопряженной системы выводим, что $\mu(t) = \psi_1(t) = \psi_2(t) = 0$ $\forall\, t \in [0, T_\varepsilon]$. Это однако противоречит (1.22) при $t = T_\varepsilon$.

Из условия максимума (1.19) имеем

$$p_\varepsilon(t) = \frac{1}{\sqrt{B_\varepsilon}} \cdot \Big((\psi_1(t) + \mu(t) x_{\varepsilon 1}(t)) \cos\alpha_\varepsilon(t) + (\psi_2(t) + \mu(t) x_{\varepsilon 2}(t)) \sin\alpha_\varepsilon(t) \Big),$$

$$w_{\varepsilon 1}(t) = \frac{1}{\sqrt{B_\varepsilon}} \cdot \varepsilon(\psi_1(t) + \mu(t) x_{\varepsilon 1}(t)), \quad w_{\varepsilon 2}(t) = \frac{1}{\sqrt{B_\varepsilon}} \cdot \varepsilon(\psi_2(t) + \mu(t) x_{\varepsilon 2}(t)),$$

где

$$B_\varepsilon = \Big((\psi_1(t) + \mu(t) x_{\varepsilon 1}(t)) \cos\alpha_\varepsilon(t) + (\psi_2(t) + \mu(t) x_{\varepsilon 2}(t)) \sin\alpha_\varepsilon(t) \Big)^2 + \\ + \varepsilon^2 (\psi_1(t) + \mu(t) x_{\varepsilon 1}(t))^2 + \varepsilon^2 (\psi_2(t) + \mu(t) x_{\varepsilon 2}(t))^2 > 0.$$

Вычислим μ. На границе фазового ограничения $\Gamma(x_\varepsilon, \alpha_\varepsilon, p_\varepsilon, u_\varepsilon, v_\varepsilon, w_\varepsilon) = 0$, т.е. (зависимость от времени в дальнейшем для сокращения записи опуская)

$$x_{\varepsilon 1}(p_\varepsilon \cos\alpha_\varepsilon + \varepsilon w_{\varepsilon 1} + \varepsilon v_{\varepsilon 1}) + x_{\varepsilon 2}(p_\varepsilon \sin\alpha_\varepsilon + \varepsilon w_{\varepsilon 2} + \varepsilon v_{\varepsilon 2}) = 0.$$

Подставляя сюда полученные выше выражения для $p_\varepsilon, w_\varepsilon, v_\varepsilon$, получаем

$$\frac{(b\mu + a)b + \varepsilon^2\mu + c}{\sqrt{(b\mu + a)^2 + \varepsilon^2\mu^2 + 2c\mu + d}} = -\frac{\varepsilon^2\mu + c}{\sqrt{\varepsilon^2\mu^2 + 2c\mu + d + \psi_3^2}},$$

где

$$\begin{aligned} a &= \psi_1 \cos\alpha_\varepsilon + \psi_2 \sin\alpha_\varepsilon, \\ b &= x_{\varepsilon 1} \cos\alpha_\varepsilon + x_{\varepsilon 2} \sin\alpha_\varepsilon, \\ c &= \varepsilon^2 (x_{\varepsilon 1}\psi_1 + x_{\varepsilon 2}\psi_2), \\ d &= \varepsilon^2 (\psi_1^2 + \psi_2^2). \end{aligned}$$

Возводим в квадрат и получаем уравнение 4-ой степени на μ,

$$r_4\mu^4 + r_3\mu^3 + r_2\mu^2 + r_1\mu + r_0 = 0,$$

где

$$\begin{aligned} r_4 &= b^4\varepsilon^2 + b^2\varepsilon^4, \\ r_3 &= 2b^2 \left(b^2 c + ab\varepsilon^2 + 2c\varepsilon^2\right), \\ r_2 &= a^2\varepsilon^2 \left(b^2 - \varepsilon^2\right) + 2abc \left(2b^2 + \varepsilon^2\right) + b^2 \left(3c^2 + b^2 d + 2d\varepsilon^2\right) + \left(b^2 + \varepsilon^2\right)^2 \psi_3^2, \\ r_1 &= 2 \left(b^2 cd + a^2 c \left(b^2 - \varepsilon^2\right) + ab \left(c^2 + d \left(b^2 + \varepsilon^2\right)\right) + (ab + c) \left(b^2 + \varepsilon^2\right) \psi_3^2\right), \\ r_0 &= a \left(-ac^2 + ab^2 d + 2bcd\right) + (ab + c)^2 \psi_3^2. \end{aligned}$$

Решая его, после исключения приобретенных корней, получаем формулу:

$$\mu = -\frac{r_3}{4r_4} \pm \frac{C_7}{2} + \frac{1}{2}\sqrt{2C_5 - C_6 - \frac{\frac{4r_2r_3}{r_4^2} - \frac{r_3^3}{r_4^3} - \frac{8r_1}{r_4}}{4C_7}}, \tag{1.24}$$

где

$$C_7 := \sqrt{C_5 + C_6},$$
$$C_6 := C_4 + \frac{1}{32^{1/3}r_4}C_3,$$
$$C_5 := \frac{r_3^2}{4r_4^2} - \frac{2r_2}{3r_4},$$
$$C_4 := \frac{2^{1/3}C_1}{3r_4C_3},$$
$$C_3 := \left(C_2 + \sqrt{-4C_1^3 + C_2^2}\right)^{1/3},$$
$$C_2 := 2r_2^3 - 9r_1r_2r_3 + 27r_0r_3^2 + 27r_1^2r_4 - 72r_0r_2r_4,$$
$$C_1 := r_2^2 - 3r_1r_3 + 12r_0r_4.$$

Формула (1.24) используется для расчета $\mu(t)$ только на границе фазового ограничения, т.е. на тех участках траектории робота, где $x_{\varepsilon 1}^2(t) + x_{\varepsilon 2}^2(t) = 1$. Вне границы μ не изменяется со временем. Кроме того, μ непрерывна, что позволяет найти точки стыка (т.е. точки, в которых экстремаль выходит на границу или сходит с нее). Полученное значение для μ подставляется в выражения для $p_\varepsilon, v_\varepsilon, w_\varepsilon$ и в сопряженное уравнение. Таким образом, условия принципа максимума сводятся к решению соответствующей краевой задачи.

Итак, построена регулярная аппроксимация исходной задачи (1.1) и приведены нужные формулы для сведения к краевой задаче. При этом возмущенная задача (1.16), как и исходная, также является задачей быстродействия, что мы специально отмечаем. В общем случае, т.е. для задачи управления (1.2), регуляризация быстродействия быстродействием представляется нетривиальной задачей. По всей видимости, такой тип регуляризации осуществим лишь при некоторых дополнительных предположениях. Тем не менее, если пренебречь свойством быстродействия аппроксимирующей задачи и, тем самым, возмущать помимо динамики еще и минимизирующий функционал, то оказывается возможным предложить достаточно общую схему регуляризации задачи управления. Более того, можно добиться даже более сильного условия регулярности относительно фазового ограничения. Такую общую схему регуляризации обсуждаем в следующем параграфе.

1.3.3 Применение метода регуляризации к мобильному роботу

Теперь применим описанную выше схему возмущения к задаче о движении мобильного робота (1.1). При этом понятно, что задача (1.12) представляет собой некоторую общую схему возмущения относительно фазового ограничения, но не учитывает возможных особых режимов. Однако, очевидно, что не все время в пути робот будет разворачиваться с максимальной по модулю угловой скоростью и поэтому анализ особого режима по угловой скорости является критически важным для модели (1.1). Проблема особых режимов можно решить по аналогии с помощью дополнительной регуляризации задачи и введения дополнительных переменных управления. Другим подходом может служить возмущение минимизируемого функционала с помощью малой интегральной добавки.

В диссертационной работе используется именно второй подход как альтернативу первому подходу, рассмотренному в п. 1.3.2.

Рассмотрим возмущенную задачу:

$$J(p,u,v,T) := T + \varepsilon\int_0^T |p|^2dt + \varepsilon\int_0^T |u|^2dt + \int_0^T |v|^2dt \to \min, \tag{1.25a}$$

$$\begin{cases} \dot{x}_1 = p\cdot\cos\alpha + \varepsilon v_1, \\ \dot{x}_2 = p\cdot\sin\alpha + \varepsilon v_2, \\ \dot{\alpha} = u, \end{cases} \tag{1.25b}$$

$$x(0) = A,\ x(T) = B,\ \alpha(0) = \alpha_0, \tag{1.25c}$$

$$p\in[-1,1],\quad u\in[-1,1], \tag{1.25d}$$

$$\frac{1}{2}(x_1^2 + x_2^2) \geq \frac{1}{2}. \tag{1.25e}$$

Здесь $v = (v_1, v_2)$ – дополнительная двумерная переменная управления, отвечающая за регуляризацию. (Поясним, что рассмотрение переменной v_3 излишне, поскольку градиент фазового ограничения по α тождественно равен нулю.) Ясно, что задача (1.25) представляет собой смягчение задачи (1.1), поскольку любой допустимый процесс в задаче (1.1) является допустимым в задаче (1.25), если положить $v = 0$. Кроме того, существует решение задачи (1.25). Обозначим это решение через $(x_\varepsilon, \alpha_\varepsilon, p_\varepsilon, u_\varepsilon, v_\varepsilon, T_\varepsilon)$.

Лемма 3 *Для любой числовой последовательности $\{\varepsilon_k\}$, сходящейся к нулю, существует решение $(x^*, \alpha^*, u^*, p^*, T^*)$ задачи (1.1) и подпоследовательность $\{\varepsilon_{k_i}\}$ такий, что $x_{\varepsilon_{k_i}} \rightrightarrows x^*$, $\alpha_{\varepsilon_{k_i}} \rightrightarrows \alpha^*$ и $T_{\varepsilon_{k_i}} \to T^*$ при $i\to\infty$. Если решение задачи (1.1) единственно, то можно считать, что $\varepsilon_{k_i} = \varepsilon_k$.*

Это утверждение является простым следствием слабой последовательной компактности единичного шара в гильбертовом пространстве и того факта, что рассматриваемая ε-задача является упрощением задачи (1.1).

Важной особенностью ε-задачи, которая делает ее пригодной для дальнейшего анализа, является тот факт, что условие R) уже выполнено в отношении возмущенных данных. Следовательно, теорема 1 может быть применена. Далее мы записываем его условия.

Имеем

$$\Gamma(x,\alpha,p,u,v) = -x_1(p\cdot\cos\alpha+\varepsilon v_1) - x_2(p\cdot\sin\alpha+\varepsilon v_2),$$

$$\begin{aligned}\bar{H}(x,\alpha,p,u,v,\psi,\mu,\lambda) &= (\psi_1+\mu x_1)(p\cos\alpha+\varepsilon v_1)+(\psi_2+\mu x_2)(p\sin\alpha+\varepsilon v_2)+\\ &\quad \psi_3 u-\varepsilon\lambda|p|^2-\varepsilon\lambda|u|^2-\lambda|v|^2.\end{aligned}$$

Рассмотрим зависящие от параметра ε число $\lambda > 0$ (случай $\lambda = 0$, как несложно проверить, исключен), абсолютно непрерывную вектор-функцию $\psi(t)$ и непрерывную и убывающую скалярную функцию $\mu(t)$, которые удовлетворяют принципу максимума. Имеем

$$\begin{cases}\dot\psi_1(t) = -\mu(t)(p_\varepsilon(t)\cos\alpha_\varepsilon(t)+\varepsilon v_{\varepsilon1}(t)),\\ \dot\psi_2(t) = -\mu(t)(p_\varepsilon(t)\sin\alpha_\varepsilon(t)+\varepsilon v_{\varepsilon2}(t)),\\ \dot\psi_3(t) = p_\varepsilon(t)\left[\sin\alpha_\varepsilon(t)(\psi_1(t)+\mu(t)x_{\varepsilon1}(t))-\cos\alpha_\varepsilon(t)(\psi_2(t)+\mu(t)x_{\varepsilon2}(t))\right],\end{cases} \tag{1.26}$$

а также

$$\psi_3(T_\varepsilon) = 0. \tag{1.27}$$

Анализ условия максимума по переменным v_1, v_2 дает:

$$\begin{aligned}v_{\varepsilon1}(t) &= \frac{\varepsilon}{2\lambda}\Big(\psi_1(t)+\mu(t)x_{\varepsilon1}(t)\Big),\\ v_{\varepsilon2}(t) &= \frac{\varepsilon}{2\lambda}\Big(\psi_2(t)+\mu(t)x_{\varepsilon2}(t)\Big),\end{aligned} \tag{1.28}$$

Условие максимума по переменной p влечет:

$$p_\varepsilon(t) = \operatorname{sgn} a_\varepsilon(t)\min\Big\{\frac{|a_\varepsilon(t)|}{2\varepsilon\lambda},1\Big\}, \tag{1.29}$$

где

$$a_\varepsilon(t) = (\psi_1(t)+\mu(t)x_{\varepsilon1}(t))\cos\alpha_\varepsilon(t)+(\psi_2(t)+\mu(t)x_{\varepsilon2}(t))\sin\alpha_\varepsilon(t). \tag{1.30}$$

Из условия максимума по переменной u следует, что

$$u_\varepsilon(t) = \operatorname{sgn}\psi_3(t)\min\Big\{\frac{|\psi_3(t)|}{2\varepsilon\lambda}, 1\Big\}. \tag{1.31}$$

Заметим, что $|\psi_1(T_\varepsilon)| + |\psi_2(T_\varepsilon)| > 0$. Действительно, иначе, учитывая, что $\mu(T_\varepsilon) = 0$[1] и $\psi_3(T_\varepsilon) = 0$ в виду условия трансверсальности по времени получаем $\lambda = 0$. Таким образом, после нормировки можно считать, что

$$(\psi_1(T_\varepsilon))^2 + (\psi_2(T_\varepsilon))^2 = 1. \tag{1.32}$$

Найдем формулы для множителей λ и μ. Используя условие трансверсальности по времени, учитывая, что $\mu(T_\varepsilon) = 0$, $\psi_3(T_\varepsilon) = 0$, и (1.32), решая соответствующее квадратное уравнение и отбрасывая лишние корни, приходим к следующим выражениям для λ.

Случай 1. Пусть $a_\varepsilon(T_\varepsilon) \le -\frac{\varepsilon^2}{\sqrt{1-\varepsilon}}$. Тогда

$$\lambda = \lambda_1 = \frac{-a_\varepsilon(T_\varepsilon) + \sqrt{a_\varepsilon^2(T_\varepsilon) + \varepsilon^2(\varepsilon+1)}}{2(\varepsilon+1)}. \tag{1.33}$$

Случай 2. Пусть $a_\varepsilon(T_\varepsilon) \in (-\frac{\varepsilon^2}{\sqrt{1-\varepsilon}}, \frac{\varepsilon^2}{\sqrt{1-\varepsilon}})$. Тогда

$$\lambda = \lambda_2 = \sqrt{\frac{a_\varepsilon^2(T_\varepsilon) + \varepsilon^3}{4\varepsilon}}. \tag{1.34}$$

Случай 3. Пусть $a_\varepsilon(T_\varepsilon) \ge \frac{\varepsilon^2}{\sqrt{1-\varepsilon}}$. Тогда

$$\lambda = \lambda_3 = \frac{a_\varepsilon(T_\varepsilon) + \sqrt{a_\varepsilon^2(T_\varepsilon) + \varepsilon^2(\varepsilon+1)}}{2(\varepsilon+1)}. \tag{1.35}$$

На границе фазового ограничения имеем

$$\Gamma(x_\varepsilon, \alpha_\varepsilon, p_\varepsilon, u_\varepsilon, v_\varepsilon) = 0,$$

то есть

$$x_{\varepsilon 1}(p_\varepsilon\cos\alpha_\varepsilon + \varepsilon v_{\varepsilon 1}) + x_{\varepsilon 2}(p_\varepsilon\sin\alpha_\varepsilon + \varepsilon v_{\varepsilon 2}) = 0.$$

(Зависимость от времени в дальнейшем для удобства записи опущена.) Подставляя в это равенство полученные выше выражения для p_ε и v_ε, при $a_\varepsilon \le -2\varepsilon\lambda$, $a_\varepsilon \in (-2\varepsilon\lambda, 2\varepsilon\lambda)$ и $a_\varepsilon \ge 2\varepsilon\lambda$ получаем следующие выражения для вычисления μ.

[1] В теореме 1, это не является ограничительным для установки $\mu(T^*) = 0$ (см. Замечание 3.1 в [70]).

Случай А: $a_\varepsilon(t) \le -2\varepsilon\lambda$. Тогда

$$\mu = \frac{2\lambda}{\varepsilon^2}(x_{\varepsilon 1}\cos\alpha_\varepsilon + x_{\varepsilon 2}\sin\alpha_\varepsilon) - (\psi_1 x_{\varepsilon 1} + \psi_2 x_{\varepsilon 2}). \tag{1.36}$$

Случай Б: $a_\varepsilon(t) \in (-2\varepsilon\lambda, 2\varepsilon\lambda)$. Тогда

$$\mu = -\frac{(\psi_1\cos\alpha_\varepsilon + \psi_2\sin\alpha_\varepsilon)(x_{\varepsilon 1}\cos\alpha_\varepsilon + x_{\varepsilon 2}\sin\alpha_\varepsilon) + \varepsilon^3(\psi_1 x_{\varepsilon 1} + \psi_2 x_{\varepsilon 2})}{(x_{\varepsilon 1}\cos\alpha_\varepsilon + x_{\varepsilon 2}\sin\alpha_\varepsilon)^2 + \varepsilon^3}. \tag{1.37}$$

Случай В: $a_\varepsilon(t) \ge 2\varepsilon\lambda$. Тогда

$$\mu = -\frac{2\lambda}{\varepsilon^2}(x_{\varepsilon 1}\cos\alpha_\varepsilon + x_{\varepsilon 2}\sin\alpha_\varepsilon) - (\psi_1 x_{\varepsilon 1} + \psi_2 x_{\varepsilon 2}). \tag{1.38}$$

Заметим, что каждого λ_1, λ_2 и λ_3 реализуются свои случаи А, Б и В (итак, всего имеется 9 разных случаев).

Вне границы фазового ограничения μ не изменяется со временем. Кроме того, μ непрерывна, что позволяет найти точки стыка (т.е. точки, в которых экстремаль выходит на границу или сходит с нее). Полученное значение для μ подставляется в выражения для $p_\varepsilon, v_\varepsilon$ и в сопряженное уравнение. Таким образом, условия принципа максимума сводятся к решению соответствующей краевой задачи.

Теперь, считая выполненным (1.32), $\mu(T_\varepsilon) = 0$ и $\psi_3(T_\varepsilon) = 0$, получим необходимые оценки сверху на $|\psi|$ и μ в зависимости от ε. Поскольку из формул для λ в общем случае имеем $\lambda = O(\varepsilon^{-1})$, то из формул для v_ε находим

$$\mu(t) \le \mathrm{const}\cdot\Big(\varepsilon^{-2}|v_\varepsilon(t)| + |\psi_1(t)| + |\psi_2(t)|\Big).$$

Отсюда, используя сопряженное уравнение (1.26) и неравенство Гронуолла (здесь напомним, что v_ε непрерывна), стандартными рассуждениями получаем, что

$$|\psi(t)| \le \|v_\varepsilon\|_{L_2}\cdot O(\varepsilon^{-2}) \quad\Rightarrow\quad \mu(0) = \|v_\varepsilon\|_{L_2}\cdot O(\varepsilon^{-2}).$$

Поскольку $\|v_\varepsilon\|_{L_2} \to 0$ при $\varepsilon \to 0$, имеем $\mu(0) = o(\varepsilon^{-2})$. Однако, знанием о скорости сходимости $\|v_\varepsilon\|_{L_2}$ к нулю мы не обладаем. Таким образом, можно гарантировать лишь следующее. Пусть

$$|\psi_1(T_\varepsilon)| + |\psi_2(T_\varepsilon)| = \varepsilon^2.$$

Тогда ψ и μ ограничены на всем отрезке движения одной константой, которая не зависит от ε.

В завершение теоретической части отметим, что радиус окружности задающей фазовое ограничение, может быть выбран произвольным образом, т.е. не обязательно быть единичным. В таком случае, немного меняется формула для μ, где появляется радиус окружности r. Тем не менее с точки зрения вычислений, случай именно единичной окружности интересен тем, что он является пограничным в следующем смысле. Ясно, что робот может двигаться одновременно с максимальными угловой и продольной скоростями только по единичной окружности. Если как-либо изменить радиус, то это уже не так: при $r > 1$ угловая скорость не максимальна, а при $r < 1$ – продольная.

1.4 Алгоритм решения задачи методом возмущений

Рассмотрим следующую схему приближенного решения исходной задачи (1.1).

$$\begin{cases} J(p,u,T) := T \to \min, \\ \dot{x}_1 = p \cdot \cos\alpha, \\ \dot{x}_2 = p \cdot \sin\alpha, \\ \dot{\alpha} = u, \\ x(0) = A, \;\; x(T) = B, \\ \alpha(0) = \alpha_0, \\ p \in [-1,1], \\ u \in [-1,1], \\ x_1^2 + x_2^2 \geq 1. \end{cases}$$

Фиксируем некоторое $\varepsilon \in (0,1)$ и переходим к возмущенной задаче (1.39), которая регуляризована.

$$J(p,u,v,T) := T + \varepsilon \int_0^T |p|^2 dt + \varepsilon \int_0^T |u|^2 dt + \int_0^T |v|^2 dt \to \min, \tag{1.39a}$$

$$\begin{cases} \dot{x}_1 = p \cdot \cos\alpha + \varepsilon v_1, \\ \dot{x}_2 = p \cdot \sin\alpha + \varepsilon v_2, \\ \dot{\alpha} = u, \end{cases} \tag{1.39b}$$

$$x(0) = A, \; x(T) = B, \; \alpha(0) = \alpha_0, \tag{1.39c}$$

$$p \in [-1,1], \quad u \in [-1,1], \tag{1.39d}$$

$$\frac{1}{2}(x_1^2 + x_2^2) \geq \frac{1}{2}. \tag{1.39e}$$

К возмущенной задаче применяем численный метод, предложенный в [67, 68]. Используем решение этой задачи для заданного ε в качестве начального приближения к решению задачи (1.39) с меньшим значением ε. Решение этой задачи для наименьшего из рассмотренных значений ε считаем приближением к решению исходной задачи (1.1).

Принцип максимума для этой задачи сводится к краевой задаче с использованием формул из раздела 1.3.3. Краевая задача принципа максимума решается методом стрельбы в обратном времени (т.е. перебор значений параметров, определяющих решение краевой задачи, производится при $t = T$). В качестве начального времени для расчета берется $T = 0$, в то время как движение осуществляется в обратном направлении. В чем:

- Угол θ на плоскости рассматривается как соответствующая параметризация единичной окружности $(\psi_1(0))^2+(\psi_2(0))^2 = 1$. Таким образом, θ задает начальное значение для ψ. Напомним, что $\psi_3(0) = \psi_4(0) = 0$, поскольку время $T = 0$ соответствует точке B на плоскости. Другой угол на плоскости β отвечает за конечный угол поворота робота (т.е. значение $\alpha(T)$). Конечная продольная скорость - это число γ. Конечная угловая скорость равна числу δ. Обсуждаются границы для γ, δ.

- Выполняется итерация по $(\theta, \beta, \gamma, \delta)$. Эта четверка фиксирована при условии, что $\mu = 0$, в то время как x, p, α, ψ вычисляются.

- Как только граница достигнута, μ вычисляется по формуле, рассмотренной в предыдущем разделе. Условие непрерывности для μ проверяется. Если это условие нарушено, то соответствующий угол съемки $(\theta, \beta, \gamma, \delta)$ отбрасывается.

- Далее движение вдоль границы происходит в соответствии с предоставленными формулами. В каждой точке границы проверяется "выход из граничного условия": μ принимается постоянным, x, p, α, ψ вычисляются. Если x попадает в A с заданной точностью, в то время как p, α попадает в p_0, α_0 без нарушения заданной области, то строится экстремаль. Если кто-то снова добирается до границы государственного ограничения, процедура повторяется.

Мы также следим за монотонностью μ: если монотонность нарушена, то соответствующая четверка $(\theta, \beta, \gamma, \delta)$ исключается. Напомним, что $\mu(t)$ - убывающая

функция. Затем мы приводим ε к нулю и анализируем ε-решения для $\varepsilon = 0.1$, $\varepsilon = 0.01$, $\varepsilon = 0.001$ и т.д. и оцениваем порядок сходимости к решению исходной задачи.

Алгоритм.

Итак, рассматривается краевая задача для системы обыкновенных дифференциальных уравнений (1.39b) и (1.26) и краевых условий (1.39c) и (1.27). При $t = T$ имеем следующие условия на решение: мобильный робот находится в точке B, (1.39c), и третья компонента сопряженной переменной равна нулю, (1.27). Используя (1.32), считаем, что $\psi_1(T) = \cos(\theta)$ и $\psi_2(T) = \sin(\theta)$, где $\theta \in [0, 2\pi)$. Также положим $\alpha(T) = \beta$, где $\beta \in [0, 2\pi)$. Если значения θ и β заданы, то при $t = T$ заданы все функции, входящие в рассматриваемую систему дифференциальных уравнений. Суть метода стрельбы состоит в том, чтобы численно найти значения параметров θ и β, чтобы решение $(x(t), \psi(t))$ системы дифференциальных уравнений (1.39b) и (1.26), удовлетворяющее указанным краевым условиям при $t = T$, также удовлетворяло бы (в заданной точностью[2]) краевым условиям при $t = 0$. Такая траектория является решением краевой задачи, полученной из принципа максимума Понтрягина, и, следовательно, является экстремалью рассматриваемой задачи оптимального управления.

При заданном наборе значений величин θ и β, система уравнений (1.39b) и (1.26) численно интегрируется в обратном времени классическим методом Рунге-Кутты четвертого порядка с постоянным шагом. Перед началом интегрирования вычисляется значение λ по формулам (1.33)-(1.35). Считая, что исходная точка B не принадлежит границе, полагаем $\mu(t) = 0$, пока траектория не достигнет границы фазового ограничения (1.39e). При вычислении правых частей дифференциальных уравнений функции управления $p(t)$, $u(t)$, $v_1(t)$ и $v_2(t)$ вычисляются по формулам (1.29), (1.31) и (1.28) соответственно.

Правые части интегрируемых дифференциальных уравнений, вообще говоря, негладкие, ввиду возможных переключений режимов управления $p(t)$ и $u(t)$, т.е. при t когда выполнено $|a_\varepsilon(t)| = 2\varepsilon\lambda$ (переключение $p(t)$, $a_\varepsilon(t)$ задано выражением (1.30)) и $|\psi_3(t)| = 2\varepsilon\lambda$ (переключение $u(t)$). В эти моменты времени соответствующее управление становится или перестает быть максимальным (равным 1 по абсолютной величине). Чтобы избежать потерю точности в окрестности таких точек, при расчете правых частей в методе Рунге-Кутты, считаем шаг неуспешным, если при его вычислении происходит такое переключение для одного из

[2]Значения всех параметров, определяющих точность численного решения задачи, приведены в следующей части.

управлений. В таком случае уменьшаем шаг интегрирования вдвое и повторяем попытку выполнения шага. Уменьшение шага прекращается после семикратного уменьшения начального шага (что отвечает уменьшению начального шага примерно на два порядка), после чего продолжаем интегрирование с начальным шагом по времени с новым режимом по рассмотренному управлению.

Если траектория пересекает границу фазового ограничения (1.39e), то в точке пересечения вычисляем значение меры-множителя Лагранжа μ используя выражения (1.36)-(1.38). Если его абсолютное значение не превосходит заданной (малой) величины, считаем, что в этой точке $\mu(t)$ непрерывно, в этот момент времени происходит заход на границу фазового ограничения, дальше траектория следует по этой границе, расчет меры-множителя на каждом шаге по времени происходит по (1.36)-(1.38). Если же его значение больше заданной величины, то, ввиду разрыва $\mu(t)$ в этой точке, построенная траектория не является частью экстремали и, следовательно, не представляет интереса.

Для траекторий проходящих по границе фазового ограничения, осуществляем следующую процедуру. На каждом шаге по времени фиксируем μ и продолжаем интегрирование системы дифференциальных уравнений с этим фиксированным значением меры-множителя Лагранжа. Такая траектория сходит с границы, и может являться частью экстремали, соединяющую границу с точкой A или снова заходящей на границу.

Для упрощения вычислений, при движении вне границы, всегда считаем $\mu = 0$, поэтому при построении траектории, уходящей с границы в момент времени $t = \widehat{t}$, $\mu(\widehat{t}) = \widehat{\mu}$, проводили расчет такой траектории полагая $\mu(t) = 0$ для $t \geq \widehat{t}$ и переопределяя, в соответствии с Замечанием 4,

$$(\psi_1(\widehat{t}), \psi_2(\widehat{t})) \leftarrow (\psi_1(\widehat{t}), \psi_2(\widehat{t})) + \widehat{\mu}(x_1(\widehat{t}), x_2(\widehat{t}))$$

(значение $\psi_3(\widehat{t})$ остается неизменным).

На каждом шаге по времени измеряется расстояние от текущей позиции робота до краевого условия при $t = 0$ в (1.39c), и если оно меньше заданного значения, то построенная траектория считается решением краевой задачи. Если таких траекторий найдено несколько, то выбираем ту из них, которая отвечает меньшему значению функционала (1.39a).

Были использованы следующие условия, по которым траектория отвергалась из кандидатов в экстремали рассмотренной задачи:

- ◇ время движения робота больше заданного (достаточно большого) значения,
- ◇ при следовании по границе: нарушение монотонности μ,

◇ при сходе с границы: нарушение фазового ограничения.

Для наибольшего значения ε начальный выбор значений параметров θ и β проводили на равномерной сетке в квадрате $[0, 2\pi)^2$, локальные минимумы соответствующих невязок уточнялись методом деления пополам. Для бóльших значений ε в качестве начального значения указанных параметров брали их «оптимальное» значение найденное для меньшего значения ε.

1.5 Результаты численных экспериментов

В этом разделе приведены результаты численного решения задачи оптимального управления (1.39) с использованием принципа максимума, описанного в разделе 1.3. Траектории, удовлетворяющие двухточечной краевой задаче для обыкновенных дифференциальных уравнений в (1.39) и (1.26), которые являются результатом характеристики экстремалей задачи (1.39), заданной условиями максимального принцип. Оптимальным решением является экстремум, связанный с функцией минимальной стоимости J из (1.39).

Граничные значения в конечный момент времени T параметризуются углами β и θ следующим образом: $\alpha(T) = \beta, \psi_1 = \cos(\theta)$ и $\psi_2 = \sin(\theta)$. Изменяя параметры β и θ, интегрируем управляющие ОДУ в обратном направлении во времени, чтобы найти (если они существуют) траектории, удовлетворяющие начальным граничным условиям в (1.39). Также находим траектории, пересекающие границу препятствия, такие, что в точке пересечения множитель μ является непрерывным. Далее, интегрирование системы для таких траекторий приводит к их продолжению вдоль границы набора ограничений состояния. Любая точка, в которой экстремали, эволюционирующие вдоль границы ограничения состояния, покидают ее, находится путем интегрирования управляющих ОДУ (с равномерной сеткой вдоль границы) с фиксированным значением множителя μ. Если траектория ухода удовлетворяет начальным граничным значениям в (1.39)), то она представляет собой отрезок экстремали.

Используя этот алгоритм, решена задача (1.39) для $A = (-2, -0,5), B = (2, 0), \alpha_0 = 0$ и для уменьшения значений ε с 1 до $0,005$. Соответствующие значения функции затрат J и времени прохождения по оптимальным решениям приведены в табл. 1.1.

Таблица 1.1: Функция затрат J и времени в пути для оптимального решения задачи (1.39) как функция от ε.

ε	J	T_ε
1	3.24	2.16
0.5	3.70	2.93
0.1	4.30	4.16
0.05	4.33	4.28
0.01	4.32	4.32
0.005	4.32	4.32

Можно сделать вывод, что при уменьшении ε время в пути монотонно увеличивается до значения $T_\varepsilon \approx 4.32$, и, таким образом, это значение близко к минимальной стоимости исходной задачи. Данные в табл. 1.1 позволяют заключить, что оптимальное решение регуляризованной задачи (1.39) для $\varepsilon = 0.01$ достаточно близко к предельному, соответствующему $\varepsilon = 0$ и, следовательно, можно рассматривать как разумное приближение к оптимальному решению задачи (1.1).

Оптимальные траектории в плоскости (x_1, x_2) показаны на рис. 1.1 для $\varepsilon = 1$, 0.1 и 0.01.

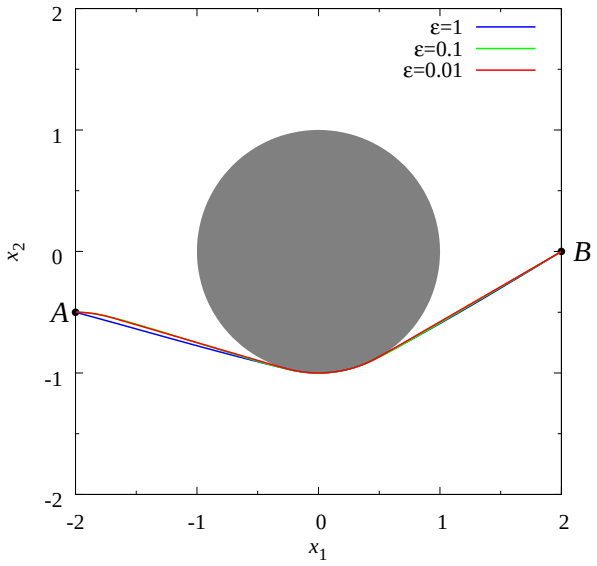

Рис. 1.1: Траектории в плоскости (x_1, x_2), соответствующие оптимальному решению задачи (1.39) для $\varepsilon = 1$ (синий), 0,1 (зеленый) и 0,01 (красный).

Для $\varepsilon = 0.01$ координаты (x_1, x_2) и α представлены как функция времени на рис. 1.2.

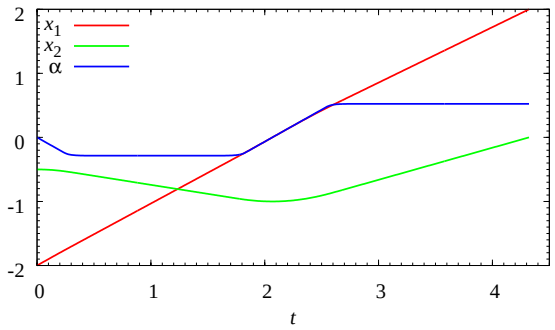

Рис. 1.2: Эволюция переменных $x_1(t)$ (красный), $x_2(t)$ (зеленый) и $\alpha(t)$ (синий) соответствует оптимальному решению задачи (1.39) для $\varepsilon = 0.01$.

Соответствующие управляющие функции u и p, а также v_1 и v_2 показаны на рис. 1.3.

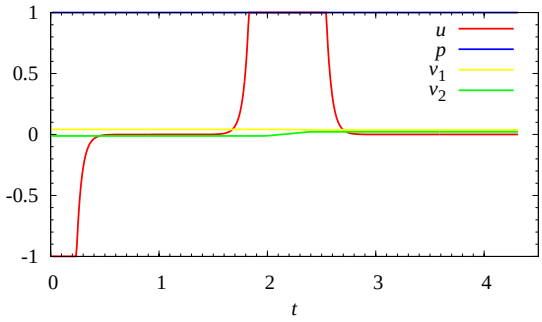

Рис. 1.3: Эволюция функций управления $u(t)$ (красный), $p(t)$ (синий), $v_1(t)$ (желтый) и $v_2(t)$ (зеленый) соответствует оптимальному решению задачи (1.39) для $\varepsilon = 0.01$.

Для проведения следующей группы экспериментов программа, реализующая указанный алгоритм, численно решила задачу оптимального управления (1.39) для следующих исходных данных:

$$x(0) = A = (-1.5,\ 0.3),$$
$$x(T) = B = (1.5, -0.5),$$
$$\alpha(0) = \alpha_0 = -\pi/4.$$

Расположение начальной и конечной точек маршрута робота, а также препятствие показаны на Рис. 1.4(а).

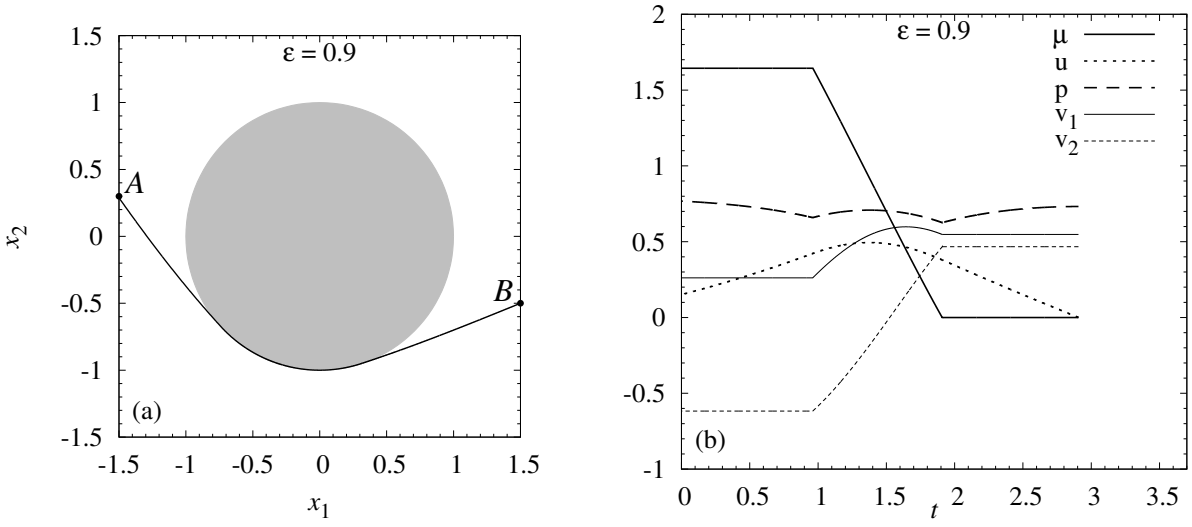

Рис. 1.4: Вычисленное решение регуляризованной задачи (1.39) при $\varepsilon = 0.9$. Показаны оптимальная траектория робота (a) и эволюция вдоль этой траектории меры-множителя Лагранжа $\mu(t)$ и управлений $u(t), p(t), v_1(t)$ и $v_2(t)$ (b). Препятствие показано на (a) сплошным серым цветом.

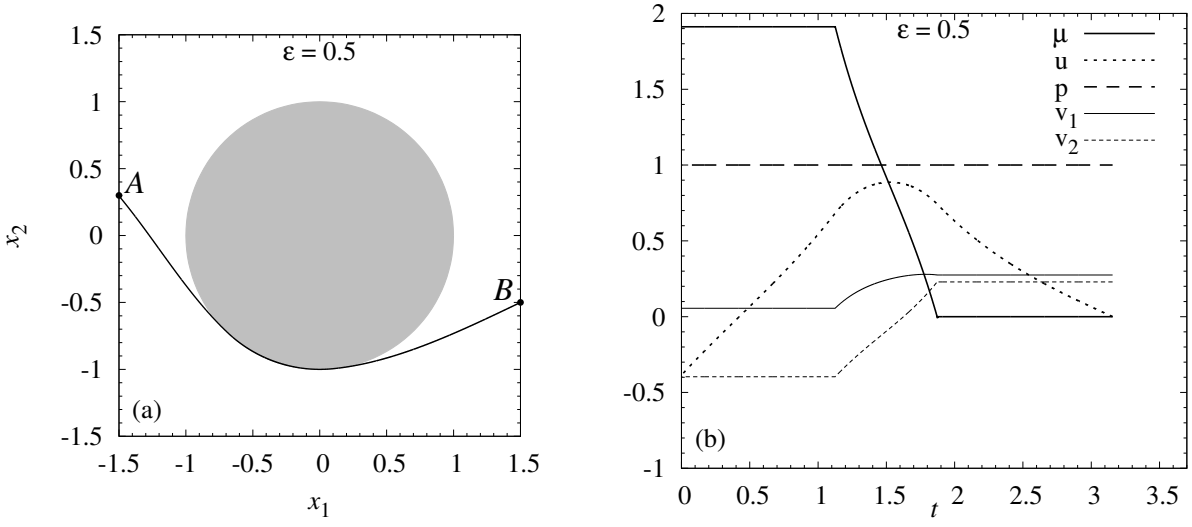

Рис. 1.5: То же что на Рис. 1.4 при $\varepsilon = 0.5$.

Шаг интегрирования системы дифференциальных уравнений брали равным 10^{-3}. Если траектория пересекала границу фазового ограничения (1.39e), точку

пересечения находили линейной интерполяцией, в этой точке вычисляли μ и считали ее непрерывной (т.е. в этой точке происходит заход на границу) если абсолютная величина этого значения не превосходила 10^{-2}. Решение краевой задачи считалось построенным, если расстояние от траектории до начального условия (т.е. в (1.39c) при $t = 0$) не превосходило 10^{-2}. Построение траектории останавливалось, если время следования по траектории превосходило 20 единиц времени.

Хотя, как описано выше, решение задачи строилось в обратном времени, то есть от $\widetilde{t} = 0$ до $\widetilde{t} = -T$, для построения рисунков все функции преобразованы и показаны в прямом времени, $t = \widetilde{t} + T$. Чтобы упростить сравнение, для всех значений ε на вертикальных и горизонтальных осях всех рисунков, приведенных ниже, показаны одинаковые интервалы значений.

Были проведены расчеты для $\varepsilon = 0.9$, 0.5, 0.2, 0.1 и 0.05, вычисленные оптимальные траектории и соответствующие управления показаны на Рис. 1.4–1.8 соответственно.

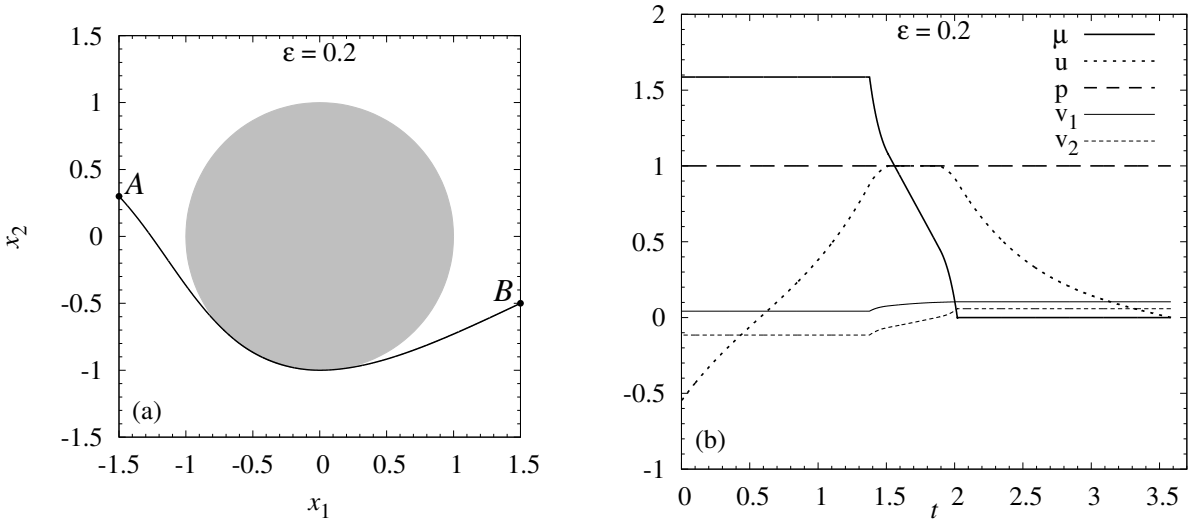

Рис. 1.6: То же что на Рис. 1.4 при $\varepsilon = 0.2$.

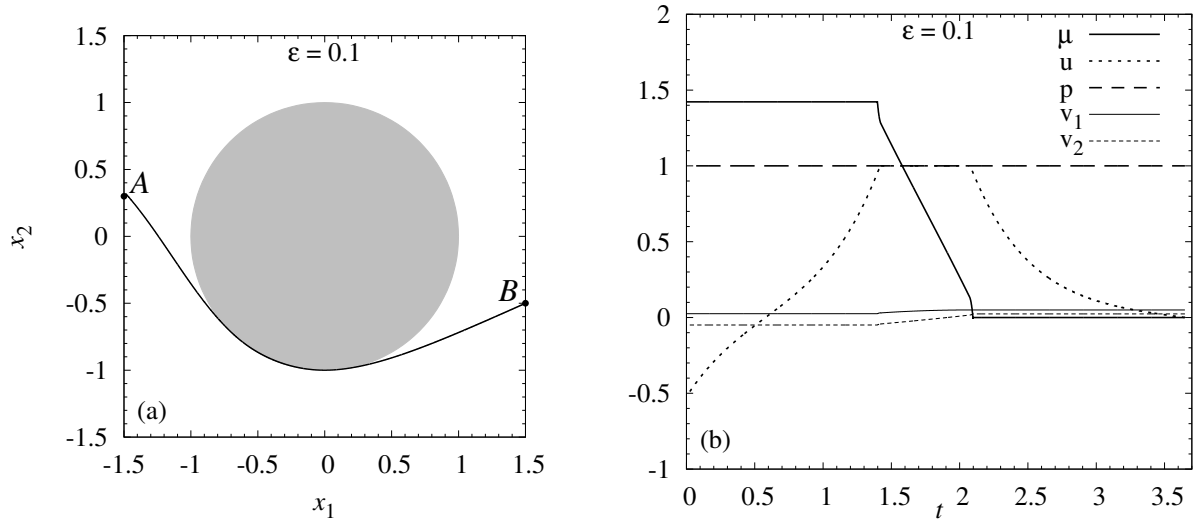

Рис. 1.7: То же что на Рис. 1.4 при $\varepsilon = 0.1$.

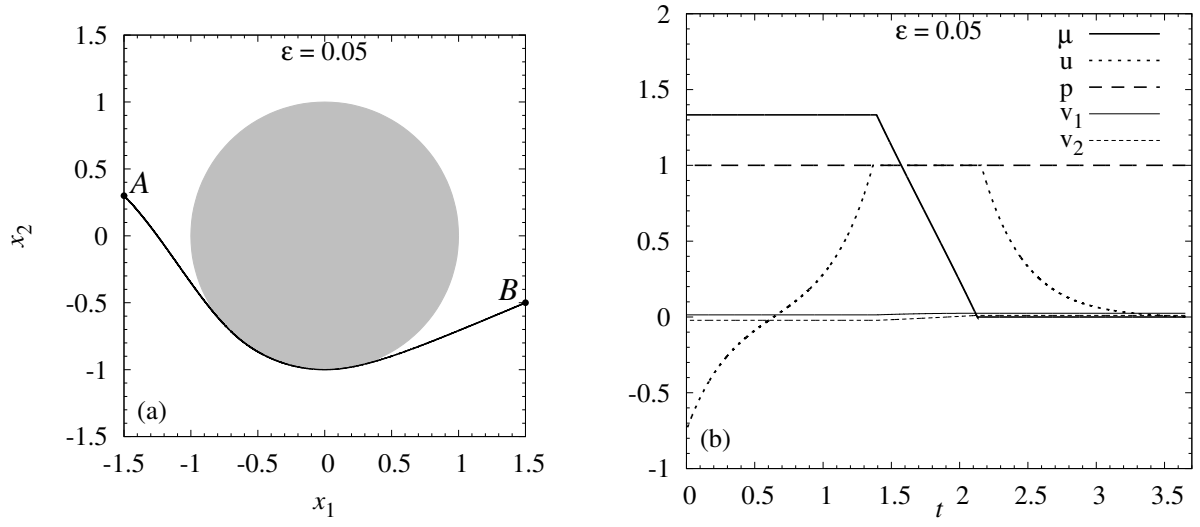

Рис. 1.8: То же что на Рис. 1.4 при $\varepsilon = 0.05$.

На первый взгляд, траектории робота, отвечающие решениям задачи оптимального управления при разных значениях ε, отличаются друг от друга мало (ср. Рис. 1.4-1.8 (a)). Каждая из этих траекторий состоит из трех частей – части заходящей на границу фазового ограничения, движению по границе и части,

сходящей с границы. Однако, режимы управления, обеспечивающее такое движение, различаются существенно. Так, при $\varepsilon = 0.9$ ни скорость поворота робота, $u(t)$, ни его продольная скорость, $p(t)$, не являются максимальными ни в какой точке траектории (Рис. 1.4(б)). При $\varepsilon = 0.5$ (Рис. 1.5(б)) скорость поворота снова нигде не достигает максимального значения, но его продольная скорость становится максимальной на всем участке движения, т.е. $p(t) = 1\ \forall t \in [0, T]$. При $\varepsilon = 0.2$ (см. Рис. 1.6(б)) продольная скорость остается всегда максимальной, а скорость поворота достигает максимального значения на некотором участке движения вдоль границы. При $\varepsilon = 0.1$ и 0.05 качественного изменения в сравнении с предыдущим случаем нет, но скорость поворота достигает максимального значения на большем участке границы (Рис. 1.7(б) и Рис. 1.8(б)), чем для траектории при $\varepsilon = 0.2$.

Время следования по оптимальным траекториям, T_ε, растет с уменьшением ε: $T_{0.9} = 2.91$, $T_{0.5} = 3.16$, $T_{0.2} = 3.58$, $T_{0.1} = 3.64$ и $T_{0.05} = 3.65$. Мы не рассматривали меньших значений ε, поскольку разница между двумя последними, $T_{0.05}$ и $T_{0.1}$, мала – порядка 10^{-2}. Кроме этого, интересно отметить поведение регуляризаторов, т.е. функций v_1 и v_2 (тонкие линии на Рис. 1.4-1.8 (b)). Из рисунков видно, что они стремятся к нулю в равномерной метрике при $\varepsilon \to 0$, причем с линейной относительно ε скоростью, что также свидетельствует о сходимости метода.

Также следует отметить характер изменения μ при уменьшении ε (при движении по границе, так как при движении вне границы величина меры-множителя Лагранжа не меняется). При $\varepsilon = 0.9$ (Рис. 1.4(б)) между точкой входа и схода с границы $\mu(t)$ – практически линейная функция. При $\varepsilon = 0.5$ и 0.2 (Рис. 1.5(б) и 1.6(б)) заметно характерное нелинейное поведение – вблизи точки захода и схода с границы функция быстро меняется оставаясь близкой к линейной при удалении от этих точек. Такая нелинейность (рост градиента) особенно заметна при $\varepsilon = 0.1$ (Рис. 1.7(б)). Однако, при $\varepsilon = 0.05$ линейность уже полностью восстанавливается. Это означает, что предельная мера-множитель Лагранжа, которая отвечает принципу максимума в исходной задаче (1.1), по всей вероятности непрерывна. В этой связи нужно отметить, что в известной литературе существует совсем немного примеров, которые демонстрируют наличие сингулярной составляющей меры, и в частности, атомов, см. например, [59, 79]. Представляется любопытным, что в такой существенно нерегулярной задаче как движение мобильного робота относительно препятствия мера-множитель все-же сохраняет свойство непрерывности. Заметим, что такое свойство не гарантируется никакой существующей теорией и установлено пока лишь эмпирически.

1.6 Другие примеры применения метода регуляризации к модели мобильного робота

Как было сказано раньше существует множество различных подходов к регуляризации. В данном разделе рассмотрим еще два способа, наиболее очевидных в контексте задач управления мобильным роботом.

1.6.1 Применение метода регуляризации к гусеничному мобильному роботу

Рассмотрим следующую постановку задачи:

$$\begin{cases} J(p,u,v,T) := T + \int_0^T |v|^2 dt \to \min, \\ \dot{x}_1 = p \cdot \cos\alpha + \varepsilon v_1, \\ \dot{x}_2 = p \cdot \sin\alpha + \varepsilon v_2, \\ \dot{\alpha} = u, \\ x(0) = A, \\ x(T) = B, \\ \alpha(0) = \alpha_0, \\ p \in [-1,1], \\ u \in [-1,1], \\ x_1^2 + x_2^2 \geq 1. \end{cases} \tag{1.40}$$

Здесь $v = (v_1, v_2) \in \mathbb{R}^2$ является дополнительной переменной управления. Заметим, что рассмотрение переменной v_3 при α не требуется, так как градиент фазовых ограничений относительно α исчезает.

Из этой задачи имеем

$$\Gamma(x,\alpha,p,u,v) = -x_1(p_1 \cdot \cos\alpha + \varepsilon v_1) - x_2(p_1 \cdot \sin\alpha + \varepsilon v_2),$$

и, следовательно, закономерность выполняется для задачи (1.40) для некоторого $\varepsilon > 0$, см. Предложение 3.

Таким образом, к задаче (1.40) может быть применен принцип максимума из теоремы 1 и обеспечена непрерывность ε-множителя. Анализ условия максимума относительно переменных v_1, v_2 дает формулы для регуляризаторов:

$$v_{\varepsilon 1}(t) = \frac{\varepsilon}{2\lambda}\Big(\psi_1(t) + \mu(t)x_{\varepsilon 1}(t))\Big),$$

$$v_{\varepsilon 2}(t) = \frac{\varepsilon}{2\lambda}\Big(\psi_2(t) + \mu(t)x_{\varepsilon 2}(t))\Big),$$

которые используются при сведении условий принципа максимума к соответствующей краевой задаче. Условие регулярности хорошо помогает выражать μ через функции состояния и совместного состояния, имеющие u, p, уже выраженные в силу условия максимума.[3] Соответствующая краевая задача может быть решена тем же методом выстрела, что и в [65–67].

1.6.2 Применение метода регуляризации к одноколесному велосипеду

Взяв число $\varepsilon > 0$, мы рассмотрим следующее возмущение для задачи (1.1).

$$\begin{aligned}
&J(p,u,v,T) := T + \varepsilon\int_0^T |u|^2dt + \int_0^T |v|^2dt \to \min,\\
&\dot{x}_1 = p\cdot\cos\alpha + \vartheta_1(x) + \varepsilon v_1,\\
&\dot{x}_2 = p\cdot\sin\alpha + \vartheta_2(x) + \varepsilon v_2,\\
&\dot{p} = u_1 + u_2,\\
&\ddot{\alpha} = u_1 - u_2,\\
&x(0) = A,\ x(T) = B,\\
&p(0) = p_0,\ \alpha(0) = \alpha_0,\ \dot{\alpha}(0) = \beta_0,\\
&u = (u_1,u_2) \in [-1,1]\times[-1,1],\\
&x_1^2 + x_2^2 \geq 1.
\end{aligned} \tag{1.41}$$

Здесь $v = (v_1, v_2)$ - это дополнительная двумерная управляющая переменная, отвечающая за регуляризацию. (Следует пояснить, что переменные v_3, v_4 и v_5

[3] Важно отметить, что применение принципа максимума к (1.40) осложняется наличием сингулярных режимов управления относительно p, u, и поэтому условие максимума может не работать для указанной цели. Единичные элементы управления представляют собой задачу, которая имеет совершенно иное происхождение по сравнению с масштабом данного конкретного исследования. Однако задача сингулярных управлений может быть решена аналогично путем добавления малых членов в минимизирующий функционал:

$$J(p,u,v,T) := T + \int_0^T |v|^2dt + \varepsilon\int_0^T |p|^2dt + \varepsilon\int_0^T |u|^2dt,$$

или любым другим известным способом. В отношении таких методов, и особого управления в теории оптимального управления, см., например, книгу [19].

метода опущены, поскольку градиент ограничения состояния относительно p, α и $\dot{\alpha}$ одинаково равен нулю.)

Задача (1.41) имеет решение, которое будет обозначено подиндексом ε в дальнейшем. Несложно проверить, что предельный набор траекторий для $(x_\varepsilon, p_\varepsilon, \alpha_\varepsilon)$ как $\varepsilon \to 0$ содержится в наборе глобальных минимизаторов задачи (1.1), см., например., в [41]. В то же время условие R) выполняется в (1.41) для всех $\varepsilon > 0$.

Используя этот факт, применим теорему 1 к решению задачи (1.41). Получается

$$\begin{aligned}\Gamma(x,p,\alpha,u,v) &= -x_1(p\cdot\cos\alpha+\vartheta_1(x)+\varepsilon v_1)\\ &\quad -x_2(p\cdot\sin\alpha+\vartheta_2(x)+\varepsilon v_2);\end{aligned}$$

$$\begin{aligned}&\bar{H}(x,p,\alpha,u,v,\psi,\mu,\lambda)\\ &\quad=(\psi_1+\mu x_1)(p\cos\alpha+\vartheta_1(x)+\varepsilon v_1)\\ &\qquad+(\psi_2+\mu x_2)(p\sin\alpha+\vartheta_2(x)+\varepsilon v_2)\\ &\qquad+\psi_3(u_1+u_2)+\psi_4(u_1-u_2)\\ &\qquad-\varepsilon\lambda(u_1^2+u_2^2)-\lambda(v_1^2+v_2^2).\end{aligned}$$

Здесь сопряженная переменная, которая соответствует $\dot{\alpha}$, была опущена, поскольку она полностью исчезает из-за условия трансверсальности в конечной точке.

Обратите внимание, что случай $\lambda = 0$, как это легко проверить, исключен из-за замечания 1. Поэтому рассмотрим число $\lambda > 0$, абсолютно непрерывную векторнозначную функцию $\psi(t)$ и непрерывную и убывающую скалярнозначную функцию $\mu(t)$, которые удовлетворяют принципу максимума. Все эти множители зависят от ε. Эта зависимость опущена для простоты и ясности обозначения.

В силу принципа максимума получается

$$\begin{aligned}
\dot\psi_1(t) &= -\psi_1(t)\vartheta'_{1x_1}(x_\varepsilon(t)) - \psi_2(t)\vartheta'_{2x_1}(x_\varepsilon(t)) \\
&\quad -\mu(t)\Big(p_\varepsilon(t)\cos\alpha_\varepsilon(t) + \vartheta_1(x_\varepsilon(t)) + \varepsilon v_{\varepsilon 1}(t) \\
&\quad +x_{\varepsilon 1}(t)\vartheta'_{1x_1}(x_\varepsilon(t)) + x_{\varepsilon 2}(t)\vartheta'_{2x_1}(x_\varepsilon(t))\Big), \\
\dot\psi_2(t) &= -\psi_1(t)\vartheta'_{1x_2}(x_\varepsilon(t)) - \psi_2(t)\vartheta'_{2x_2}(x_\varepsilon(t)) \\
&\quad -\mu(t)\Big(p_\varepsilon(t)\sin\alpha_\varepsilon(t) + \vartheta_2(x_\varepsilon(t)) + \varepsilon v_{\varepsilon 2}(t) \\
&\quad +x_{\varepsilon 1}(t)\vartheta'_{1x_2}(x_\varepsilon(t)) + x_{\varepsilon 2}(t)\vartheta'_{2x_2}(x_\varepsilon(t))\Big), \qquad (1.42) \\
\dot\psi_3(t) &= -\cos\alpha_\varepsilon(t)(\psi_1(t) + \mu(t)x_{\varepsilon 1}(t)) \\
&\quad -\sin\alpha_\varepsilon(t)(\psi_2(t) + \mu(t)x_{\varepsilon 2}(t)), \\
\dot\psi_4(t) &= p_\varepsilon(t)\sin\alpha_\varepsilon(t)(\psi_1(t) + \mu(t)x_{\varepsilon 1}(t)) \\
&\quad -p_\varepsilon(t)\cos\alpha_\varepsilon(t)(\psi_2(t) + \mu(t)x_{\varepsilon 2}(t)).
\end{aligned}$$

В то же время, $\psi_3(T_\varepsilon) = \psi_4(T_\varepsilon) = 0$.

Максимальное условие по отношению к переменным v_1, v_2 дает:

$$v_{\varepsilon 1}(t) = \frac{\varepsilon}{2\lambda}\Big(\psi_1(t) + \mu(t)x_{\varepsilon 1}(t)\Big),$$

$$v_{\varepsilon 2}(t) = \frac{\varepsilon}{2\lambda}\Big(\psi_2(t) + \mu(t)x_{\varepsilon 2}(t)\Big).$$

Условие максимума по отношению к переменным u_1 и u_2 приводит к тому, что $u_\varepsilon(t)$ является максимизатором в следующей квадратичной задаче:

$$(u_1 + u_2)\psi_3(t) + (u_1 - u_2)\psi_4(t) - \varepsilon\lambda(u_1^2 + u_2^2) \to \max,$$

когда $-1 \le u_1, u_2 \le 1$.

Следовательно,

$$u_{\varepsilon 1} = \operatorname{sgn}\Big(\psi_3(t) + \psi_4(t)\Big)\min\Big\{\frac{|\psi_3(t) + \psi_4(t)|}{2\varepsilon\lambda}, 1\Big\},$$

$$u_{\varepsilon 2} = \operatorname{sgn}\Big(\psi_3(t) - \psi_4(t)\Big)\min\Big\{\frac{|\psi_3(t) - \psi_4(t)|}{2\varepsilon\lambda}, 1\Big\}.$$

Напомним, что $|\psi_1(T_\varepsilon)| + |\psi_2(T_\varepsilon)| > 0$. Действительно, в противном случае, принимая во внимание, что $\mu(T_\varepsilon) = 0$ и $\psi_3(T_\varepsilon) = \psi_4(T_\varepsilon) = 0$, из условия трансверсальности во времени следует, что $\lambda = 0$. Таким образом, после нормализации имеем

$$|\psi_1(T_\varepsilon)| + |\psi_2(T_\varepsilon)| = 1. \tag{1.43}$$

Далее вычисляем множитель λ. Используя условие трансверсальности во времени (зависимость от t будет опущена для простоты изложения)

$$(\psi_1 + \mu x_1)(p\cos\alpha + \vartheta_1(x) + \varepsilon v_1) + (\psi_2 + \mu x_2)(p\sin\alpha + \vartheta_2(x) + \varepsilon v_2) + \\ +\psi_3(u_1 + u_2) + \psi_4(u_1 - u_2) - \varepsilon\lambda(u_1^2 + u_2^2) - \lambda(v_1^2 + v_2^2) = \lambda.$$

учитывая, что $u_1^2 + u_2^2 = 0$, $\mu(T_\varepsilon) = 0$ и $\psi_3(T_\varepsilon) = \psi_4(T_\varepsilon) = 0$, получаем

$$\psi_1(T_\varepsilon)\Big(p(T_\varepsilon)\cos\alpha_\varepsilon(T_\varepsilon) + \vartheta_1(b_1) + \varepsilon v_1(T_\varepsilon)\Big) + \\ +\psi_2(T_\varepsilon)\Big(p(T_\varepsilon)\sin\alpha_\varepsilon(T_\varepsilon) + \vartheta_2(b_2) + \varepsilon v_2(T_\varepsilon)\Big) - \\ -\lambda(v_1^2(T_\varepsilon) + v_2^2(T_\varepsilon)) = \lambda.$$

Подставив выражения для $v_{\varepsilon 1}$ и $v_{\varepsilon 2}$, получим

$$\psi_1(T_\varepsilon)\Big(p(T_\varepsilon)\cos\alpha_\varepsilon(T_\varepsilon) + \vartheta_1(b_1)\Big) + \psi_2(T_\varepsilon)\Big(p(T_\varepsilon)\sin\alpha_\varepsilon(T_\varepsilon) + \vartheta_2(b_2)\Big) + \\ +\varepsilon\Big(\psi_1(T_\varepsilon)\frac{\varepsilon}{2\lambda}\psi_1(T_\varepsilon) + \psi_2(T_\varepsilon)\frac{\varepsilon}{2\lambda}\psi_2(T_\varepsilon)\Big) = \lambda + \frac{\varepsilon^2\psi_1^2(T_\varepsilon)}{4\lambda} + \frac{\varepsilon^2\psi_2^2(T_\varepsilon)}{4\lambda},$$

или

$$\psi_1(T_\varepsilon)\Big(p(T_\varepsilon)\cos\alpha_\varepsilon(T_\varepsilon) + \vartheta_1(b_1)\Big) + \psi_2(T_\varepsilon)\Big(p(T_\varepsilon)\sin\alpha_\varepsilon(T_\varepsilon) + \vartheta_2(b_2)\Big) + \\ +\frac{\varepsilon^2}{2\lambda}\Big(\psi_1^2(T_\varepsilon) + \psi_2^2(T_\varepsilon)\Big) = \lambda + \frac{\varepsilon^2}{4\lambda}\Big(\psi_1^2(T_\varepsilon) + \psi_2^2(T_\varepsilon)\Big),$$

Учитывая (1.43), и обозначив

$$\omega_\varepsilon \;=\; \psi_1(T_\varepsilon)\Big(p_\varepsilon(T_\varepsilon)\cos\alpha_\varepsilon(T_\varepsilon) + \vartheta_1(b_1)\Big) + \psi_2(T_\varepsilon)\Big(p_\varepsilon(T_\varepsilon)\sin\alpha_\varepsilon(T_\varepsilon) + \vartheta_2(b_2)\Big),$$

получаем квадратное уравнение

$$\omega_\varepsilon + \frac{\varepsilon^2}{4\lambda} - \lambda = 0, \quad \text{или} \quad 4\lambda^2 - 4\omega_\varepsilon\lambda - \varepsilon^2 = 0,$$

решая которое относительно λ, получаем

$$\lambda = \frac{\omega_\varepsilon \pm \sqrt{\omega_\varepsilon^2 + \varepsilon^2}}{2},$$

и отбросив отрицательный корень, приходим к формуле

$$\lambda = \frac{\omega_\varepsilon + \sqrt{\omega_\varepsilon^2 + \varepsilon^2}}{2}.$$

Теперь вычислим μ. На границе ограничения состояния имеем $\Gamma(x_\varepsilon, p_\varepsilon, \alpha_\varepsilon, u_\varepsilon, v_\varepsilon) = 0$. Следовательно,

$$\begin{aligned} x_{\varepsilon 1}\Big(p_\varepsilon \cos\alpha_\varepsilon + \vartheta_1(x_\varepsilon) + \varepsilon v_{\varepsilon 1}\Big) & \\ + x_{\varepsilon 2}\Big(p_\varepsilon \sin\alpha_\varepsilon + \vartheta_2(x_\varepsilon) + \varepsilon v_{\varepsilon 2}\Big) &= 0. \end{aligned}$$

Зависимость от времени здесь и далее опущена для удобства обозначения. Подставляя в это равенство выражения, полученные выше для v_ε, получаем

$$\begin{aligned} x_{\varepsilon 1}\Big(p_\varepsilon \cos\alpha_\varepsilon + \vartheta_1(x_\varepsilon) + \varepsilon\frac{\varepsilon}{2\lambda}(\psi_1 + \mu x_{\varepsilon 1})\Big) & + \\ + x_{\varepsilon 2}\Big(p_\varepsilon \sin\alpha_\varepsilon + \vartheta_2(x_\varepsilon) + \varepsilon\frac{\varepsilon}{2\lambda}(\psi_2 + \mu x_{\varepsilon 2})\Big) &= 0, \end{aligned}$$

или

$$\begin{aligned} x_{\varepsilon 1}p_\varepsilon \cos\alpha_\varepsilon + x_{\varepsilon 1}\vartheta_1(x_\varepsilon) + \frac{\varepsilon^2}{2\lambda}\psi_1 x_{\varepsilon 1} + \frac{\varepsilon^2}{2\lambda}\mu x_{\varepsilon 1}^2 + & \\ + x_{\varepsilon 2}p_\varepsilon \sin\alpha_\varepsilon + x_{\varepsilon 2}\vartheta_2(x_\varepsilon) + \frac{\varepsilon^2}{2\lambda}\psi_2 x_{\varepsilon 2} + \frac{\varepsilon^2}{2\lambda}\mu x_{\varepsilon 2}^2 &= 0, \end{aligned}$$

откуда

$$\begin{aligned} \mu\Big(\frac{\varepsilon^2}{2\lambda}x_{\varepsilon 1}^2 + \frac{\varepsilon^2}{2\lambda}x_{\varepsilon 2}^2\Big) = -x_{\varepsilon 1}p_\varepsilon \cos\alpha_\varepsilon - x_{\varepsilon 2}p_\varepsilon \sin\alpha_\varepsilon - x_{\varepsilon 1}\vartheta_1(x_\varepsilon) - & \\ - x_{\varepsilon 2}\vartheta_2(x_\varepsilon) - \frac{\varepsilon^2}{2\lambda}\psi_1 x_{\varepsilon 1} - \frac{\varepsilon^2}{2\lambda}\psi_2 x_{\varepsilon 2}, & \end{aligned}$$

или

$$\mu = \frac{2\lambda}{\varepsilon^2}\Big(\frac{-x_{\varepsilon 1}(p_\varepsilon \cos\alpha_\varepsilon + \vartheta_1(x_\varepsilon)) - x_{\varepsilon 2}(p_\varepsilon \sin\alpha_\varepsilon + \vartheta_2(x_\varepsilon)) - \frac{\varepsilon^2}{2\lambda}\psi_1 x_{\varepsilon 1} - \frac{\varepsilon^2}{2\lambda}\psi_2 x_{\varepsilon 2}}{x_{\varepsilon 1}^2 + x_{\varepsilon 2}^2}\Big).$$

Так как μ вычисляется на границе фазового ограничения, то $x_{\varepsilon 1}^2 + x_{\varepsilon 2}^2 = 1$, и окончательно получаем

$$\mu = -x_{\varepsilon 1}\psi_1 - x_{\varepsilon 2}\psi_2 - \frac{2\lambda}{\varepsilon^2}\Big(x_{\varepsilon 1}(p_\varepsilon \cos\alpha_\varepsilon + \vartheta_1(x_\varepsilon)) + x_{\varepsilon 2}(p_\varepsilon \sin\alpha_\varepsilon + \vartheta_2(x_\varepsilon))\Big).$$

На тех интервалах, где ограничение состояния неактивно, функция μ является постоянной из-за замечания 2. Кроме того, μ является непрерывным, что необходимо для того, чтобы найти точки пересечения (то есть точки, в которых экстремальная траектория пересекается с границей или покидает ее). Полученное значение для μ подставляется в выражения для v_ε и в сопряженное уравнение. Таким образом, условия принципа максимума сводятся к решению краевой задачи.

Таким образом, как показывает приведенная выше формула, μ может стремиться к бесконечности при $\varepsilon \to 0$. В то же время, из представленных расчетов несложно вывести следующую оценку. Если

$$|\psi_1(T_\varepsilon)| + |\psi_2(T_\varepsilon)| = \varepsilon^2,$$

тогда μ ограничено на всем временном интервале движения константой, которая не зависит от ε.

1.7 Сравнение с методами машинного обучения

Теория оптимального управления, и в частности принцип максимума Понтрягина, играет важную роль в современной робототехнике, особенно при проектировании систем управления для мобильных роботов. Такие роботы – от наземных платформ (колёсных и шагающих) до дронов и подводных аппаратов – должны перемещаться в сложной среде, минимизируя время, энергопотребление или износ, при соблюдении физических и геометрических ограничений.

Одна из ключевых задач в управлении мобильными роботами – планирование оптимальной траектории от начальной до целевой точки. Часто требуется минимизировать время движения (задача быстродействия), энергопотребление (например, заряд батареи), износ механизмов (например, избегание резких ускорений).

В данной работе рассматривается задача быстродействия.

Принцип максимума Понтрягина позволяет получить **аналитические или полуаналитические решения** таких задач, особенно в случае гладких динамических моделей. Например, для робота с ограниченным ускорением и скоростью, задача о минимальном времени перехода может быть решена с помощью анализа структуры оптимального управления (релейного или непрерывного), предсказанной принципом максимума.

Мобильные роботы работают в реальных условиях, где

- запрещены выход за пределы коридора, комнаты или тротуара (геометрические ограничения),
- необходимо избегать препятствий (ограничения вида $\|x(t) - x_{\text{преп}}\| \geq r$),
- ограничена максимальная скорость вблизи людей или хрупких объектов.

Все эти требования – фазовые ограничения, и именно здесь расширенная версия принципа максимума с учётом таких ограничений становится незаменимой. Она позволяет не только найти оптимальное управление, но и понять, как робот должен «касаться» границы допустимой зоны (например, скользить вдоль стены), и как при этом меняются сопряжённые переменные (аналоги импульсов в механике).

Хотя в реальных системах часто используются численные методы (модель прогностического управления [176], алгоритм быстрорастущего случайного дерева [177], градиентные методы [178]), принцип максимума

– служит эталоном для проверки оптимальности численных решений;

– помогает в снижении размерности задачи;

– используется в онлайновом управлении (например, в motion primitives);

– лежит в основе оптимальных фильтров и оценки состояния.

Таким образом, принцип максимума Понтрягина остается не только теоретическим инструментом, но и практически значимым подходом в разработке интеллектуальных систем управления для мобильных роботов.

Сравнение с методами машинного обучения: обучение с подкреплением С развитием вычислительных технологий и машинного обучения в робототехнике все шире применяются методы обучения с подкреплением (Reinforcement Learning, RL), которые позволяют роботам «научиться» оптимальному поведению через взаимодействие со средой. Однако классические методы оптимального управления, такие как принцип максимума Понтрягина, по-прежнему остаются актуальными. Проведём сравнение этих двух подходов по ключевым критериям: наличие математической модели мобильного робота, теоретическая обоснованность метода, вычислительная сложность, учет фазовых ограничений, гарантии оптимальности и возможность применять в режиме реального времени.

В таблице 1.2 приведены результаты сравнения методов по перечисленным параметрам.

Таблица 1.2: Сравнение принципа максимума и обучения с подкреплением

Критерий	**Принцип максимума**	**Обучение с подкреплением**
Модель системы	Требуется точная модель	Может работать без модели (model-free)
Теоретическая обоснованность	Да (необходимые условия оптимальности)	Оценка сходимости сложна
Вычислительная сложность	Аналитические или полуаналитические решения	Требует большого числа симуляций
Обобщение	Хорошее при известной структуре	Зависит от качества обучения
Фазовые ограничения	Учитываются строго (через меры и условия комплементарности)	Учитываются через штрафы в награде
Гарантии оптимальности	Да (в пределах гладкости и регулярности)	Асимптотические (при сходимости)
Онлайн-применимость	Высокая (после анализа)	Требует времени на обучение

По результатам сравнения можно сделать следующие заключения. К достоинствам принципа максимума относятся:

– гарантированная оптимальность – при выполнении условий регулярности принцип максимума даёт *необходимые условия* оптимальности, это позволяет доказать, что найденная траектория – действительно оптимальна;

– эффективность – после аналитического решения задача сводится к расчёту

траектории по явным формулам, что критично для бортовых систем с ограниченными ресурсами;

– интерпретируемость – структура оптимального управления (например, «разгон — движение — торможение») понятна и может быть адаптирована;

– точное обращение с ограничениями – фазовые ограничения встраиваются в теорию строго, а не через штрафы, что исключает нарушения границ.

К достоинствам обучения с подкреплением относятся:

– гибкость – RL может обучаться в сложных, плохо формализованных средах, где нет точной модели динамики;

– адаптивность – RL-агент может адаптироваться к изменяющейся среде (например, новым препятствиям, изменению трения);

– масштабируемость – современные алгоритмы (PPO, SAC, DDPG) позволяют решать задачи высокой размерности.

– Обучение на реальных данных – не требует априорной модели – обучается «в реальном времени».

На практике все чаще используются гибридные подходы, сочетающие сильные стороны обоих методов:

– информированное RL – использование решений, полученных по принципу максимума, как *демонстраций* для обучения агента (Imitation Learning);

– Model-based RL – построение приближенной модели системы, после чего к ней применяются методы, близкие к принципу максимума;

– Guided exploration – использование аналитических решений для ускорения сходимости RL;

– Verification – проверка RL-политики на соответствие необходимым условиям оптимальности.

Например, в задачах управления дронами часто сначала строят оптимальную траекторию с помощью принципа максимума (с учётом ограничений по тяге и углам), а затем используют RL для компенсации возмущений (ветер, дрейф датчиков).

В главе 4 данной работы принцип максимума Понтрягина используется для построения оптимальной траектории, которая используется как эталонная траектория для последующего обучения мобильного робота.

Таким образом, можно сделать вывод, что прининцип максимума Понтрягина и обучение с подкреплением — не конкуренты, а дополняющие друг друга парадигмы. Первый обеспечивает гарантии, интерпретируемость и эффективность, второй – адаптивность, гибкость и способность к обучению. В современ-

ной робототехнике будущее – за их интеграцией: использование аналитических методов как «ядро» управления, а RL – как слой адаптации и устойчивости к неопределённостям.

1.8 Заключение

В диссертационной работе предложен метод возмущений для решения нерегулярных задач оптимального управления с фазовым ограничением. В качестве критерия оптимальности рассмотрено быстродействие. Такая тема представляется актуальной в связи с рядом прикладных задач, в частности, в робототехнике. В работе обсужден подход, основанный на принципе максимума. Отмечено, что прямое применение принципа максимума не приводит к нужному результату, что обусловлено существенной нерегулярностью задачи относительно фазовых ограничений. Поэтому был разработан метод регуляризации, чтобы преодолеть такое явление, вызванное нерегулярностью. Предложено построение аппроксимирующей последовательности регулярных ε-возмущенных задач. Отсюда следует, что возмущенная задача, или ε-задача, уже является регулярной относительно фазовых ограничений и численно разрешима в рамках косвенного подхода, то есть с использованием принципа максимума, благодаря непрерывности меры-множителя, которая обеспечивается при условии регулярности. Этот факт обосновывает необходимость анализа возмущений, проделанного в данной работе. Была разработана схема регуляризации и предоставлены некоторые необходимые теоретические инструменты для дальнейшей численной реализации. Численная процедура, построенная с помощью непрямого подхода, опирается на непрерывность множителей, которая имеет место в регулярных задачах.

В данной главе диссертационной работы изучена задача управления мобильным роботом (например, движение трехколесного мобильного робота, или одноколесного велосипеда, или управление гусеничным мобильным роботом с инерционным рулевым управлением) при наличии фазового ограничения, заданного единичной окружностью. Отмечено, что прямое применение принципа максимума в модели мобильного робота не имеет смысла из-за отсутствия регулярности, а также из-за особого режима управления относительно угловой скорости. В связи с этим был предложен метод возмущений, целью которого является регуляризация исходной задачи. Этот факт позволяет применить непрямой численный метод, основанный на классе алгоритмов стрельбы.

Представленный метод был применен к тестовой задаче о безинерционном движении трехколесного мобильного робота с передним приводом. Приведены результаты численного эксперимента.

Глава 2

Непрямой подход на основе модифицированного принципа максимума Понтрягина при фазовых ограничениях глубины 2

В диссертационной работе предложен метод вычисления с заданной точностью точных экстремалей в задаче оптимального управления (задача быстродействия) движением уницикличной модели робота при наличии фазовых ограничений глубины 2. Для этого используется принцип максимума Понтрягина второго порядка в отношении ограничений состояния.

В точках выхода на границу и схода с нее учтены соответствующие оценки скачка производной множителя. Рассмотрен вопрос о непрерывности экстремального управления. Вычисление выполнено для простейшей типовой задачи, которая в тексте называется задачей о кратчайшей кривой 2-го порядка.

2.1 О фазовых ограничениях глубины 2

В данной работе рассматривается так называемая динамика управления ньютоновского типа, т.е. следующая система дифференциальных уравнений

$$\begin{cases} \dot{x}(t) = f_1(x(t); y(t)); \\ \dot{y}(t) = f_2(x(t); y(t); u(t)), \end{cases} \tag{2.1}$$

которая моделирует движение заданного объекта.

Здесь $x(t) \in R^n$ и $y(t) \in R^n$ – заданные траектории состояний; $u(t) \in R^m$ – функция управления. Простейший случай управления динамикой задается системой

$$\begin{cases} \dot{x}(t) = y(t); \\ \dot{y}(t) = u(t). \end{cases} \tag{2.2}$$

Это случай классического движения согласно закону Ньютона с управлением ускорением $u(t)$.

Рассмотрим временной интервал $[0; T]$, где конечный момент времени T не фиксирован и рассматривается как параметр. Возьмем две точки $A; B \in R^n$ и предположим, что $x(0) = A$ и $x(T) = B$. Очедидно предположить, что ускорение ограничено: $|u| \leq c$ для некоторого $c > 0$.

Какое время T минимально возможно для перемещения объекта из начальной точки A в конечную точку B, с начальным значением скорости равным нулю?

Это простейшая формулировка так называемой задачи оптимального управления с нефиксированными концами, для которой решение — это оптимальное ускорение $\bar{u}(t) = \frac{B-A}{|B-A|}$.

Решение системы дифференциальных уравнений (2.1) представляет собой не только классическую задачу теории управления, но и чрезвычайно актуальную проблему в современной науке и технике. Эта система описывает динамику объекта второго порядка, где управление осуществляется через ускорение, и находит широкое применение в робототехнике, автоматизированных системах и киберфизических устройствах.

Основные направления использования в робототехнике:

– планирование траекторий – в мобильной робототехнике (автономные роботы-уборщики, складские агенты) и роботах-манипуляторах задача сводится к построению гладких траекторий положения $x(t)$ и скорости $y(t)$ с учетом ограничений на ускорение $u(t)$, решение данной системы позволяет генерировать допустимые траектории, избегающие резких рывков и перегрузок;

– сглаживание движения – в системах позиционирования (например, 3D-принтеры, ЧПУ-станки) важно минимизировать вибрации и износ механизмов. Учет динамики $\dot{y} = u$ позволяет реализовать траектории с ограниченным ускорением и рывком (jerk), что достигается через оптимальное или кусочно-линейное управление $u(t)$;

– модельно-предиктивное управление – в современных роботах задача (2.1) используется как прогнозная модель для расчета оптимального управления в реальном времени, что позволяет роботам адаптивно реагировать на препятствия, изменяя скорость и ускорение с учётом ограничений;

– обучение с подкреплением и робастное управление – в исследованиях по автономному поведению роботов данная система служит упрощённой средой (например, задача `Pendulum`, `CartPole` в OpenAI Gym), где агент учится управлять объектом, минимизируя отклонение от целевого состояния, а знание аналитического поведения системы помогает интерпретировать и верифицировать обученные политики.

– стабилизация и слежение – при разработке регуляторов (ПИД, LQR, нелинейные наблюдатели) для роботов-колёсников или дронов в одномерных сценариях (например, подъём/спуск, движение по линии) используется линеаризованная или прямая форма данной системы для синтеза управляющих алгоритмов.

Таким образом, решение задачи (2.1) остается актуальным и востребованным как в фундаментальных исследованиях по теории управления, так и в прикладных разработках, особенно в быстро развивающейся области робототехники, где точность, безопасность и эффективность движения определяют успех всей системы.

Рассмотрим возможные методы решения задачи (2.2).

Прямое интегрирование. Пусть заданы начальные условия

$$x(0) = x_0, \quad y(0) = y_0.$$

Проинтегрируем уравнения. Из уравнения $\dot{y}(t) = u(t)$ получаем

$$y(t) = y_0 + \int_0^t u(s)\, ds.$$

Подставим в уравнение для $\dot{x}(t) = y(t)$

$$x(t) = x_0 + \int_0^t y(s)\, ds = x_0 + \int_0^t \left(y_0 + \int_0^s u(\tau)\, d\tau \right) ds.$$

После перестановки интегралов

$$x(t) = x_0 + y_0 t + \int_0^t \int_0^s u(\tau)\, d\tau\, ds.$$

Таким образом, решение зависит от вида управляющего сигнала $u(t)$.

Пусть $u(t) = a$ (постоянное ускорение). Тогда

$$y(t) = y_0 + at,$$
$$x(t) = x_0 + y_0 t + \frac{1}{2} a t^2.$$

Это классические уравнения кинематики для движения с постоянным ускорением.

Решение через матричную экспоненту. Систему (2.2) можно записать в векторной форме

$$\dot{\mathbf{z}}(t) = A\mathbf{z}(t) + Bu(t),$$

где $\mathbf{z}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$, $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

Решение однородного уравнения ($u = 0$)

$$\mathbf{z}(t) = e^{At}\mathbf{z}_0, \quad e^{At} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}.$$

Общее решение

$$\mathbf{z}(t) = e^{At}\mathbf{z}_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)\,d\tau.$$

Матрица управляемости определяется как

$$\mathcal{C} = [B \mid AB] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Ее ранг равен 2 (полный ранг), следовательно, система полностью управляема. Это означает, что при подходящем $u(t)$ можно перевести систему из любого начального состояния в любое конечное состояние за конечное время.

Применение принципа максимума Понтрягина. Рассмотрим задачу оптимального управления по быстродействию: перевести систему из начального состояния (x_0, y_0) в начало координат $(0, 0)$ за минимальное время T при ограничении $|u(t)| \le u_{\max}$.

Введём сопряжённые переменные $\psi_x(t)$, $\psi_y(t)$. Гамильтониан Понтрягина

$$H = \psi_x\dot{x} + \psi_y\dot{y} = \psi_x y + \psi_y u.$$

Сопряжённая система

$$\dot{\psi}_x = -\frac{\partial H}{\partial x} = 0, \quad \dot{\psi}_y = -\frac{\partial H}{\partial y} = -\psi_x \Rightarrow \psi_y(t) = \psi_y(0) - \psi_x t.$$

Интегрируя

$$\psi_x(t) = a = \text{const}, \quad \psi_y(t) = -at + b.$$

Управление максимизирует гамильтониан

$$u^*(t) = \arg\max_{|u|\le u_{\max}} H \quad \Rightarrow \quad u^*(t) = \text{sign}(\psi_y(t)) = \text{sign}(-at + b).$$

Оптимальная стратегия – релейная. Кривая переключения в фазовой плоскости (x, y) задаётся уравнением

$$x = -\frac{1}{2} y|y| = \begin{cases} -\frac{1}{2}y^2, & y \geq 0, \\ +\frac{1}{2}y^2, & y < 0. \end{cases}$$

Оптимальное управление

$$u^*(x, y) = -\operatorname{sign}\left(x + \frac{1}{2} y|y|\right).$$

По принципу максимума Понтрягина оптимальное по быстродействию управление кусочно-постоянно и имеет не более одного переключения между $\pm u_{\max}$ на кривой $x = -\frac{1}{2}y|y|$. Эта кривая является множеством переключения, и все траектории должны к ней «подруливать», чтобы достичь начала координат за минимальное время.

Минимизируя функционал

$$J = \int_0^T \frac{1}{2} u^2(t)\, dt \to \min,$$

используя условия Эйлера–Лагранжа или Понтрягина, получаем, что оптимальное $u(t)$ – гладкая функция (часто полиномиальная), и решение выражается через кубические или квартичные траектории.

На рис. 2.1 показаны кривая переключения (жирная синяя линия), примеры траекторий при $u = +1$ (зелёные параболы), примеры траекторий при $u = -1$ (красные параболы), оптимальная траектория (жирная чёрная линия) с переключением на кривой.

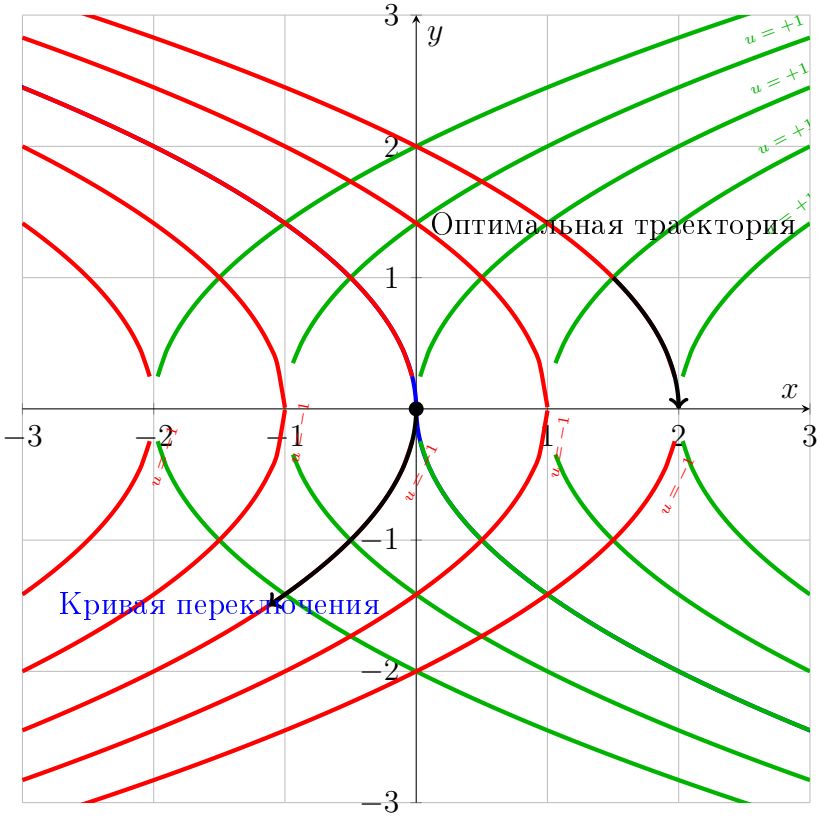

Рис. 2.1: Фазовый портрет и оптимальное управление

Добавим на фазовый портрет «концентрические окружности», задающие ограничения на начальные условия или зоны состояний: внутренняя окружность – $x^2 + y^2 = 1$ – зона малых возмущений, внешняя окружность – $x^2 + y^2 = 4$ – максимальная допустимая амплитуда состояния. Окружности $x^2 + y^2 = R^2$ могут интерпретироваться как области начальных условий (например, все начальные состояния лежат в круге радиуса 2), ограничения на энергию системы ($E = x^2 + y^2 \leq 4$), зоны безопасности (система не должна выходить за пределы $x^2 + y^2 \leq 4$), обратное время достижения (чем дальше от начала, тем больше времени нужно для стабилизации) и т.п.

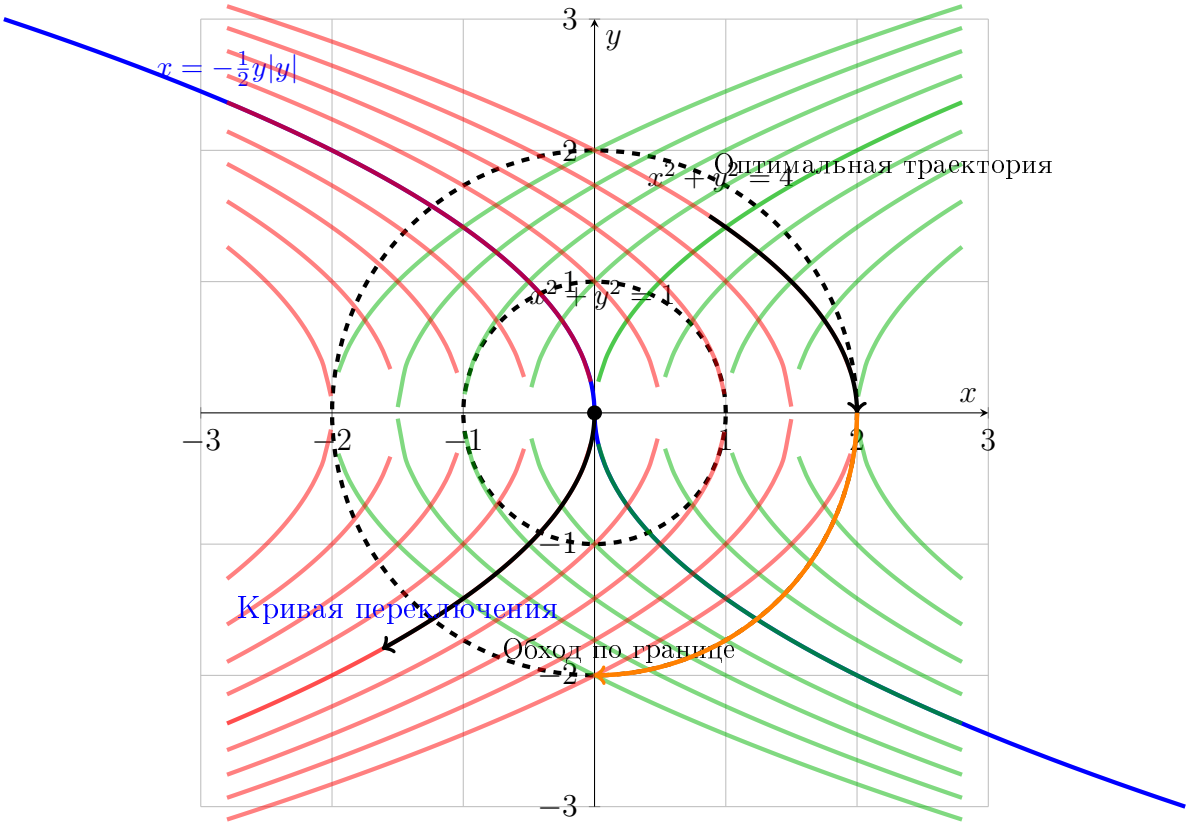

Рис. 2.2: Фазовый портрет с круговыми ограничениями и оптимальным управлением

Таким образом, оптимальное управление учитывает как динамику системы, так и геометрические ограничения. Кривая переключения и круговые границы помогают визуализировать допустимые траектории и оценить время регулирования.

Для решения поставленной задачи численными методами можно использовать **метод Эйлера**: заданном $u(t)$ система имеет вид

$$x_{k+1} = x_k + h \cdot y_k,$$
$$y_{k+1} = y_k + h \cdot u(t_k),$$

где h – шаг интегрирования.

Или более точный метод – **Метод Рунге–Кутты (4-го порядка)**

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

где k_i – векторы приращений, вычисляемые по стандартной схеме.

Для сложных критериев и ограничений используются **методы стрельбы** (shooting methods), подробно изложен в разделе 2.6 или **MPC** (Model Predictive Control) – решение задачи оптимизации на скользящем горизонте, подробно изложен в разделе 4.5.1.

Однако сложность этой задачи существенно возрастает, если учесть фазовые ограничения переменной x. Пусть на пути есть препятствие, которое нужно обойти рассматривемому объекту в процессе движения. Тогда фазовая траектория подчиняется ограничению, заданному множеством уровней некоторого отображения

$$g(x(t)) \leq 0, \quad \forall t \in [0, T], \tag{2.3}$$

которое моделирует упомянутое препятствие. В таком случае решение задачи неочевидно. Главный вопрос заключается в том, как найти точки соединения, то есть точки входа и выхода на границы ограничения, а также сооответствующий закон управления. Фактически, эта задача далека от полного решения, несмотря на простоту ее формулировки.

Ограничение (2.3) известно как фазовое ограничение глубины 2 по отношению к системе (2.1). Переменная $x \in R^n$ обычно называется положением. Переменная $y \in R^n$ — скоростью. Существует множество примеров систем управления с фазовыми ограничениями глубины 2 в различных инженерных приложениях, особенно в робототехнике.

Рассмотрим следующий пример на плоскости R^2:

$$\begin{cases} \dot{x}_1(t) = v(t)\cos\alpha(t) + V_1(x(t)), \\ \dot{x}_2(t) = v(t)\sin\alpha(t) + V_2(x(t)), \\ \dot{\alpha}(t) = \omega(t), \\ \dot{v}(t) = u_L(t) + u_R(t), \\ \dot{\omega}(t) = u_L(t) - u_R(t), \\ (u_L(t))^2 + (u_R(t))^2 \leq 1, \\ g(x(t)) \leq 0. \end{cases} \tag{2.4}$$

Здесь $x = (x_1, x_2)$ – положение робота на плоскости; α – угол поворота; v – продольная скорость; ω – угловая скорость; u_L – крутящий момент, приложенный к левой гусенице, а u_R – крутящий момент на правой гусенице. В этой модели предполагается, что сумма крутящих моментов пропорциональна продольному ускорению, а их разность – угловому ускорению. Эта система представляет собой упрощенную модель мобильного робота на гусеничном ходу без трения,

движущегося в плоскости стационарного векторного потока V с некоторым препятствием.

Система (2.4), очевидно, является частным случаем (2.1), (2.3), при $n = 3$.

Ограничения на u_L и u_R, заданные единичной окружностью, соответствуют так называемому режиму управления с низким выходом или экономией энергии батареи. В более общем виде геометрические ограничения можно определить следующим образом:

$$(u_L(t))^p + (u_R(t))^p \leq 1, \tag{2.5}$$

где $p \geq 1$, и случай $p = 1$ не исключается. Число p в такой модели является пределом допустимой выходной мощности. Уточним, что мощность привода для каждой из дорожек ограничена. Источник питания или аккумулятор, подающий контролируемый ток на привод, считается достаточным для того, чтобы оба привода могли достичь максимальной мощности в режиме управления полной выходной мощностью. Однако в режиме экономии энергии батареи это уже невозможно, и, соответственно, мощность приводов ограничена в силу распределения полного тока от батареи из-за указанного выше p-эллиптического закона. Сам ток может быть пропорционален крутящему моменту, но существует ограничение на распределение тока от основного источника. Таким образом, при $p = 1$, имеется линейное распределение токов, а выходная мощность является минимально возможной. При $p = 2$ распределение квадратичное, а выходная мощность больше, и так далее: увеличивая p, можно постепенно изменять выходную мощность. Наконец, при $p = \infty$, используется режим управления полной выходной мощностью, когда оба привода могут достичь максимальной мощности. Тогда допустимое множество задается квадратом.

Отметим, что предельный случай $p = 1$ приводит к негладкости допустимого множества. Это, в свою очередь, приводит к отсутствию некоторых свойств регулярности для данного экстремального процесса управления, которые обсуждаются далее. Более того, случай квадрата имеет также известную плохую особенность в виду особого управления. Отсутствие регулярности не позволяет в полной мере применить вычислительный метод, предложенный в этой работе, для $p = 1$. В то же время, на практике всегда можно заменить режим полного выхода на квазиполный вывод, учитывая довольно большое число p. Недостаток этого подхода, очевидно, связан с некоторой потерей эффективности, из-за искусственных ограничений на распределение общего тока. Однако, по-видимому, такое снижение эффективности должно быть в целом пренебрежимо малым по сравнению с задачами, стоящими перед роботом.

Основная идея, возникающая здесь, заключается в применении модифицированного принципа максимума, адаптированного для задач оптимального управ-

ления с фазовыми ограничениями глубины 2. На основе этих условий оптимальности предлагается некоторый вычислительный алгоритм для решения таких задач в рамках косвенного подхода, который по существу опирается на условие 2-регулярности фазовых ограничений. Для этой цели исследуется вопрос C^1-непрерывности множителя Лагранжа $\mu(t)$ и пределяются соответствующие оценки скачка его производной в точках соединения. Затем предлагаемый алгоритм применяется для нахождения численного решения типовой задачи.

Вопрос о задачах оптимального управления с фазовыми ограничениями глубины 2 ранее изучался в литературе. Например, можно отметить вклад, внесенный в [38, 39, 83, 88, 95]. В то же время этот список работ далеко не исчерпывающий, в то время как количество публикаций по данной теме велико и, следовательно, может потребовать отдельного обзора. В данной работе по существу используется подход, предложенный в [56]. Новизна исследования в отношении численных инструментов заключается в приведенных выше оценках скачка производной множителя, которые, по сути, используются при построении модифицированного алгоритма стрельбы. Это исследование также можно рассматривать как дальнейшее развитие алгоритмов из [67, 68], в которых были изучены задачи управления с фазовыми ограничениями.

2.2 Постановка задачи оптимального управления с фазовыми ограничениями глубины 2

Рассмотрим следующую задачу оптимального управления с фазовыми ограничениями глубины 2:

$$\begin{cases} T \to min; \\ \dot{x}(t) = f_1(x(t); y(t)), \\ \dot{y}(t) = f_2(x(t); y(t); u(t)), \\ x(0) = A, \\ x(T) = B, \\ (y(0), y(T)) \in S, \\ u(t) \in U \text{ для п.в. } t \in [0, T], \\ g(x(t)) \leq 0 \quad \forall t \in [0, T]. \end{cases} \tag{2.6}$$

Здесь $x \in R^n$ – положение переменной состояния, $y \in R^n$ – скорость переменной состояния и $u \in R^m$ – переменная управления.[1] Предполагается, что движение объекта начинается в точке A и, в кратчайшее время, необходимо попасть в

[1]Термины «положение» и «скорость» не должны вводить в заблуждение, поскольку они являются собирательными и, как правило, не следует предполагать, что $\dot{x} = y$. Вектор y

конечную точку B, удовлетворяя граничным условиям для скорости, заданной множеством S. В то же время положение не должно покидать заданную область, определяемую множеством уровней отображения $g(x)$. Таким образом, моделируется гарантированное отсутствие столкновения с нежелательным объектом.

Множества S и U предполагаются замкнутыми. Отображение

$$\mathbf{f} = (f_1, f_2), \quad \text{где} \quad f_1 : \mathbb{R}^{2n} \to \mathbb{R}^n, \quad f_2 : \mathbb{R}^{2n} \times \mathbb{R}^m \to \mathbb{R}^n,$$

которое является правой частью, и отображение $g : \mathbb{R}^n \to \mathbb{R}$, которое определяет фазовые ограничения, предполагаются достаточно гладкими. Допустимое управление $u(\cdot)$ рассматривается как функция из пространства $\mathcal{L}_\infty([0,T];\mathbb{R}^m)$ со значениями в U. Допустимая траектория $\xi(\cdot) = (x(\cdot), y(\cdot))$ непрерывна по Липшицу и удовлетворяет дифференциальному уравнению

$$\dot{\xi}(t) = \mathbf{f}(\xi(t), u(t))$$

для п.в. $t \in [0,T]$, граничные условия, задаются точками A, B и множеством S, а также неравенством $g(x(t)) \le 0 \ \forall t \in [0,T]$.

Четверка $(x(\cdot), y(\cdot), u(\cdot), T)$ называется допустимым процессом в задаче (2.6), когда выполнены условие попадания в конечную точку, условия на управление и фазовые ограничения, то есть когда $x(\cdot)$, $y(\cdot)$ и $u(\cdot)$ допустимы на $[0,T]$. Допустимый процесс $(\bar{x}(\cdot), \bar{y}(\cdot), \bar{u}(\cdot), T_*)$ называется оптимальным, если число T является наименьшим возможным на множестве всех допустимых процессов. Это раскрывает понятие так называемого глобального минимума. Далее предполагается, что задача (2.6) обладает по крайней мере одним оптимальным процессом $(\bar{x}(\cdot), \bar{y}(\cdot), \bar{u}(\cdot), T_*)$.

Относительно заданных конечных точек мы примем следующую гипотезу как часть постановки задачи.

Предположение 1 *Предположим, что* $g(A) < 0$ *и* $g(B) < 0$.

Это означает, что начальная и конечная точки находятся строго включены внутри множества, заданного фазовыми ограничениями. Такое допущение позволяет нам избежать известного явления вырожденности принципа максимума и, следовательно, значительно упростить представление. Явление вырожденности для ограничений глубины k смотрите, например, в [50, 71, 77] и в [56].

представляет собой совокупность скоростей различного типа для данной технической системы. Например, в соответствии с (2.4), имеем $y = (\omega, v, 0)$, и, таким образом, это совокупность угловых и продольных скоростей.

Обратите внимание, что гипотеза 1 выглядит естественной в отношении робототехнической модели, рассмотренной в разделе 2.1.

Рассмотрим производные 1-го и 2-го порядков отображения $g(x)$ относительно дифференциальнй системы (2.1). Эти производные обозначаются как Γ_1 и Γ_2 соответственно. По определению имеем

$$\Gamma_1(x,y) = \langle g'(x), f_1(x,y)\rangle,$$

$$\Gamma_2(x,y,u) = g''[f_1(x,y)]^2 + \langle g'(x), f_1'(x,y)\mathbf{f}(x,y,u)\rangle.$$

Рассмотрим функцию Гамильтона-Понтрягина второго порядка

$$\mathcal{H}(x,y,u,\psi,\mu) = \langle \psi, \mathbf{f}(x,y,u)\rangle + \mu\Gamma_2(x,y,u),$$

где $\psi = (\psi_1,\psi_2) \in (\mathbb{R}^{2n})^*$ — вектор-строка, а μ – скаляр.Эта функция играет ключевую роль в дальнейших исследованиях. Перейдем к формулировке условий оптимальности.

2.3 Принцип максимума

Далее сформулируем принцип максимума Понтрягина в терминах функции Гамильтона-Понтрягина второго порядка.

Определение 8 *Говорят, что процесс управления $(\bar{x}(\cdot),\bar{y}(\cdot),\bar{u}(\cdot),T_*)$ в задаче (2.6) удовлетворяет принципу максимума при условии, что существуют множители Лагранжа: вектор-функция $\psi = (\psi_1,\psi_2) \in \mathbb{W}_{1,\infty}([0,T_*];(\mathbb{R}^{2n})^*)$, число $\lambda \geq 0$, и скалярная функция $\mu \in \mathbb{W}_{1,\infty}([0,T_*];\mathbb{R})$ такие, что $\dot{\mu}(t)$ убывает, в то время как $\mu(T_*) = \dot{\mu}(T_*) = 0$, и выполняются следующие условия:*

$$\begin{cases} \dot{\psi}_1(t) = -\mathcal{H}'_x(\bar{x}(t),\bar{y}(t),\bar{u}(t),\psi(t),\mu(t)), \\ \dot{\psi}_2(t) = -\mathcal{H}'_y(\bar{x}(t),\bar{y}(t),\bar{u}(t),\psi(t),\mu(t)) \quad \textit{для п.в. } t \in [0,T_*], \end{cases} \tag{2.7}$$

$$\big(\psi_2(0) + \mu(0)\Gamma'_{1y}(A,\bar{y}(0)), -\psi_2(T_*)\big) \in N_S(\bar{p}), \tag{2.8}$$

$$\max_{u\in U} \mathcal{H}(\bar{x}(t),\bar{y}(t),u,\psi(t),\mu(t)) = \mathcal{H}(\bar{x}(t),\bar{y}(t),\bar{u}(t),\psi(t),\mu(t)) \textit{ для п.в. } t \in [0,T_*], \tag{2.9}$$

$$\max_{u\in U} \mathcal{H}(B,\bar{y}(T_*),u,\psi(T_*),0) = \lambda, \tag{2.10}$$

$$\int_0^{T_*} g(\bar{x}(t))d\dot{\mu}(t) = 0, \tag{2.11}$$

$$\lambda + |\psi(0)| > \mu(0). \tag{2.12}$$

Здесь $\bar{p} = (\bar{y}(0), \bar{y}(T_*))$, $N_S(p)$ – предельный нормальный конус к S в точке p, как определено в [37]; $\dot{\mu}$ – производная множителя; $\mathbb{W}_{1,\infty}$ – пространство функций, непрерывных по Липшицу. Понятно, что в силу наложенных условий функция $\dot{\mu}$ неотрицательна, а функция μ возрастает и неположительна. Следовательно, последнее неравенство (2.12) означает, что все множители не обращаются в нуль одновременно.

Дифференциальное уравнение (2.7) называется сопряженным. Включение (4.62) – это условие трансверсальности. Условие (2.9) является условием максимума. Равенство (2.10) – это условие трансверсальности по времени. Равенство (2.11) называется условием дополняющей нежесткости. Условие (2.12) – это условие нетривиальности. Процесс управления, удовлетворяющий принципу максимума, называется экстремальным.

Эта форма условий оптимальности, основанная на расширенной функции Гамильтона-Понтрягина 2-го порядка, была предложена в [39], где она была введена в общей постановке k-го порядка и на фиксированном временном интервале. Наряду с представленным принципом максимума необходимо учитывать следующие важные замечания.

Замечание 6 *Из условия* (2.11) *следует, что* $\mu(\cdot)$ *является линейной функцией на любом временном интервале* $[c, d]$, *на котором* $\bar{x}(t)$ *строго включено в фазовые ограничения, то есть, когда* $g(\bar{(t)}) < 0 \ \forall t \in [c; d]$. *Таким образом,* $\mu(t) = \alpha_0 + \alpha_1(t - c)$, *где* $\alpha_0 = \mu(c)$, $\alpha_1 = \dot{\mu}(c)$. *Действительно,* $\dot{\mu}(t)$ *является постоянной на* $[c; d]$ *ввиду* (2.11).

Замечание 7 *Из условий принципа максимума выводится закон сохранения в следующем виде:*

$$\max_{u \in U} \mathcal{H}(\bar{x}(t), \bar{y}(t), u, \psi(t), \mu(t) - \dot{\mu}(t)\Gamma_1(\bar{x}(t), \bar{y}(t)) = \lambda \quad \forall \in [0, T_*]. \tag{2.13}$$

Доказательство этого условия может быть найдено в [56].

Замечание 8 *Рассмотрим множество множителей Лагранжа* (λ, ψ, μ), *которое удовлетворяет принципу максимума. Возьмем произвольный вектор* $a = (\alpha, \beta) \in \mathbb{R}^2$. *Тогда множество множителей*[2]

$$(\lambda, \psi(t) + \alpha g'(\bar{x}(t)) - (\alpha t + \beta)\Gamma_1'(\bar{x}(t(, \bar{y}(t)), \mu(t) + \alpha t + \beta) \tag{2.14}$$

[2]Здесь и далее производная $g'(\xi)$ понимается как производная относительно полной переменной состояния $\xi = (x, y)^T$.

снова удовлетворяет условиям (2.7), (2.9) *и* (2.11), *[56]. В то же время условия* (4.62), (2.10) *и* (2.12) *изменяются при линейном сдвиге. В связи с этим возможно рассмотреть инвариантную форму этих трех условий оптимальности в следующем виде:*

$$\big(\psi_2(0) + \mu(0)\Gamma'_{1y}(A, \bar{y}(0)), -\psi_2(T_*) - \mu(T_*)\Gamma'_{1y}(B, \bar{y}(T_*))\big) \in N_S(\bar{p}), \tag{2.15}$$

$$\max_{u\in U} \mathcal{H}(B, \bar{y}(T_*), u, \psi(T_*), \mu(T_*)) - \dot{\mu}(T_*)\Gamma_1(B, \bar{y}(T_*)) = \lambda, \tag{2.16}$$

$$\lambda + |\psi(0)| + \dot{\mu}(0) - \dot{\mu}(T_*) > 0. \tag{2.17}$$

в то время как множитель $\mu(\cdot)$ *таков, что* $\mu(\tau) = \dot{\mu}(\tau) = 0$ *для некоторой произвольной и априори фиксированной точки* $\tau \in [0, T_*]$. *Несложно доказать, что условия* (2.15)–(2.17) *не меняются при преобразовании* (2.14). *Следовательно, существуют две эквивалентные формулировки принципа максимума из определения 8. Одна формулировка задается условиями* (2.7)–(2.12) *плюс граничное условие на* $\mu(t)$. *Другая формулировка* (2.7), (2.9), (2.11) *и* (2.15)–(2.17) *является инвариантной. Первая форма удобна с точки зрения приложений, в то время как вторая форма условий оптимальности важна с теоретической точки зрения. Заметим, что закон сохранения* (2.13) *также является инвариантным условием.*

Справедливо следующее утверждение.

Теорема 3 *Предположим, что процесс управления* $(\bar{x}(\cdot), \bar{y}(\cdot), \bar{u}(\cdot), T_*)$ *в задаче* (2.6) *оптимален. Тогда он удовлетворяет принципу максимума.*

Эта теорема легко следует из уже известного условия оптимальности в силу следующего сокращения:

$$\begin{cases} T \to min; \\ \dot{x}(t) = f_1(x(t); y(t)), \\ \dot{y}(t) = f_2(x(t); y(t); u(t)), \\ \dot{\chi}_1(t) = \chi_2(t), \\ \dot{\chi}_2(t) = \Gamma_2(x(t), y(t), u(t)), \\ x(0) = A, \quad x(T) = B, \\ p = (y(0), y(T)) \in S, \\ \chi_1(0) = g(A), \\ \chi_2(0) = \Gamma_1(A, y(0)), \\ u(t) \in U \text{ для п.в. } t \in [0, T], \\ \chi_1(t) \le 0 \quad \forall t \in [0, T]. \end{cases} \tag{2.18}$$

Задачи (2.6) и (2.18) эквивалентны и их множителя одинаковы. Тогда применение обычного принципа максимума для задач с фазовыми ограничениями, см., например, в [72], к (2.18) дает искомое утверждение.

Более того, ввиду замены сопряженной переменной

$$\tilde{\psi}(t) = \psi(t) - \mu(t)g'(\bar{x}(t)) + \dot{\mu}(t)\Gamma_1'(\bar{x}(t), \bar{y}(t)), \tag{2.19}$$

обе формы условий оптимальности, то есть из определения 8, и обычная форма, эквивалентны, поскольку можно перейти от одного набора множителей Лагранжа к другому. Подробнее см. [56].

Теперь перейдем к понятию 2-регулярности.

2.4 Условия 2-регулярности и C_1-непрерывности для множителя

Реализация наших вычислительных усилий и алгоритма потребует некоторых дополнительных ограничений на данные задачи. В этом разделе такие дополнительные предположения представлены вместе с соответствующими утверждениями о свойствах множителя, необходимого для вычислений. Здесь и далее будем считать, что

$$U = \{u \in \mathbb{R}^m : \varphi(u) \leq 0\},$$

где $\varphi : \mathbb{R}^m \to \mathbb{R}^q$ – заданная гладная вектор-функция. Напомним, что точка $u \in U$ называется регулярной при условии, что

$$\dim \operatorname{im} P(u)\varphi'(u) = |I(u)|,$$

где $I(u) = \{i : \varphi^i(u) = 0$ – множество активных индексов, $P(u)$ – диагональная квадратная матрица, элементы (i, j) которой равны бесконечности, если $i \in I(u)$ нулю в противном случае, и $|M|$ означает мощность данного множества M. Другими словами, градиенты активного φ^i предполагаются линейно независимыми.

Что касается допустимого множества U, то везде выдвигается следующая гипотеза.

Предположение 2 *Любая точка $u \in U$ регулярна.*

Основное условие регулярности, которое накладывается на фазовое ограничение, заключается в следующем.

Определение 9 *Фазовые ограничения называются 2-регулярными, если для всех* $x, y \in \mathbb{R}^n$ *и* $u \in U$ *таких, что* $g(x) = \Gamma_1(x, y) = \Gamma_2(x, y, u) = 0$, *выполняется соотношение*

$$(\Gamma_2)'_u(x, y, u) \notin \operatorname{im} P(u)\varphi'(u). \tag{2.20}$$

Понятие 2-регулярности является глобальным предположением, которое легко проверить априори для заданных данных. Этот факт важен с точки зрения приложений. В то же время, следует отметить, что такое условие регулярности довольно сильное и может не выполняться во многих представляющих интерес случаях (например, всегда, если существует $u_0 \in U : |I(u_0)| \geq m$). В приведённых ниже доказательствах используется лишь регулярность относительно опорных экстремальных значений $\bar{x}(\cdot)$, $\bar{y}(\cdot)$ и $\bar{u}(\cdot)$. Таким образом, понятие 2-регулярности может быть ослаблено с глобального типа до локального. Однако локальный тип условия обычно не может быть проверен априори, а только в ходе вычисления экстремалей.

Начнем изучение с линейного случая, описанного в следующем предложении. Такой случай можно встретить в приложениях. В частности, то имеет место в задаче (2.4) при $p = \infty$.

Лемма 4 *Предположим, что отображения* $f_2(x, y, u)$ *и* $\varphi(u)$ *являются линейными функциями относительно переменной управления* u, *а фазовые ограничения 2-регулярны. Рассмотрим точку* $t_0 \in [0, T]$. *Предположим, что множество индексов* $I(\bar{u}(t))$ *непрерывно в точке* t_0.

Тогда для любого набора множителей Лагранжа $(\bar{x}(\cdot), \bar{y}(\cdot), \bar{u}(\cdot), T_*)$, *удовлетворяющего принципу максимума из определения 8, существует производная* $\mu(t)$ *в точке* $t = t_0$.

Доказательство леммы приведено в [36].

В случае нарушения линейности неясно, как доказать непрерывную дифференцируемость множителя при предложенной гипотезе. Действительно, исходя из приведенных выше рассуждений, придется потребовать непрерывной дифференцируемости экстремального управления. Однако это означает, что толчок является непрерывным, в то время как такое требование слишком ограничительно и не имеет отношения к задаче с фазовыми ограничениями, поскольку толчок (в отличие от ускорения) обычно должен быть разрывным при выходе на границу фазовых ограничений, что будет численно подтверждено далее. Тем не менее, следует предложить некоторую оценку для скачка $\dot{\mu}$ в момент выхода на границу. Обозначим $\mathcal{Y}_* = [0, T_*]$ и $\mathcal{Y}_0 = \{t \in \mathcal{Y}_* : g(\bar{x}(t)) = 0\}$. Очевидно, что для любого $t \in \mathcal{Y}_0$ имеем $\Gamma_1(\bar{x}(t), \bar{y}(t)) = 0$. В дальнейшем мы будем опираться на следующие основные предположения.

Предположение 3 *Множество $\mathcal{Y}_0$ является объединением конечного числа интервалов.*

Эта гипотеза требует, чтобы число точек касания было конечным. Напомним, что точка $\tau = \partial\mathcal{Y}_0$ называется точкой касания. Очевидно, что такое предположение не ограничивает вычисления, поскольку касание границы всегда происходит с некоторой «нулевой точностью».

Следующая гипотеза на самом деле также не является ограничительной в отношении задачи о роботе (2.4), как будет показано в дальнейшем изложении.

Предположение 4 *Оптимальное управление $\bar{u}(t)$ является кусочно-непрерывным.*

Можно оценить скачок производной множителя следующим образом. Точка пересечения $\tau \in T_0$, такая, что $g(\bar{x}(\tau+\varepsilon)) < 0$ для всех достаточно малых $\varepsilon > 0$, называется точкой выхода за границу. Если $g(\bar{x}(\tau-\varepsilon)) < 0$ для всех достаточно малых $\varepsilon > 0$, то называется точкой пересечения границы. Для данной функции $\omega(t)$ положим $\omega(\tau^+) = \lim_{t\to\tau^+}$ и $\omega(\tau^-) = \lim_{t\to\tau^-}$, его правый и левый пределы в τ соответственно.

Лемма 5 *Предположим, что процесс управления $(\bar{x}(\cdot), \bar{y}(\cdot), \bar{u}(\cdot), T_*)$ оптимален для задачи (2.6), и фазовые ограничения 2-регулярны. Рассмотрим точку $t_0 \in \mathcal{Y}_0$ выхода за границу. Предположим, что*

$$\Gamma_2(\bar{x}(t_0), \bar{y}(t_0), u_0^+) = 0, \tag{2.21}$$

где $u_0^+ = \bar{u}(t_0^+)$.

Тогда для любого набора множителей Лагранжа (λ, ψ, μ), удовлетворяющего принципу максимума из Определения 1, справедлива оценка

$$\Delta\dot{\mu}(t_0) \le C \cdot \left|\psi(t_0) - \dot{\mu}(t_0^-)g'(\bar{x}(t_0)) + \mu(t_0)\Gamma_1'(\bar{x}(t_0), \bar{y}(t_0))\right|, \tag{2.22}$$

где $\Delta\dot{\mu} = \dot{\mu}(t_0^-) + \dot{\mu}(t_0^+)$, а C – это некоторая константа, которая не зависит от множителей.

Аналогично, если $\Gamma_2(\bar{x}(t_0), \bar{y}(t_0), u_0^-) = 0$, где $u_0^- = \bar{u}(t_0^-)$, и t_0 – точка пересечения с границей, тогда у нас есть оценка

$$\Delta\dot{\mu}(t_0) \le C \cdot \left|\psi(t_0) - \dot{\mu}(t_0^+)g'(\bar{x}(t_0)) + \mu(t_0)\Gamma_1'(\bar{x}(t_0), \bar{y}(t_0))\right|. \tag{2.23}$$

Доказательство леммы приведено в [36].

Полученные оценки пока не имеют смысла с точки зрения вычислений, если не вычислять присутствующую в них константу C. Следующая лемма даёт выражение для такой константы, хотя и при некоторых дополнительных предположениях.

Введём понятие зоны регулярности. Определим

$$\Phi(\xi,u)=\max_{d\in\ker P(u)\varphi'(u),|d|=1}\langle(\Gamma_2)'_u(\xi,u),d\rangle.$$

Здесь максимум по пустому множеству считается равным нулю. Тогда легко увидеть, что $\Phi(\xi,u)\geq 0$. Рассмотрим число $\delta>0$. Тогда зоной регулярности будет множество точек

$$M(\delta)=\left\{(\xi,u)\in\mathbb{R}^{2n}\times U:\Phi(\xi,u)\geq\delta\right\},$$

в то время как число N является параметром для вычислительного метода.

Рассмотрим оптимальный процесс $(\bar{x}(\cdot),\bar{y}(\cdot),\bar{u}(\cdot),T_*)$. Определим постоянные величины

$$\kappa_1=\max_{t\in[0,T_*]}\left|\mathbf{f}'_\xi(\bar{x}(t),\bar{y}(t),\bar{u}(t))\right|,$$

$$\kappa_2=\max_{t\in[0,T_*]}\left|(\Gamma_2)'_\xi(\bar{x}(t),\bar{y}(t),\bar{u}(t))\right|,$$

$$\kappa_3=\max_{t\in[0,T_*]}\left|\mathbf{f}'_u(\bar{x}(t),\bar{y}(t),\bar{u}(t))\right|.$$

Можно заметить, что когда U и S компактны, а отображение f удовлетворяет условию линейного роста относительно ξ, эти три остоянные величины можно оценить с помощью общей универсальной постоянной, действительной для всех допустимых значений x, y, u, и которая монотонно зависит от T_*. Таким образом, если $T\leq T_{adm}$ для некоторого априори известного допустимого перемещения во времени T_{adm}, то постоянные κ_1, κ_2, κ_3 можно заменить универсальной и априори заданной постоянной κ, которая не зависит от исходного процесса управления. (Точное выражение для κ получить просто.)

Лемма 6 *Предположим, что* $(\bar{x}(t),\bar{y}(t),\bar{u}(t))\in M(\delta)\ \forall t\in\mathcal{Y}_*$ *для некоторого* $\delta>0$. *Пусть* ε_* *– минимально возможное расстояние между двумя последовательными точками пересечения границы и выхода с нее* [3]*, а* s_* *– единственный положительный корень уравнения*

$$s\left(\kappa_1+\kappa_2 K e^{\kappa_1 s}\right)=1,$$

[3]Если таких двух последовательных точек не существует, тогда положим $\varepsilon_*=+\infty$.

где $K = \kappa_3 \delta^{-1}$.

Тогда, согласно лемме 2, определим

$$C = \frac{2Ke^{\kappa_1 \tau_*}}{2\tau_* - K\kappa_2 \tau_*^2 e^{\kappa_1 \tau_*}},$$

где $\tau_* = \min\{s_*, \varepsilon_*\}$.

Доказательство леммы приведено в [36].

Введение зоны регулярности позволяет добиться некоторой адаптивности вычислительного метода, предложенного в данной работе. Наряду с определением 1 можно также рассмотреть следующее понятие 2-регулярности. Пусть $\mathcal{Y}_0(\varepsilon)$ обозначает ε-окрестность множества активных точек $\mathcal{Y}_0$.

Определение 10 *Процесс управления* $(\bar{x}(\cdot), \bar{y}(\cdot), \bar{u}(\cdot), T_*)$ *называется 2-регулярным относительно фазовых ограничений, если существуют числа* $\delta > 0$ *и* $\sigma > 0$ *такие, что*

$$(\bar{x}(\cdot), \bar{y}(\cdot), \bar{u}(\cdot)) \in M(\delta) \quad \forall t \in \mathcal{Y}_0(\delta). \tag{2.24}$$

Числа δ и σ являются параметрами адаптивного вычислительного метода, предлагаемого далее. Понятие 2-регулярности процесса управления имеет локальный характер и оно не поддается априорной проверке, в отличие от понятия 2-регулярности из Определения 2, которое является глобальным и легко подтверждается или отбрасывается заранее. В то же время условие (2.21) в общей ситуации выглядит ограничивающим[4]. Однако, согласно определению 3, можно предложить следующий аналог леммы 2, который не основан на (2.21).

Следствие 1 *Предположим, что процесс управления* $(\bar{x}(\cdot), \bar{y}(\cdot), \bar{u}(\cdot), T_*)$ *оптимален и является 2-регулярным относительно фазовых ограничений. Тогда для каждой точки* $t_0 \in \mathcal{Y}_0$ *выхода за границу имеет место оценка* (2.22)*, а для каждой точки пересечения границы оценка* (2.23)*, в которой*

$$C = \frac{2Ke^{\kappa_1 \tau_*}}{2\tau_* - K\kappa_2 \tau_*^2 e^{\kappa_1 \tau_*}},$$

где $\tau_* = \min\{s_*, \varepsilon_*, \sigma\}$*, а* K*,* s_**,* ε_* *определены в лемме 3.*

[4]Это условие заведомо выполняется, когда экстремальное управление непрерывно в точках сопряжения. Вопрос непрерывности управления обсуждается далее.

Доказательство такое же, как в леммах 2 и 3.

Заметим, что условие 2-регулярности можно рассматривать на любом заданном подинтервале $[a;b] \subseteq \mathcal{Y}_*$, который охватывает некоторые заданные точки стыка. Тогда константы δ и σ зависят от $[a;b]$. Более того, для заданной точки стыка t_0 можно иметь свои собственные δ и σ, поскольку аргументы следствия 1 должны, по сути, выполняться на подинтервале $[t_0 - \sigma, t_0]$, если t_0 – точка пересечения границы, и на $[t_0; t_0 + \delta]$, если t_0 – точка выхода с границы. Это замечание приводит к вопросу об адаптивности, поскольку для различных точек соединения могут быть разные δ и σ, что может улучшить основную константу x в (2.23) для конкретной заданной точки соединения t_0. Таким образом, согласно вычислительному алгоритму, предложенному в данной работе, необходимо обновлять δ и σ во время вычисления экстремального процесса, особенно в точках соединения. В этих точках алгоритму необходимо знать верхнюю границу для скачка производной множителя. Затем вызывается условие 2-регулярности вместе с соответствующими числами δ и σ.

Во-первых, рассмотрим случай встречи с границей в некоторой точке пересечения t_0. В процессе управления стрельбой кандидат на экстремальность вычислен для $t < t0$, теперь требуется найти δ и σ для t_0. Тогда удет известна постоянная в (2.23). В то же время вычисляется $\mu(t)$ пока траектория проходит вдоль границы. Таким образом, скачок $\dot{\mu}$ в t_0 известен, и необходимо проверить, соответствует ли он оценке (2.23) если нет, то текущий угол стрельбы следует отбросить. После того, как точка покидает границу, мы продолжаем отслеживать зону регулярности, то есть условие $(\bar{x}(\cdot), \bar{y}(\cdot), \bar{u}(\cdot)) \in M(\delta)$, чтобы понять необходимое значение σ. Пусть первый выход из зоны регулярности обозначен как t_*, то есть момент времени, когда $t_* > t0$ и $\Phi(\bar{x}(t_*), \bar{y}(t_*), \bar{u}(t_8) = \delta$. Как только $t_* - t_0 \geq \tau_*$, все правильно, и процедура вычисления продолжается. Если $t_* - t_0 < \tau_*$, то необходимо повторно вычислить константу C, указанную выше, заменив τ_* ее значением $\sigma = t_* - t_0$. Новое значение константы больше старого значения. Следовательно, дальность скачка производной увеличивается ввиду постоянного обновления константы. При этом, если постоянная C достигнет некоторого верхнего предела L, который можно рассматривать как ещё один параметр метода, то программа формирует следующее выходное сообщение:

Ошибка угла на границе! Пожалуйста, увеличьте зону регулярности.

В этом случае можно попытаться избежать возникающей ошибки, уменьшив число δ. В этом смысле понимется адаптивность.

Таким образом, при подходе, основанном на зоне регулярности, условие (2.21) уже избыточно, и тем самым гипотеза 3 также может быть снята, поскольку они

были связаны при использовании. Следует отметить, что условия леммы 3 или следствия 1 могут быть проверены только в процессе вычисления и, следовательно, не могут быть выполнены априори. Следовательно, вопрос о некоторых априорных достаточных условиях непрерывности экстремального управления остаётся актуальным для приложений. Ниже мы покажем, что предположение о кусочной непрерывности экстремального управления, выдвинутое в Гипотезе 3, выполняется в задаче о роботе (2.4) с точностью до множества нулевой меры Лебега.

Лемма 7 *Предположим, что допустимое множество U строго выпукло, а $f_2(x;y;u)$ линейна относительно u. Тогда экстремальное управление $\bar{u}(t)$ непрерывно в любой точке t_0, в которой*

$$\psi_2(t) + \mu(t)(\Gamma_1)'_y(\bar{x}(t_0), \bar{y}(t_0)) \neq 0.$$

Доказательство. Это утверждение непосредственно следует из условия максимума (2.9), того факта, что $\mu(t)$ является непрерывной функцией, а также из формулы

$$\Gamma_2(x,y,u) = \langle \Gamma'_1(x,y), \mathbf{f}(x,y,u) \rangle.$$

Действительно, поскольку допустимое множество строго выпукло, аргмаксимум линейной ненулевой функции в (2.9) единственен. Следовательно, он непрерывно зависит от времени, как и траектория и множители.

Таким образом, лемма утверждает, что скачки экстремального управления могут происходить только в нулях функции

$$\alpha(t) = \psi_2(t) + \mu(t)(\Gamma_1)'_y(\bar{x}(t), \bar{y}(t)).$$

Поэтому интерес представляет вопрос о том, может ли такая функция обращаться в нуль на всем подинтервале, но не в одной точке. Чтобы ответить на этот вопрос, сначала рассмотрим более сильное условие нетривиальности.

Лемма 8 *Предположим, что процесс управления $(\bar{x}(\cdot), \bar{y}(\cdot), \bar{u}(\cdot), T_*)$ оптимален и является 2-регулярным относительно фазовых ограничений. Тогда выполняется следующее условие нетривиальности*

$$\psi(t) - \dot{\mu}(t) g'(\bar{x}(t)) + \mu(t)\Gamma'_1(\bar{x}(t), \bar{y}(t)) \neq 0, \quad \forall t \in \mathcal{Y}_*. \tag{2.25}$$

Доказательство. Доказательство следует из замечания 3, закона сохранения и тех же аргументов, приведённых в доказательстве леммы 2. Действительно, пусть (2.25) не выполняется в некоторой точке t_0. Применяя тот же приём, что и в доказательстве леммы 2, можно считать, что $\dot{\mu}(t_0) = \mu(t_0) = 0$, и тогда $\psi(t_0) = 0$. Тогда, с учетом закона сохранения, имеем $\lambda = 0$. Используя неравенство Гронуолла, легко убедиться, что $\mu(t) = |\psi(t)| = 0$ для всех $t \in [t_0, t_0 + \varepsilon_0]$ для некоторого числа $\varepsilon_0 > 0$. В то же время число ε_0 может быть выбрано равномерно в некоторой окрестности $\mathcal{Y}_0$ с учетом предположения о 2-регулярности. Для тех частей, где $t \notin \mathcal{Y}_0$, множитель является линейной функцией с учетом замечания 1 и, следовательно, равен нулю. Таким образом, все множители обращаются в нуль одновременно, что противоречит утверждению теоремы 1.

Рассмотрим замкнутое множество нулей $Z = \{t \in \mathcal{Y}_* : \alpha(t) = 0\}$. Предположим, что $l(Z) > 0$. Используя стандартню технику вычислений, можно получить

$$\begin{aligned}\dot{\alpha}(t) = \dot{\psi}_2(t) + \dot{\mu}(t)(\Gamma_1)'_y(t) + \mu(t)(\Gamma_1)''_{\xi y}(t)\mathbf{f}(t) = \\ = -\psi(t)\mathbf{f}'_y(t) - \mu(t)(\Gamma_2)'_y(t) + \dot{\mu}(t)g'(t)\mathbf{f}'_y(t) + \mu(\Gamma_2)'_y(t) - \mu\Gamma'_1(t)\mathbf{f}'_y(t) = \\ = -\big(\psi(t) - \dot{\mu}(t)g'(t) + \mu(t)\Gamma'_1(t)\big)\mathbf{f}'_y(t) = 0\end{aligned}$$

для почти всех $y \in Z$. Следовательно, из лемм 4 и 5 следует следующее утверждение, которое может обосновать выдвинутую Гипотезу 3 для задачи управления мобильным роботом (2.4).

Следствие 2 *Пусть* $\det(f_1)'_y(\bar{x}(t), \bar{y}(t)) \neq 0 \ \forall t \in \mathcal{Y}_*$. *Тогда множество нулей функции* $\alpha(t)$ *имеет нулевую меру Лебега, то есть* $l(Z) = 0$. *Экстремальное управление строго непрерывно в каждой точке* $t \notin Z$, *то есть замыкание относительно меры Лебега является однозначным множеством.*

Очевидно, что, как правило, скачки экстремального управления также могут возникать в точках соединения, если $\alpha(t)$ обращается в нуль при пересечении границы или выходе за её пределы. Заметим, что условие $\det(f_1)'_y(\bar{x}(t), \bar{y}(t)) \neq 0$ имеет значение невырожденного условия 1-го порядка. Таким образом, в силу условия 2-регулярности, представленного в определении 3, сильная непрерывность экстремального управления обеспечивается почти для всех точек в $\mathcal{Y}_*$.

Далее рассмотрены несколько конкретных примеров, в которых будут проверены обсуждаемые выше условия и построена зона регулярности.

2.5 Алгоритм метода стрельбы

В данном разделе приведен алгоритм метода стрельбы для задачи оптимального управления с фазовыми ограничениями глубины 2.

Алгоритм

Шаг 1. Начало. Стреляем из начальной точки вперед во времени из сферы S3: $|\psi_1(0)|^2 + |\psi_2(0)|^2 = 1$. Установить $\mu(0) = \dot{\mu}(0) = 0$. Тогда $\mu(t)$ является убывающей отрицательной конечной функцией.

Шаг 2. Выход на границу в точке t_0. Проверить условие пересечения границы в некоторой точке t_0 по следующим критериям

$$g(\bar{x}(t_0)) = \Gamma_1(\bar{x}(t_0), \bar{y}(t_0)) = 0 \Leftrightarrow |\bar{x}(t_0)| = R, \quad \langle \bar{x}(t_0), \bar{y}(t_0) \rangle = 0,$$

то есть, 2 уравнения, которые могут быть дополнены еще одним уравнением

$$\Gamma_2(\bar{x}(t_0), \bar{y}(t_0), \bar{u}(t_0)) = 0 \Leftrightarrow |\bar{y}(t_0)|^2 + \langle \bar{x}(t_0), \bar{u}(t_0) \rangle = 0,$$

при $\alpha(t_0) \neq 0$. Действительно, это связано с непрерывностью экстремального управления (лемма 4). Здесь t_0 – точка пересечения границы.

В этой точке имеет место оценка (2.23), записанная в следующем виде

$$\Delta\dot{\mu} \leq C\sqrt{|\psi_1(t_0) - \dot{\mu}(t_0^+)\bar{x}(t_0) + \mu(t_0)\bar{y}(t_0)|^2 + |\psi_2(t_0) - \mu(t)\bar{x}(t_0)|^2},$$

где постоянная C найдена по формуле

$$C = \frac{2Ke^{\kappa_1\tau_*}}{2\tau_* - K\kappa_2\tau_*^2 e^{\kappa_1\tau_*}},$$

где $\tau_* = \min\{s_*, \varepsilon_*, \sigma\}$, а K, s_*, ε_* определены в лемме 3.

Если это неравенство не выполняется, то текущий выстрел отбрасывается.

Шаг 3. Следование по границе. Вычислить $\mu(t)$, используя полученную формулу, и проверить, удовлетворяет ли эта функция известным свойствам. Тогда траектория выстрела проходит вдоль границы.

Шаг 4. Выход с границы. Организовать дополнительную стрельбу с границы следующим образом. На каждом шаге на границе мы продолжаем $\mu(t)$ как линейную функцию с углом отклонения, вычисленным по формуле

$$\Delta\dot{\mu} \leq C\sqrt{|\psi_1(t_0) - \dot{\mu}(t_0^-)\bar{x}(t_0) + \mu(t_0)\bar{y}(t_0)|^2 + |\psi_2(t_0) - \mu(t)\bar{x}(t_0)|^2}.$$

В то же время, в продолжение необходимо проверить фазовые ограничения в смысле подтверждения выхода за пределы границы и зоны регулярности. Если регулярность не выполняется, необходимо выполнить действие, описанное ранее в Разделе 2.4.

Шаг 5. Попадание в цель. Те углы с которыми можно поразить цель $x(\tau) = B$ при стрельбе для некоторого значения $\tau > 0$ с заданной точностью ε_0 и $\tau < T_{adm}$, и такие, при которых выполняется (2.27), рассматриваются как экстремальные вместе с вычисленной траекторией $x(t)$ и управлением $u(t)$, в то время время τ задается в качестве текстового значения экстремального времени T_{EXT}.

Таким образом, экстремали могут быть вычислены. После выбора экстремали можно найти решение, найдя минимально возможное T_{EXT}.

2.6 Результаты численных экспериментов

Рассмотрим двумерную задачу Ньютона об оптимальном времени с заданным на плоскости фазовым ограничением

$$\begin{aligned} &T \to \min, \\ &\dot{x}(t) = y(t), \\ &\dot{y}(t) = u(t), \\ &x(0) = A, \quad x(T) = B, \\ &y(0) = 0, \\ &|u(t)| \le 1, \\ &|x(t)| \ge R, \end{aligned} \tag{2.26}$$

где $x, y, u \in \mathbb{R}^2$, $t \in [0, T]$. Основная идея состоит в том, чтобы двигаться по закону Ньютона и обойти препятствие в виде окружности радиуса R на плоскости. Решение этой задачи можно также назвать кратчайшей кривой второго порядка. Внешняя часть окружности может быть эквивалентно определена неравенством $\frac{1}{2}|x(t)|^2 \ge \frac{R^2}{2}$. Это полезно для некоторого упрощения выражений.

Легко видеть, что условие невырожденности 1-го порядка справедливо везде в (2.26), т.е. $\det(f_1)'_y \neq 0$. Вычислим зону регулярности $M(\delta)$. Согласно (2.26) имеем $g(x) = \frac{1}{2}(R^2 - |x|^2)$ и $\varphi(u) = |u|^2 - 1$ и, следовательно,

$$\Gamma_1(x, y) = -\langle x, y\rangle \qquad \text{и} \qquad \Gamma_2(x, y, u) = -|y|^2 - \langle x, u\rangle.$$

Таким образом,

$$\Phi(x,u) = \begin{cases} |x|, & \text{если } |u| < 1, \\ |\langle x, u^{\perp}\rangle|, & \text{если } |u| = 1. \end{cases}$$

Поскольку $|x| \geq R$, нерегулярность может возникать только в тех точках, где $|u| = 1$ и $\langle x, u^{\perp}\rangle = 0$, которые не содержатся в $M(\delta)$ для любого $\delta > 0$. То есть, когда $u = \pm\frac{x}{|x|}$. Следовательно, полная зона регулярности в данном случае – это множество точек

$$(x,u) : u \neq \pm\frac{x}{|x|}, \quad |u| = 1.$$

Если экстремальный процесс полностью входит в это множество (и, следовательно, в $M(\delta)$ для некоторого достаточно малого $\delta > 0$), то предложенный здесь вычислительный метод корректен.

Предположим, что $\Gamma_2(x,y,u) = 0$ в то время как $|x| = R$. Это соответствует ситуации когда робот остается на границе. Рассмотрим нерегулярную точку (x,u). Тогда, $u = -\frac{x}{|x|}$ и $|x| = |y|^2 = R$ в то время как зона регулярности несколько уменьшается, поскольку значение $u = \frac{x}{|x|}$ недостижимо, когда траектория находится на границе. Может ли какая-то часть допустимой траектории $x(t)$ на границе полностью выходить за пределы зоны регулярности? Ответ положительный. Пусть

$$\ddot{x}(t) = u(t) = -\frac{x(t)}{R}$$

ля всех $t \in [a,b]$ для некоторого $a < b$. Тогда мы имеем дело с равномерным круговым движением, и процесс управления, например, может быть

$$x(t) = \begin{bmatrix} R\cos\frac{t}{\sqrt{R}} \\ R\sin\frac{t}{\sqrt{R}} \end{bmatrix}, \quad y(t) = \begin{bmatrix} -\sqrt{R}\sin\frac{t}{\sqrt{R}} \\ \sqrt{R}\cos\frac{t}{\sqrt{R}} \end{bmatrix}, \quad u(t) = \begin{bmatrix} -\cos\frac{t}{\sqrt{R}} \\ -\sin\frac{t}{\sqrt{R}} \end{bmatrix}$$

Очевидно, что $(x(t), y(t), u(t)) \notin M(\delta)\ \forall\delta > 0,\ \forall t \in [a,b]$.

Тем не менее, может ли такой допустимый процесс управления быть частью решения? Ответ, очевидно, отрицательный. На окружности можно предложить более быстрое и сложное круговое движение, при котором ускорение не направлено к центру окружности, а норма скорости непрерывно возрастает. Таким образом, можно ожидать, что экстремали, как правило, не должны покидать зону регулярности $M(\delta)$ при некотором достаточно малом $\delta > 0$. Это подтверждается последующими расчетами.

Далее применим принцип максимума к задаче (2.26). Имеем

$$\mathcal{H}(x,y,u,\psi,\mu) = \langle\psi_1, y\rangle + \langle\psi_2, u\rangle - \mu(|y|^2 + \langle x, u\rangle)$$

Таким образом,

$$\dot{\psi}_1(t) = \mu(t)\bar{u}(t)$$
$$\dot{\psi}_2(t) = -\psi_1(t) + 2\mu(t)\bar{y}(t)$$

Из условия максимума следует

$$\max_{|u|\le 1}\langle \psi_2(t) - \mu(t)\bar{x}(t), u\rangle = \langle \psi_2(t) - \mu(t)\bar{x}(t), \bar{u}(t)\rangle \Rightarrow$$

$$\Rightarrow \bar{u}(t) = \frac{\psi_2(t) - \mu(t)\bar{x}(t)}{|\psi_2(t) - \mu(t)\bar{x}(t)|}, \quad \forall t : \alpha(t) \neq 0,$$

где $\alpha(t) = \psi_2(t) - \mu(t)\bar{x}(t)$.

На любом граничном интервале, на котором α не имеет нулей, имеем

$$|\bar{y}(t)|^2 + \langle \bar{x}(t), \bar{u}(t)\rangle = 0 \Rightarrow$$

$$\Rightarrow \langle \bar{u}(t), \bar{x}(t)\rangle = \frac{\langle \psi_2(t) - \mu(t)\bar{x}(t), \bar{x}(t)\rangle}{|\psi_2(t) - \mu(t)\bar{x}(t)|} = -|y(t)|^2.$$

Возводя в квадрат, мы получаем квадратное уравнение относительно μ

$$R^2 r(t)\mu^2 - 2\beta(t)r(t)\mu + |\bar{y}(t)|^4|\psi_2(t)|^2 - (\beta(t))^2 = 0,$$

где

$$r(t) = |\bar{y}(t)|^4 - R^2, \beta(t) = \langle \psi_2(t), \bar{x}(t)\rangle.$$

После отбрасывания лишнего корня (рассматривается только случай $r \le 0$) получаем формулу для множителя $\mu(t)$ на границе фазового ограничения

$$\mu(t) = \frac{\beta(t)r(t) - |\bar{y}(t)|^2\sqrt{r(t)\left((\beta(t))^2 - |\psi_2(t)|^2R^2\right)}}{R^2 r(t)}.$$

Заметим, что на окружности радиуса R автоматически выполняется равенство $|\beta(t)| \le |\psi_2(t)|R$ и $|\bar{y}(t)|^2 \le R$. Тогда дискриминант D неотрицателен. В то же время, если $r(t) = 0$, решение невозможно. Этот случай, однако, соответствует точкам (x, u), в которых условие регулярности не выполняется. Также следует помнить, что $\mu(t)$ – это возрастающая, конечная и неположительная функция в теореме 1. Это условие может служить дополнительным фильтром для отбрасывания заведомо неверных углов стрельбы.

Далее необходимо вычислить $\dot{\mu}$ для использования оценки (2.23) и для проведения стрельбы с границы. Имеем

$$\dot{\mu} = \frac{(\dot{\beta}r + \beta\dot{r} - \frac{\dot{r}q + |\bar{y}|^4\dot{q}}{2\sqrt{D}})R^2 r - (\beta r + \sqrt{D})R^2\dot{r}}{R^4 r^2}$$

где

$$\begin{aligned} q &= r(\beta^2 - |\psi_2|^2 R^2), \\ \dot{q} &= \dot{r}(\beta^2 - |\psi_2|^2 R^2) + r(2\beta\dot{\beta} - 2\langle\psi_2, \dot{\psi}_2\rangle R^2), \\ \dot{\beta} &= \langle\bar{y}, \psi_2\rangle + \langle\dot{\psi}_2, \bar{x}\rangle, \\ \dot{\psi}_2 &= -\psi_1 + 2\mu\bar{y}, \\ \dot{r} &= 4|y|^2\langle\bar{y}, \bar{u}\rangle, \\ D &= r\left(\beta^2 - |\psi_2|^2 R^2\right). \end{aligned}$$

Здесь для краткости опущена зависимость от времени.

Из условий трансверсальности, поскольку $\mu(T_*) = 0$, получаем

$$\psi_2(T_*) = 0, \quad \langle\psi_1(T_*), \bar{y}(T_*)\rangle = \lambda \geq 0. \tag{2.27}$$

Наконец, было бы также уместно определить постоянные, введённые в разделе 2.6. Для задачи (2.26), очевидно, принять $\kappa_1 = \kappa_3 = 1$ и $\kappa_2 = 1 + 2c$, где c ограничивает абсолютное значение скорости y: ясно, что $c \leq T_*$. Положим $\varepsilon_* = \sigma = +\infty$, если ожидаем, что ускорение не коллинеарно положению робота (см. выше обсуждение вопроса регулярности). Тогда $\tau_* = s_*$, и, рассматривая достаточно большое допустимое время перемещения T_{adm} такое, что $T_* \leq T_{adm}$, число s_* является единственным положительным корнем следующего нелинейного уравнения

$$s\left(\delta + (1 + 2T_{adm})e^s\right) = \delta.$$

Таким образом, константу C в (2.23) можно определить следующим образом

$$C = \frac{2\delta^{-1}e^{T_*}}{2\tau_* - \delta^{-1}(1 + 2T_{adm})\tau_*^2 e^{t_*}}.$$

Заметим, что число $\delta > 0$ служит параметром метода, что обеспечивает некоторую адаптивность в том смысле, что δ всегда можно уменьшить при необходимости. Однако большие значения δ^{-1} подразумевают рост вычислительной сложности. В то же время большие δ уменьшают фазовое пространство, и тогда некоторые экстремали могут быть потеряны.

В диссертационной работе были проведены численные эксперименты с предложенным методом и были получены следующие результаты, рис. 2.3–2.6.

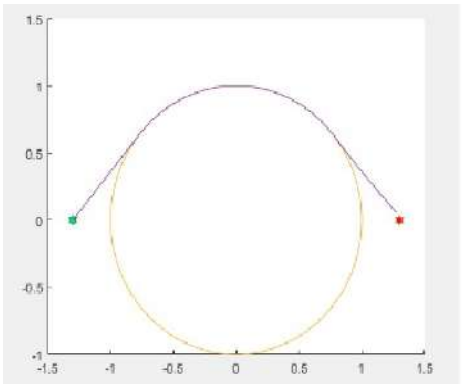

Рис. 2.3: Вычисленное решение задачи (2.26) с точностью $0,075$. Показана оптимальная траектория робота. Препятствие показано сплошным серым цветом.

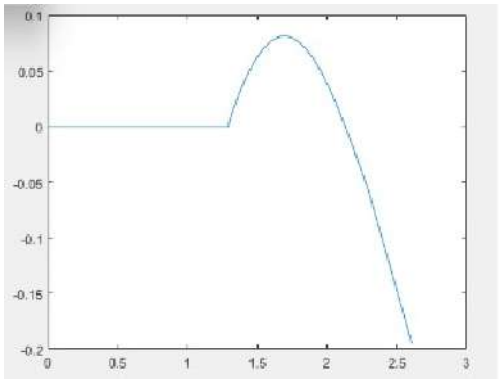

Рис. 2.4: Вычисленное решение задачи (2.26) с точностью $0,075$. Показана эволюция вдоль этой траектории меры-множителя Лагранжа $\mu(t)$.

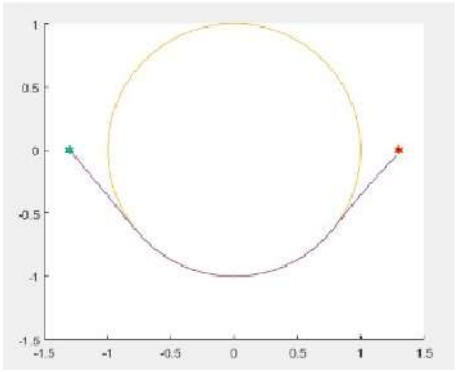

Рис. 2.5: Вычисленное решение задачи (2.26) с точностью $0,0375$. Показана оптимальная траектория робота. Препятствие показано сплошным серым цветом.

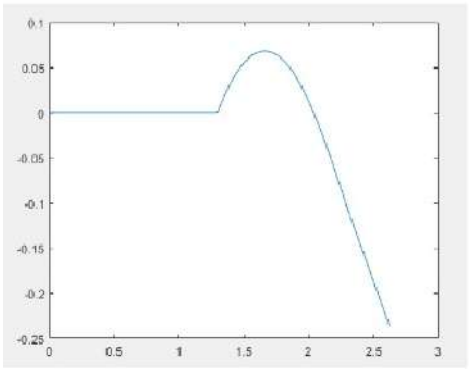

Рис. 2.6: Вычисленное решение задачи (2.26) с точностью $0,0375$. Показана эволюция вдоль этой траектории меры-множителя Лагранжа $\mu(t)$.

2.7 Принцип максимума и ПИД-регулятор

Система дифференциальных уравнений (2.2)

$$\dot{x}(t) = y(t),$$
$$\dot{y}(t) = u(t).$$

является фундаментальной моделью в теории управления, для которого широко применяется ПИД-регулятор (пропорционально-интегрально-дифференциальный регулятор). Эта связь обусловлена тем, что система описывает динамику второго порядка, где управление воздействует на ускорение, а основная цель –

стабилизация положения или слежение за заданной траекторией. Ее решение возможно аналитически при простых $u(t)$, а при наличии целей управления — с помощью методов оптимального управления. Численные методы позволяют решать задачи в реальном времени. Понимание этой системы необходимо для проектирования современных систем автоматики и роботов, где точное и плавное управление движением критически важно.

Это линейная система с постоянными коэффициентами, записанная в форме пространства состояний. Она описывает так называемый двойной интегратор, где

– $x(t)$ – положение (или первая переменная состояния), – $y(t)$ – скорость (вторая переменная состояния), – $u(t)$ – управляющее воздействие (обычно ускорение).

Данный тип системы часто используется в робототехнике и управлении движением, планировании траекторий, задачах оптимального управления (например, минимизация энергии или времени), в методах управления. На практике такие системы используются в автомобильной промышленности (для управления скоростью и остановкой, например, круиз-контроль, парковка; в антиблокировочных системах (ABS), при управлении двигателем), авиации и космонавтике (стабилизация углов тангажа, крена, высоты, при посадке аппаратов с ограниченным ускорением), промышленной автоматике (в системах точного позиционирования, при регулировании температуры, давления, расхода, уровня), в роботах (при управлении скоростью моторов, позиционировании манипуляторов, балансировании, например, в сегвеях), бытовой технике (термостаты, стиральные машины, кондиционеры) и пр.

ПИД-регулятор (пропорционально-интегрально-дифференцирующий регулятор) – это устройство в системе автоматического управления с обратной связью, которое формирует управляющий сигнал с целью точного и качественного поддержания заданного значения контролируемого параметра.

ПИД-регулятор применяется для решения следующих ключевых задач:

– стабилизация системы, т.е. удержание объекта в заданном состоянии (например, поддержание температуры, уровня жидкости, положения);

– слежение за заданной траекторией, т.е. обеспечение движения объекта по заранее заданному закону (например, движение робота к цели);

– компенсация возмущений, т.е. подавление внешних воздействий (ветер, трение, нагрузка), которые отклоняют систему от желаемого поведения;

– устранение статической ошибки, интегральная составляющая позволяет полностью устранить постоянное отклонение, недостижимое при использовании только пропорционального регулятора.

Рассмотрим математическую формализацию ПИД-регулятора в непрерывной и дискретной формах, а также применение к системе второго порядка.

Непрерывная форма ПИД-регулятора. Пусть $e(t) \in \mathbb{R}$ – ошибка управления в момент времени t, определяемая как разность между заданным значением $x_{\text{ref}}(t)$ и текущим состоянием $x(t)$

$$e(t) = x_{\text{ref}}(t) - x(t).$$

Цель управления – асимптотическое стремление положения $x(t)$ к заданному значению $x_{\text{ref}} \in \mathbb{R}$, при этом $x_{\text{ref}} = \text{const}$, т.е. сделать $e(t) \to 0$ при $t \to \infty$ с требуемыми динамическими свойствами (быстродействие, отсутствие перерегулирования и т.д.).

Управляющее воздействие $u(t) \in \mathbb{R}$ формируется по следующему закону – это сумма трех компонентов, умноженных на свои коэффициенты усиления

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}, \tag{2.28}$$

где $K_p > 0$ – коэффициент пропорциональной составляющей, $K_i > 0$ – коэффициент интегральной составляющей, $K_d > 0$ – коэффициент дифференциальной составляющей.

Дискретная форма ПИД-регулятора. В цифровых системах управления ПИД-регулятор реализуется в дискретном времени. Пусть $t_k = kh$, где $h > 0$ – шаг дискретизации, $k = 0, 1, 2, \ldots$

Обозначим

– $e_k = e(t_k) = x_{\text{ref}} - x_k$,

– $u_k = u(t_k)$, тогда дискретный ПИД-регулятор может быть записан в позиционной форме

$$u_k = K_p e_k + K_i h \sum_{i=0}^{k} e_i + K_d \frac{e_k - e_{k-1}}{h},$$

где сумма аппроксимирует интеграл, а конечная разность – производную.

Альтернативно, используется приращённая форма (velocity form), удобная для реализации в микроконтроллерах

$$\begin{aligned} \Delta u_k &= u_k - u_{k-1} \\ &= K_p(e_k - e_{k-1}) + K_i h\, e_k + K_d \frac{e_k - 2e_{k-1} + e_{k-2}}{h}, \end{aligned}$$

и тогда

$$u_k = u_{k-1} + \Delta u_k.$$

Применение к системе $\dot{x} = y$, $\dot{y} = u$. Рассмотрим задачу стабилизации положения $x_{\text{ref}} = \text{const}$. Тогда

$$\frac{de(t)}{dt} = -y(t).$$

Подставляя $u(t)$ в динамику системы (2.28), учитывая, что

$$\dot{e}(t) = \dot{x}_{\text{ref}}(t) - y(t), \quad \ddot{e}(t) = \ddot{x}_{\text{ref}}(t) - u(t),$$

получаем уравнение замкнутой системы:

$$\ddot{e}(t) + K_d\dot{e}(t) + K_p e(t) + K_i \int_0^t e(\tau)\,d\tau = \ddot{x}_{\text{ref}}(t)$$

или

$$u(t) = K_p(x_{\text{ref}} - x(t)) + K_i \int_0^t (x_{\text{ref}} - x(\tau))\,d\tau - K_d y(t).$$

В случае стабилизации в точке ($x_{\text{ref}} = \text{const}$) правая часть обращается в ноль. Дифференцируя уравнение, приходим к линейному дифференциальному уравнению третьего порядка

$$e'''(t) + K_d e''(t) + K_p e'(t) + K_i e(t) = 0.$$

Устойчивость и качество переходного процесса определяются расположением корней характеристического уравнения

$$s^3 + K_d s^2 + K_p s + K_i = 0.$$

Коэффициенты K_p, K_i, K_d подбираются (например, методом Зиглера–Николса, корней характеристического уравнения или с использованием программной настройки) для обеспечения устойчивости и требуемых динамических характеристик.

Таким образом, управление зависит от текущего отклонения (пропорциональная часть), накопленной ошибки (интегральная часть – устраняет статическую ошибку), скорости изменения положения (дифференциальная часть – демпфирует колебания). Система $\dot{x} = y$, $\dot{y} = u$ является классическим объектом для применения ПИД-регулятора. Она демонстрирует, как простой, но эффективный алгоритм управления может обеспечить стабильное и точное поведение динамической системы. Эта связь делает задачу важной не только в теории автоматического управления, но и в практической инженерной деятельности, особенно в робототехнике, где ПИД-регуляторы до сих пор составляют основу систем управления движением.

Отметим некоторые недостатки ПИД-регулятора:

– интегральное насыщение (wind-up) – при больших ошибках интегральная часть может накапливаться чрезмерно, для решения проблемы можно ограничивать интеграл или использовать анти-wind-up алгоритмы;

– фильтрация производной – для уменьшения влияния шума производная фильтруется

$$\frac{de}{dt} \approx \frac{s}{\tau s + 1} e(t), \quad \tau > 0;$$

– задание производной по выходу – чтобы избежать «бросков» при изменении задания, дифференциальная часть вычисляется по $x(t)$, а не по $e(t)$

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

В операторной форме (по Лапласу), ПИД-регулятор имеет передаточную функцию

$$C(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s = K_d s + K_p + \frac{K_i}{s}.$$

Это – несобственный фильтр (степень числителя $>$ знаменателя), поэтому в реальных системах дифференциальная часть «заводится» через фильтр низких частот

$$C_{\text{real}}(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{\tau s + 1}, \quad \tau \ll 1.$$

Таким образом, при применении к системе $\dot{x} = y$, $\dot{y} = u$, ПИД обеспечивает полную обратную связь по состоянию (через e, $\int e$, $\dot{e}$), что делает его эффективным инструментом для стабилизации и слежения. Несмотря на простоту, ПИД-регулятор требует аккуратной настройки коэффициентов и учёта практических ограничений при реализации.

Приведем математическая реализацию ПИД-регулятора для системы второго порядка. Рассмотрим систему в замкнутом виде. Подставим $u(t)$ в исходную систему

$$\begin{cases} \dot{x}(t) = y(t), \\ \dot{y}(t) = K_p(x_{\text{ref}} - x(t)) + K_i \int_0^t (x_{\text{ref}} - x(\tau))\, d\tau - K_d y(t). \end{cases}$$

Введём дополнительную переменную

$$z(t) = \int_0^t e(\tau)\, d\tau = \int_0^t (x_{\text{ref}} - x(\tau))\, d\tau,$$

тогда $\dot{z}(t) = e(t) = x_{\text{ref}} - x(t)$.

Теперь система становится автономной и записывается в виде трёх дифференциальных уравнений

$$\begin{cases} \dot{x}(t) = y(t), \\ \dot{y}(t) = K_p(x_{\text{ref}} - x(t)) + K_i z(t) - K_d y(t), \\ \dot{z}(t) = x_{\text{ref}} - x(t). \end{cases}$$

Рассмотрим отклонение от положения равновесия. При $x(t) \to x_{\text{ref}}$, $y(t) \to 0$, $z(t) \to z^*$ (константа). Положим $\tilde{x}(t) = x(t) - x_{\text{ref}}$, тогда

$$e(t) = -\tilde{x}(t), \quad \dot{z}(t) = -\tilde{x}(t).$$

В отклонениях система принимает вид

$$\begin{cases} \dot{\tilde{x}} = y, \\ \dot{y} = -K_p\tilde{x} + K_i z - K_d y, \\ \dot{z} = -\tilde{x}. \end{cases}$$

Матрица системы

$$\dot{\mathbf{X}} = A\mathbf{X}, \quad \mathbf{X} = \begin{bmatrix} \tilde{x} \\ y \\ z \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 & 0 \\ -K_p & -K_d & K_i \\ -1 & 0 & 0 \end{bmatrix}.$$

Устойчивость определяется собственными значениями матрицы A. Характеристическое уравнение имеет вид

$$\det(sI - A) = \begin{vmatrix} s & -1 & 0 \\ K_p & s + K_d & -K_i \\ s & 0 & s \end{vmatrix} = s \cdot \begin{vmatrix} s & -1 \\ K_p & s + K_d \end{vmatrix} + (-K_i) \cdot \begin{vmatrix} s & -1 \\ s & 0 \end{vmatrix}$$

После вычислений получаем

$$\det(sI - A) = s^3 + K_d s^2 + K_p s + K_i.$$

Для асимптотической устойчивости все корни этого уравнения должны иметь отрицательные вещественные части. Это обеспечивается при выполнении условий критерия Гурвица

$$K_p > 0, \quad K_i > 0, \quad K_d > 0, \quad K_d K_p > K_i.$$

Например, при выборе $K_p = 4, \quad K_d = 3, \quad K_i = 2$, характеристическое уравнение

$$s^3 + 3s^2 + 4s + 2 = 0.$$

имеет корни $s_1 \approx -1$, $s_2, s_3 \approx -1 \pm i$, все они лежат в левой полуплоскости. Следовательно, система устойчива.

Математическая реализация ПИД-регулятора для системы второго порядка сводится к введению интегральной переменной и анализу устойчивости замкнутой системы третьего порядка. Успешное управление возможно при корректном выборе коэффициентов K_p, K_i, K_d, удовлетворяющих условиям устойчивости. Представленная формализация позволяет не только реализовать регулятор на практике, но и провести строгий анализ его динамических свойств.

В робототехнике ПИД-регуляторы играют ключевую роль в точном управлении движением и положением роботов. Их основные задачи заключаются в обеспечении устойчивого удержания заданного положения, скорости или траектории движения благодаря постоянному анализу ошибки – разницы между фактическими показаниями датчиков и желаемыми значениями.

Применения ПИД-регуляторов в робототехнике включают:

– управление положением и ориентацией робота или его частей (например, манипуляторов) за счет плавного и точного коррекционного воздействия;

– стабилизация и регулировка скорости двигателя постоянного тока, бесщеточных или асинхронных двигателей с помощью управляющего сигнала, формируемого ПИД-контроллером;

– управление движением роботов по линиям, где ПИД-регулятор обеспечивает корректное следование по траектории, минимизируя ошибки отклонения и предотвращая колебания;

– управление беспилотными летательными аппаратами (дронами), где стабилизация пространственной ориентации и динамики полёта происходит путем регулирования тяги двигателей;

– реализация управляющих алгоритмов на микроконтроллерах, которые обрабатывают данные с датчиков, рассчитывают ошибку и формируют управляющий сигнал для исполнительных механизмов робота.

Важные особенности ПИД-регулятора в робототехнике:

1. пропорциональная составляющая (P) зависит от текущей ошибки, то есть разницы между заданным значением и текущим измерением, чем больше ошибка, тем сильнее управляющее воздействие;

2. интегральная составляющая (I) помогает устранить долговременные отклонения, накапливая ошибку и увеличивая управляющее воздействие, ес-

ли робот долго находится в состоянии ошибки;

3. дифференциальная составляющая (D) предотвращает перерегулирование и колебания, особенно на высоких скоростях, учитывает скорость изменения ошибки, предсказывает будущее поведение системы и сглаживает отклонения, улучшая стабильность движения;

Если отключить одну или несколько составляющих, можно получить упрощенные версии регуляторов: пропорциональный (П), пропорционально-интегральный (ПИ), пропорционально-дифференциальный (ПД) и т.д.

ПИД-алгоритмы демонстрируют эффективность, простоту реализации и универсальность, позволяя применять их для широкого спектра задач управления роботами. В робототехнике они обеспечивают высокую точность, устойчивость и гибкость управления движением и положением, что делает их одним из базовых инструментов автоматизации и управления в этой области.

Приведем еще несколько примеров применения ПИД-регуляторов в робототехнике.

– Мобильные роботы. Управление движением к целевой координате по линии: ошибка $e(t)$ – отклонение от заданного положения, $u(t)$ – команда на изменение скорости (через ускорение).

– Дроны. Стабилизация высоты: если $x(t)$ – высота, $y(t)$ – вертикальная скорость, $u(t)$ – тяга (пропорциональная ускорению), то ПИД корректирует ускорение по высоте для удержания уровня.

– Роботы-манипуляторы. Позиционирование звеньев: если $x(t)$ – угловое положение, $y(t)$ – угловая скорость, а $u(t)$ – момент, то ПИД-регулятор обеспечивает точное достижение заданного угла.

Как и любой метод ПИД-регулятор имеет свои преимущества и недостатки. К преимуществам следует отнести

– простоту понимания и настройки – даже инженер без глубоких знаний теории управления может настроить ПИД по эмпирическим методам (например, метод Зиглера–Николса);

– высокую эффективность для линейных систем – особенно для объектов второго порядка, таких как $\dot{x} = y$, $\dot{y} = u$;

– широкую реализуемость – может быть реализован как на аналоговой электронике, так и в цифровых контроллерах (микроконтроллерах);

– работу в реальном времени – вычисления просты и требуют минимальных ресурсов.

Из недостатков следует отметить

– возможность интегрального насыщения (wind-up) при больших ошибках;

– не учитывает ограничения на управление $|u(t)| \leq u_{\max}$;

– не подходит для сильно нелинейных или многомерных систем без дополнительной линеаризации;

– требует ручной или автоматической настройки коэффициентов;

– чувствителен к шуму в измерениях (особенно дифференциальная часть).

В таких случаях применяются более сложные методы такие как LQR, MPC, нелинейные наблюдатели, обучение с подкреплением. Однако даже в этих системах ПИД-регулятор часто используется как базовый контроллер или элемент архитектуры.

Связь между принципом максимума и ПИД-регулятором. С математической точки зрения, система (2.1) демонстрирует такие важные свойства, как полная управляемость, минимальная фазовость и возможность синтеза оптимальных решений с помощью принципа максимума Понтрягина. Это делает её идеальной моделью для обучения и тестирования новых алгоритмов.

ПИД-регулятор нужен, потому что он эффективно решает задачи стабилизации и слежения, устраняет статическую ошибку и демпфирует колебания, прост в реализации и настройке, широко применяется в промышленности и технике. Несмотря на свою простоту, ПИД остаётся «рабочей лошадкой» автоматики и составляет основу большинства систем управления в реальном мире. Его понимание и умение настраивать – обязательный навык для специалиста в области управления, робототехники и автоматизации.

Хотя принцип максимума и ПИД-регулятор основаны на разных подходах, связь между ними может быть установлена на уровне целей управления: принцип максимума дает оптимальную траекторию при заданном критерии (например, быстродействие или минимум энергии). ПИД-регулятор применяется как приближённая реализация оптимального управления в окрестности рабочей точки, особенно если оптимальный закон гладкий.

Например, в задачах слежения за траекторией $x_{ref}(t)$, оптимальное управление (по Понтрягину) может быть аппроксимировано с помощью ПИД, если отклонения малы. Коэффициенты ПИД-регулятора можно настраивать на основе оптимального закона управления, полученного из принципа максимума (например, через линеаризацию и LQR, который близок к принципу максимума в линейно-квадратичном случае).

Принцип максимума позволяет найти оптимальное управление для данной задачи, часто в виде релейного или линейного закона обратной связи. ПИД-регулятор – это практический способ стабилизации, который можно рассмат-

ривать как аппроксимацию оптимального управления в окрестности целевого состояния.

При квадратичном функционале и линейной системе оптимальное управление (по принципу максимума) является линейной функцией состояния, что аналогично пропорционально-дифференциальной части ПИД. Интегральная часть ПИД может быть обоснована с точки зрения компенсации ошибки или расширения системы (введение интегратора), что также можно учесть в функционале качества.

Таким образом, ПИД-регулятор можно рассматривать как эмпирическую реализацию идей оптимального управления, а принцип максимума даёт теоретическую основу для синтеза более совершенных законов, которые могут включать или улучшать ПИД-структуру.

2.8 Выводы

В диссертационной работе рассмотрены задачи оптимального управления с фазовыми ограничениями глубины 2 и предложена численная процедура для вычисления их точных экстремалей с заданной точностью. Результаты этой процедуры, примененные к задаче поиска кратчайшей кривой второго порядка, показывают, что теоретические результаты полностью подтверждаются результатами проведенных экспериментов.

Другой интересный вывод, полученный в результате вычислений, заключается в следующем наблюдении, касающемся задачи о роботе при строго выпуклом допустимом наборе управления. Как известно, при фазовом ограничении первого порядка скорость в таких задачах непрерывна, а ускорение, как правило, разрывно в точках стыка. В задаче второго порядка наблюдается, что и скорость, и ускорение непрерывны, но толчок разрывен в таких точках. Тогда можно ожидать, что в следующей задаче с фазовыми ограничениями глубины 3, скорость, ускорение и толчок будут непрерывными, в то время как связка будет разрывной. В этой связи можно предложить некоторое общее правило в качестве гипотезы, справедливой в произвольном k-порядке: первые k производных от x непрерывны, в то время как $(k+1)$-я производная может демонстрировать скачки в точках попадания на границу фазового ограничения или выхода с нее.

Важным выводом главы 2 является практическая применимость принципа максимума для задач с динамикой управления ньютоновского типа. Принцип максимума – отличный рабочий инструмент для получения оптимальных траекторий. При решении практических задач робототехники принцип максимума используется для построения точной траектории и нахождения управле-

ния, которое можно аппроксимировать другими методами, особенно в линейно-квадратичной постановке, например, ПИД-регулятором, что позволяет использовать идеи оптимальности для обоснованной настройки коэффициентов ПИД.

Таким образом, получаем новый способ исследования задач оптимального управления – объединение теортетического, оптимизационного принципа максимума и практического, инженерного ПИД-регулятора. именно такая идея использована при построении алгоритмов для решения задачи управления мобильным роботом в реальном времени, см. главу 4.

Глава 3

Принцип разделения траектории для задачи оптимального быстродействия в системе с обратной связью

В современной робототехнике задача навигации автономных мобильных роботов является одной из ключевых. Эффективное и надёжное перемещение робота из начальной точки в конечную в условиях неопределённости и наличия препятствий требует сочетания точного планирования траектории, учёта динамики системы и устойчивости к ошибкам измерений.

На сегодняшний день существует множество подходов к решению задачи навигации. Среди них – спутниковые системы позиционирования (GPS), радиомаяки, одометрия, лидары и методы одновременной локализации и построения карты (SLAM). Однако каждый из этих методов имеет свои ограничения. Например, GPS недоступен в помещениях и характеризуется невысокой точностью (в среднем 15 м), а визуальная одометрия и SLAM-системы, такие как ORB-SLAM2 [232], чувствительны к условиям освещения и отсутствию характерных ориентиров [180, 181]. Альтернативные решения, включая инерциальные и тепловые системы [182, 183], а также радиолокационные датчики [184], развиваются, но также сталкиваются с трудностями в сложных условиях.

В данной главе диссертационной работы рассматривается задача оптимального управления мобильным роботом, движущимся в плоскости $\mathbb{R}^2$ от заданной начальной точки к терминальной, с учётом фазовых ограничений и наличия круговых препятствий. Предполагается, что робот получает информацию о сво-

ём положении от системы позиционирования с известной погрешностью $\varepsilon > 0$, причём измерения поступают с фиксированным интервалом τ. Важно подчеркнуть, что неточность измерений может привести к отклонению от рассчитанной траектории и, как следствие, к промаху по цели.

Для построения оптимальной по времени траектории используется принцип разделения допустимых траекторий, предложенный В.А. Березневым [185]. Согласно этому принципу, оптимальный путь состоит из отрезков прямых и дуг окружностей, касающихся границ препятствий. На основе этих элементов строится связный граф, в котором вершины соответствуют точкам старта, финиша и касания, а рёбра — возможным участкам движения. Кратчайший путь на этом графе, определяемый, например, алгоритмом Дейкстры [121], даёт оптимальную траекторию в отсутствие возмущений.

Однако в реальных условиях движение робота сопровождается внешними и внутренними возмущениями, из-за которых он может отклоняться от заданной траектории. В связи с этим в работе представлен новый алгоритм коррекции траектории, который гарантирует достижение конечной точки с погрешностью, не превышающей 2ε, при любых ошибках измерений, ограниченных ε. Данный результат является теоретически обоснованным и подтверждён доказательством в разделе 3.2.

Таким образом, основной вклад статьи заключается в разработке и обосновании алгоритма коррекции, сочетающего простоту реализации с гарантией точности приближения к целевой точке в условиях неточной навигации.

3.1 Введение

Разделение траектории мобильного робота на участки движения по прямым отрезкам и дугам окружностей является ключевым элементом предлагаемого подхода к задаче оптимального управления. Данный метод не является эвристическим приёмом, а опирается на строгие теоретические основы, включая принцип разделения допустимых траекторий [185] и теорию оптимального управления для систем второго порядка [10, 18].

Реальный мобильный робот, особенно с автомобильным или дифференциальным приводом, не способен выполнять произвольные маневры. Его движение ограничено минимальным радиусом поворота, максимальной скоростью и ускорением. Движение по прямой и по дуге окружности фиксированного радиуса представляет собой физически реализуемые режимы движения, которые могут быть точно воспроизведены системой управления. Следовательно, построение траектории из таких элементов гарантирует её исполнимость на практике.

Принцип разделения допустимых траекторий [185] утверждает, что в задаче движения из начальной точки в конечную при наличии круговых препятствий оптимальная по времени траектория состоит из

– прямых отрезков, соединяющих точки касания с препятствиями;

– дуг окружностей, совпадающих с границами препятствий.

Это позволяет сформировать связный граф $\Gamma(S, V)$, вершины V которого соответствуют начальной и конечной точкам, а также точкам касания прямых с окружностями-препятствиями.

Ребра S графа – это либо прямые участки, либо дуги окружностей. Длина каждого ребра определяется временем прохождения, вычисленным с учётом динамики робота (ускорение, ограничение скорости). На рис. 3.1 приведен пример графа.

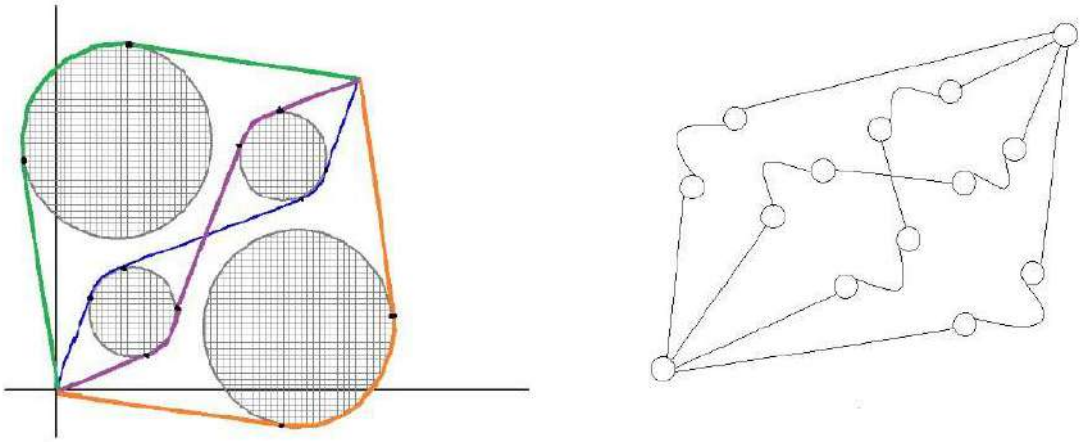

Рис. 3.1: Пример графа после разбиения пути на участки

В итоге задача поиска оптимальной траектории сводится к классической задаче поиска кратчайшего пути на графе, для решения которой могут быть применены эффективные алгоритмы, такие как алгоритм Дейкстры [121].

Поскольку каждый элемент траектории (прямая или дуга) соответствует оптимальному режиму управления в отсутствие возмущений, построенная траектория является глобально оптимальной по времени при заданных фазовых ограничениях. Это особенно важно для систем реального времени, где требуется не просто достижение цели, а достижение её за минимальное время.

Одним из главных вкладов данной работы является алгоритм коррекции траектории, учитывающий погрешность системы позиционирования. Разделение траектории на простые геометрические элементы делает возможным эффективный мониторинг отклонения робота от заданного пути.

На прямолинейных участках боковое отклонение $z_2(t)$ легко контролируется. При получении данных от системы позиционирования с известной погрешностью ε, алгоритм принимает решение о необходимости коррекции (см. алгоритм в разделе 3.2). Если текущее измерение $y(t_i)$ и прогноз движения за интервал τ указывают на возможный выход за пределы допустимой полосы, активируется корректирующее управление.

Как показано в Теореме 4, такой подход гарантирует, что истинное положение робота будет удовлетворять условию $|z_2(t)| < 2\varepsilon$ на всём протяжении движения, что обеспечивает надёжное попадание в окрестность конечной точки.

Таким образом, разделение траектории на прямые и дуги окружностей соответствует физическим возможностям робота, позволяет свести задачу к поиску кратчайшего пути на графе, гарантирует оптимальность по времени, обеспечивает основу для разработки эффективного алгоритма коррекции, дает теоретическую возможность доказать достижение цели с заданной точностью.

Без такого структурированного представления траектории невозможно было бы построить управление, сочетающее оптимальность, реализуемость и устойчивость к ошибкам измерений.

3.2 Принцип разделения траекторий

Традиционным подходом к решению этой задачи является использование методов оптимального управления, основанных на принципе максимума Л. С. Понтрягина (см. [10, 18]).

Поведение робота, который должен переместиться из некоторой точки $x(0) = x^0$ в терминальную точку $x(T) = \hat{x}$ за минимальное время T, описывается дифференциальными уравнениями второго порядка

$$\ddot{x}(t) = f(t, x(t), \dot{x}(t), u(t)), \tag{3.1}$$

где $x(t) \in \mathbb{R}_2$ - искомая оптимальная траектория, а $u(t) \in \mathbb{R}_2$ - допустимый управляющий параметр, который подчиняется условию

$$u^- \leq u(t) \leq u^+, \quad u^-, u^+ \in \mathbb{R}_2. \tag{3.2}$$

Предполагается, что компонента $u_1(t)$ задаёт ускорение (положительное при наборе скорости и отрицательное при торможении), а $u_2(t)$ - положение руля робота. Естественно также предполагать, что скорость робота ограничена, т.е. для любого t выполняются неравенства

$$0 \leq \dot{x}(t) \leq \hat{v}. \tag{3.3}$$

Условие непересечения траектории робота с круговыми препятствиями означает, что для любых t кривая $x(t)$ должна удовлетворять неравенству

$$\|x(t) - C_k\| \geq r_k^+, \quad k \in \overline{1, K}, \tag{3.4}$$

где K - общее число препятствий, C_k - координаты центра k-го препятствия. [1]

Эти условия означают, что область допустимых траекторий робота не является выпуклой. По этой причине использование методов оптимального управления, а также методов математического программирования (см., например, [118], [178]) становится весьма трудоемким.

Предлагаемые результаты основаны на хорошо известной теории синтеза оптимальных управлений в системах второго порядка (см., например, [18]), а также на методе построения кратчайшего пути на связном графе. В частности, предлагаемый подход заключается в следующем.

Окружности, которые являются границами круговых препятствий, снабжены некоторыми точками, объявленными вершинами $v_i, \quad i \in \overline{1, n}$ связного графа $\Gamma(S, V)$, где $V = \{v_i\}$ - множество вершины графа, и $S = \{s_{ij}\}$ - множество ребер $i, j \in \overline{1, n}$. Длина ребра $s_{ij} \in S$ определяется временем прохождения робота из вершины v_i в вершину v_j.

В диссертационной работе рассмотрен линейный случай, когда $\ddot{x}(t) = u_1(t)$. При этом предполагается, что на отрезке $[v_i, v_j]$ препятствий нет. В противном случае считаем, что длина ребра графа $s_{ij} = +\infty$. Кроме того, будем предполагать, что в точке v_i объект начинает движение с некоторой скоростью $\dot{x} = q_i > 0$, а в конечной точке v_j должен достичь скорости $\dot{x}t = q_j > 0$. В этих случаях, как следует из [18], управляемый объект принадлежит классу, для которого принцип максимума оказывается не только необходимым, но и достаточным условием оптимальности и, кроме того, верно утверждение, что (см. [18], стр. 282) каждый элемент управления, который выполняет переход от любой начальной точки к любой конечной точке, принимает только значения $u_1(t) = u_1^-$ или $u_1(t) = u_1^+$ и имеет не более одного переключения.

[1] мы допускаем в (3.4) нестрогое неравенство, поскольку в практических расчётах полагаем k $r_k^+ = r_k + \rho$, где r_k - фактический радиус k-го кругового препятствия, а $\rho > 0$ - минимальная величина, обеспечивающая движение робота без касания препятствия с учетом его габаритов.

В связи с этим уместно сделать одно важное замечание. Утверждение о количестве переключений управления, сделано при отсутствии каких-либо ограничений сверху на $\dot{x}(t)$. Однако очевидно, что если существует ограничение (3.3), то в зависимости от значения u_1^+ и длины ребра s_{ij} возможны одно или два переключения со значениями $u_1^+, 0, u_1^-$.

Обратимся к вопросу о назначении вершин. Для их построения используется естественное требование гладкости траектории робота. Это означает, что бход роботом кругового препятствия должен начинаться и заканчиваться в точках касания с окружностями, являющимися границами препятствий. К числу вершин (отличных от точек касания) естественно должны быть добавлены начальная $v_0 = x^0$ и конечная $v_n = \hat{x}$ точки траектории.

Пусть $A(x_{01}, x_{02})$ - произвольная точка на плоскости, а $C(x_{j1}, x_{j2})$ - заданная точка, которая является центром j-го кругового препятствия (рис. 3.2). Тогда очевидно, что координаты точек касания являются решениями нелинейной системы уравнений

$$\begin{cases} (x_{01} - x_1)(x_{j2} - x_2) + (x_{02} - x_2)(x_{j1} - x_1) = 0, \\ (x_1 - x_{j1})^2 + (x_2 - x_{j2})^2 = r_j^2, \end{cases} \tag{3.5}$$

где r_j - радиус j-го кругового препятствия.

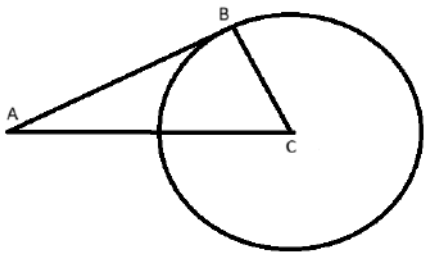

Рис. 3.2: Расположение точек A и C.

Исключим из системы уравнений (3.5) переменную x_2 с целью получить уравнение относительно x_1. Раскроем первое уравнение в (3.5)

$$(x_{01} - x_1)(x_{j2} - x_2) = x_{01}x_{j2} - x_{01}x_2 - x_1x_{j2} + x_1x_2,$$

$$(x_{02} - x_2)(x_{j1} - x_1) = x_{02}x_{j1} - x_{02}x_1 - x_2x_{j1} + x_2x_1.$$

Сложим и приравняем нулю

$$\begin{aligned} &x_{01}x_{j2} + x_{02}x_{j1} - \\ &- x_{01}x_2 - x_2x_{j1} - \\ &- x_1x_{j2} - x_{02}x_1 + \\ &+ x_1x_2 + x_2x_1 = 0. \end{aligned}$$

Заметив, что $x_1x_2 + x_2x_1 = 2x_1x_2$, сруппируем

$$x_{01}x_{j2} + x_{02}x_{j1} - (x_{j2} + x_{02})x_1 - (x_{01} + x_{j1})x_2 + 2x_1x_2 = 0.$$

Обозначим

$$\begin{aligned} A &= x_{01}x_{j2} + x_{02}x_{j1}, \\ B &= -(x_{j2} + x_{02}), \\ C &= -(x_{01} + x_{j1}), \end{aligned}$$

тогда первое уравнение в (3.5) принимет вид

$$A + Bx_1 + Cx_2 + 2x_1x_2 = 0. \tag{3.6}$$

Раскроем второе уравнение в (3.5)

$$\begin{gathered} (x_1 - x_{j1})^2 + (x_2 - x_{j2})^2 = r_j^2 \\ x_1^2 - 2x_{j1}x_1 + x_{j1}^2 + x_2^2 - 2x_{j2}x_2 + x_{j2}^2 = r_j^2 \\ x_1^2 + x_2^2 - 2x_{j1}x_1 - 2x_{j2}x_2 + \underbrace{x_{j1}^2 + x_{j2}^2 - r_j^2}_{E} = 0, \end{gathered} \tag{3.7}$$

где $E = x_{j1}^2 + x_{j2}^2 - r_j^2$.
Из уравнения (3.6) получаем

$$A + Bx_1 + x_2(C + 2x_1) = 0 \quad \Rightarrow \quad x_2 = -\frac{A + Bx_1}{C + 2x_1}. \tag{3.8}$$

Подставим (3.8) в (3.7)

$$x_1^2 + \left(-\frac{A + Bx_1}{C + 2x_1}\right)^2 - 2x_{j1}x_1 - 2x_{j2}\left(-\frac{A + Bx_1}{C + 2x_1}\right) + E = 0.$$

и упростим

$$x_1^2 + \frac{(A + Bx_1)^2}{(C + 2x_1)^2} - 2x_{j1}x_1 + \frac{2x_{j2}(A + Bx_1)}{C + 2x_1} + E = 0.$$

Умножим обе части на $(C+2x_1)^2$, чтобы избавиться от знаменателей

$$\begin{aligned} & x_1^2(C+2x_1)^2 \\ &+(A+Bx_1)^2 \\ &-2x_{j1}x_1(C+2x_1)^2 \\ &+2x_{j2}(A+Bx_1)(C+2x_1) \\ &+E(C+2x_1)^2=0. \end{aligned}$$

Раскроем скобки

1. $x_1^2(C+2x_1)^2$**:**

$$(C+2x_1)^2 = C^2+4Cx_1+4x_1^2,$$
$$x_1^2(C^2+4Cx_1+4x_1^2) = C^2x_1^2+4Cx_1^3+4x_1^4.$$

2. $(A+Bx_1)^2$**:**

$$= A^2+2ABx_1+B^2x_1^2.$$

3. $-2x_{j1}x_1(C+2x_1)^2$**:**

$$= -2x_{j1}x_1(C^2+4Cx_1+4x_1^2) = -2x_{j1}C^2x_1-8x_{j1}Cx_1^2-8x_{j1}x_1^3.$$

4. $2x_{j2}(A+Bx_1)(C+2x_1)$**:**

$$(A+Bx_1)(C+2x_1) = AC+2Ax_1+BCx_1+2Bx_1^2 = AC+(2A+BC)x_1+2Bx_1^2,$$

$$\text{умножаем на } 2x_{j2}: \quad 2x_{j2}AC+2x_{j2}(2A+BC)x_1+4x_{j2}Bx_1^2.$$

5. $E(C+2x_1)^2$**:**

$$= E(C^2+4Cx_1+4x_1^2) = EC^2+4ECx_1+4Ex_1^2.$$

Соберем коэффициенты при степенях x_1

При x_1^4**:**

$$4$$

При x_1^3**:**

$$4C-8x_{j1}$$

При x_1^2:

$$C^2 + B^2 - 8x_{j1}C + 4x_{j2}B + 4E$$

При x_1:

$$2AB - 2x_{j1}C^2 + 2x_{j2}(2A + BC) + 4EC$$

Свободный член:

$$A^2 + 2x_{j2}AC + EC^2$$

Таким образом, после подстановки и упрощений получаем уравнение четвёртой степени относительно x_1

$$ax_1^4 + bx_1^3 + cx_1^2 + dx_1 + e = 0,$$

где:

$$\begin{aligned} a &= 4, \\ b &= 4C - 8x_{j1}, \\ c &= C^2 + B^2 - 8x_{j1}C + 4x_{j2}B + 4E, \\ d &= 2AB - 2x_{j1}C^2 + 2x_{j2}(2A + BC) + 4EC, \\ e &= A^2 + 2x_{j2}AC + EC^2, \end{aligned} \tag{3.9}$$

и

$$\begin{aligned} A &= x_{01}x_{j2} + x_{02}x_{j1}, \\ B &= -(x_{j2} + x_{02}), \\ C &= -(x_{01} + x_{j1}), \\ E &= x_{j1}^2 + x_{j2}^2 - r_j^2. \end{aligned} \tag{3.10}$$

Из приведенных вычислений видно, что в самом общем случае, когда $x_{01} \neq x_{j1}$ и $x_{02} \neq x_{j2}$ решениями системы (3.5) являются точки (x_1, x_2), где составляющая x_1 является решением уравнений четвертой степени $ax_1^4+bx_1^3+cx_1^2+dx_1+e = 0$, с коэффициентами, найденными по формулам (3.9) и (3.10).

Задача значительно упрощается, если рассмотреть частный случай

$$x_{02} = x_{j2}.$$

Подставим условие $x_{02} = x_{j2}$ в систему (3.5). Обозначим

$$x_{02} = x_{j2} = \alpha.$$

Тогда первое уравнение в (3.5) имеет вид

$$(x_{01} - x_1)(\alpha - x_2) + (\alpha - x_2)(x_{j1} - x_1) = 0.$$

Заметим, что оба слагаемых содержат множитель $\alpha - x_2$. Вынесем его за скобки

$$(\alpha - x_2)\left[(x_{01} - x_1) + (x_{j1} - x_1)\right] = 0.$$

Упростим выражение в скобке

$$x_{01} - x_1 + x_{j1} - x_1 = x_{01} + x_{j1} - 2x_1.$$

Таким образом, первое уравнение в (3.5) принимает вид

$$(\alpha - x_2)(x_{01} + x_{j1} - 2x_1) = 0.$$

Это уравнение выполняется, если хотя бы один из множителей равен нулю

$$\alpha - x_2 = 0 \quad \text{или} \quad x_{01} + x_{j1} - 2x_1 = 0,$$

то есть

$$x_2 = \alpha = x_{j2} \quad \text{или} \quad x_1 = \frac{x_{01} + x_{j1}}{2}.$$

Теперь рассмотрим каждый случай отдельно и подставим в уравнение окружности – второе уравнение в (3.5)

$$(x_1 - x_{j1})^2 + (x_2 - x_{j2})^2 = r_j^2. \tag{2}$$

Случай 1: $x_2 = x_{j2}$ Тогда $x_2 - x_{j2} = 0$, и второе уравнение в (3.5) упрощается

$$(x_1 - x_{j1})^2 = r_j^2 \quad \Rightarrow \quad x_1 - x_{j1} = \pm r_j \quad \Rightarrow \quad x_1 = x_{j1} \pm r_j.$$

Таким образом, получаем два решения

$$x_1 = x_{j1} + r_j \quad \text{или} \quad x_1 = x_{j1} - r_j,$$

при условии $x_2 = x_{j2}$.

Это соответствует горизонтальным точкам на окружности (точки на одном уровне с центром по координате x_2).

Случай 2: $x_1 = \dfrac{x_{01} + x_{j1}}{2}$ Обозначим

$$x_1 = \frac{x_{01} + x_{j1}}{2} =: \bar{x}_1.$$

Подставим это значение во второе уравнение в (3.5)

$$(\bar{x}_1 - x_{j1})^2 + (x_2 - x_{j2})^2 = r_j^2.$$

Выразим x_2

$$(x_2 - x_{j2})^2 = r_j^2 - (\bar{x}_1 - x_{j1})^2.$$

Обозначим $\Delta = r_j^2 - (\bar{x}_1 - x_{j1})^2$. Тогда
если $\Delta > 0$, два решения: $x_2 = x_{j2} \pm \sqrt{\Delta}$,
если $\Delta = 0$, одно решение,
если $\Delta < 0$, решений нет.
Таким образом, в этом случае

$$x_1 = \bar{x}_1 = \frac{x_{01} + x_{j1}}{2}, \quad x_2 = x_{j2} \pm \sqrt{r_j^2 - \left(\frac{x_{01} + x_{j1}}{2} - x_{j1}\right)^2}.$$

Упростим выражение в скобке

$$\frac{x_{01} + x_{j1}}{2} - x_{j1} = \frac{x_{01} - x_{j1}}{2}.$$

Тогда

$$x_2 = x_{j2} \pm \sqrt{r_j^2 - \left(\frac{x_{01} - x_{j1}}{2}\right)^2}.$$

Всего может быть до четырёх решений, но они распадаются на два независимых множества

1.
$$x_2 = x_{j2}, \quad x_1 = x_{j1} \pm r_j$$

2.
$$x_1 = \frac{x_{01} + x_{j1}}{2}, \quad x_2 = x_{j2} \pm \sqrt{r_j^2 - \left(\frac{x_{01} - x_{j1}}{2}\right)^2},$$

при условии, что выражение под корнем неотрицательно.

При построении касательных к двум окружностям легко вычислить некоторую точку A, которая является единственной точкой пересечения касательных и прямой, соединяющей центры окружностей. Следовательно, можно вычислить координаты точки A, после чего задача сводится к рассмотренной выше.

Таким образом, любая траектория движения робота представляет собой последовательность участков, некоторые из которых представляют собой прямые участки между двумя вершинами графа без препятствий, а другие – участки движения с постоянной скоростью по дуге окружности. Для любого робота такие траектории образуют конечное множество состояний, из которого можно сделать выбор, чтобы минимизировать время в пути. Для этой цели можно использовать, например, алгоритм Дейкстры [121] построения кратчайшего пути на графе. Оптимальную траекторию, полученную этим методом, можно увидеть на 3.3.

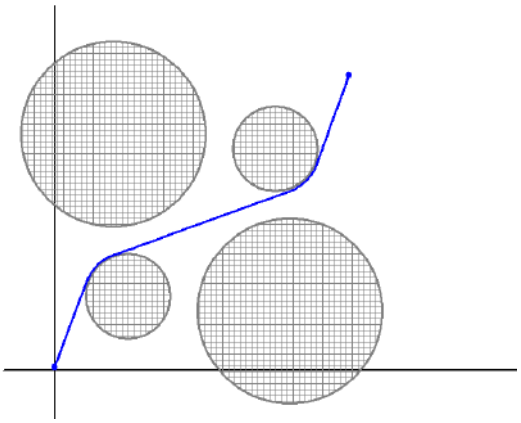

Рис. 3.3: Траектория, полученная предложенным методом

Для сравнения на рис. 3.4 приведен пример траектории, найденной только методом Дейкстры, без разделения траекторий, а также (рис. 3.5) метод Дейкстры с процедурой сглаживания при движении по границе.

Рассмотрим произвольный прямолинейный фрагмент полученной траектории $x(t)$. Не нарушая общности, можно считать, что фрагмент начинается в точке $x^0 = (0, 0)$ и заканчивается в точке $\hat{x} = (\hat{x}_1, 0)$. Будем также считать, что начальное положение робота в точке x^0 известно точно и на всем интервале $[x^0, \hat{x}]$ скорость робота постоянна, т.е. $\dot{x}(t) = v = const$. Будем также предполагать, что управление осуществляется с использованием некоторой системы позиционирования, которая через каждый интервал времени τ передает на управляемый объект его координаты $y(t_i) = (y_1(t_i), y_2(t_i))$ с некоторой малой ошибкой δ_i, причем $|\delta_i| < \varepsilon$, где $\varepsilon > 0$ — известная точность позиционирования.

Через $z(t) = (z_1(t), z_2(t))$ будем обозначать неизвестные текущие координаты робота, т.е. $z(t_i) = y(t_i) + \delta_i$. Координаты $z(t)$ предполагаются неизвестными в силу самых различных причин. Это могут быть как всевозможные внешние воз-

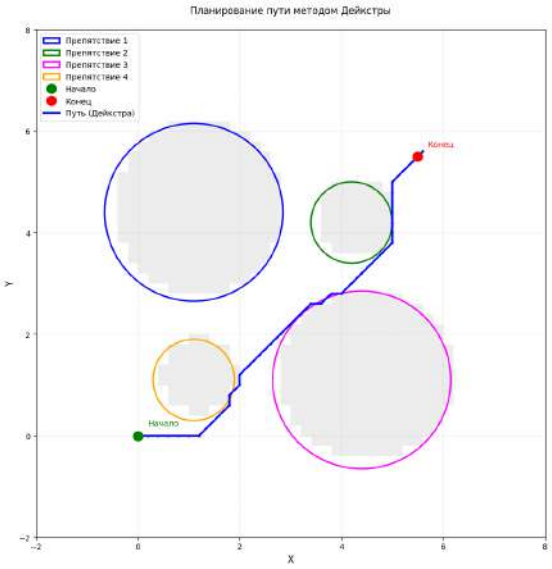

Рис. 3.4: Траектория, полученная методом Дейкстры

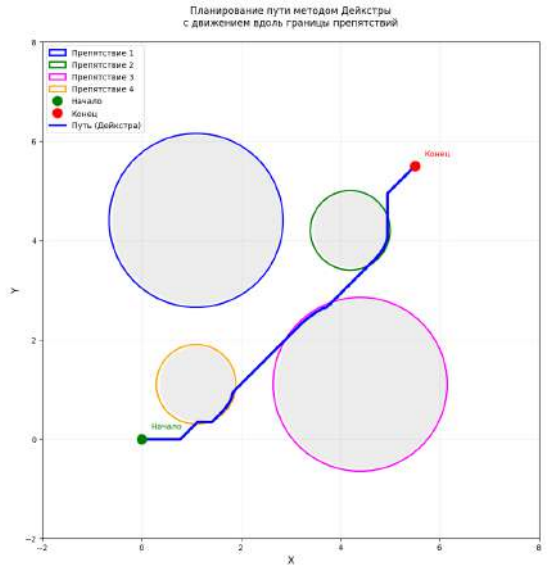

Рис. 3.5: Траектория, полученная методом Дейкстры с процедурой сглаживания

действия на управляемый объект (погодные условия, состояние поверхности, по которой перемещается робот и т.п.), так и технические (конструктивные) особенности самого робота. Иными словами, робот может отклониться от траектории $x(t)$ и двигаться в направлении, при котором он не сможет достичь терминальной точки $\hat{x}$. Следовательно, в какой-то момент его движение должно быть скорректировано таким образом, чтобы можно было гарантировать попадание по крайней мере в некоторую приемлемую окрестность точки

Суть предлагаемого с этой целью алгоритма заключается в следующем. Если передаваемые системой позиционирования координаты $y(t_i)$ не выходят за границы ε—полосы с обеих сторон теоретической траектории робота $x(t)$, нет необходимости в корректировке его движения так как в этом случае есть вероятность, что объект не отклонился от заданной траектории, а уклонение $y(t_i)-x(t_i)$ является следствием погрешности системы позиционирования. Если же с учетом скорости v и интервала времени τ возможен выход точки $y(t_{i+1})$ за пределы указанной полосы, то следует внести корректирующее траекторию движения управление $u = \hat{x} - y(t_i)$. Таким образом, пошаговый алгоритм принимает следующий вид.

Алгоритм

Шаг 1. Положить $i = 0, \quad t_i = 0$. Перейти к шагу 2.

Шаг 2. Положить $i := i + 1, \quad t_i := t_{i-1} + \tau$, принять данные системы позиционирования $y(t_i)$. Перейти к шагу 3.

Шаг 3. Если $|y_2(t_i)| + v\tau < \varepsilon$, то перейти к шагу 2. В противном случае перейти к шагу 4.

Шаг 4. Перенастроить управление роботом на движение вдоль вектора $u = \hat{x} - y(t_i)$. Перейти к шагу 5.

Шаг 5. Если $|y_1(t_i) - \hat{x}_1| < \varepsilon$, то остановиться. В противном случае перейти к шагу 2.

Таким образом, справедлива следующая

Теорема 4 *Пусть управление осуществляется в соответствии с алгоритмом. Тогда для любого t справедливо неравенство $|z_2(t)| < 2\varepsilon$.*

Доказательство. Предположим, что $|z_2(t_k)| \geq 2\varepsilon$ для некоторого k, а для всех $i < k$ справедливо неравенство $|y_2(t_i)| + v\tau < \varepsilon$. Тогда

$$|z_2(t_k)| \leq |z_2(t_{k-1})| + v\tau = |y_2(t_{k-1}) + \delta_{k-1}| + v\tau \leq$$

$$\leq |y_2(t_{k-1})| + |\delta_{k-1}| + v\tau < |y_2(t_{k-1})| + \varepsilon + v\tau < 2\varepsilon.$$

Полученное противоречие доказывает утверждение.

На рис. 3.6 показан пример оптимальной траектории (сплошная линия), построенной с учётом кругового препятствия. Траектория состоит из

– прямого участка от старта до точки касания,

– дуги окружности вдоль границы препятствия,

– прямого участка до конечной точки.

Также на рисунке изображён процесс коррекции при неточных измерениях. Робот движется по прямому участку, и каждые τ секунд получает оценку своего положения $y(t_i)$ с погрешностью $\|\delta_i\| < \varepsilon$. Эти измерения отмечены крестиками.

Зона, в которой отклонение считается допустимым («ε-полоса»), заштрихована серым. Пока измерения остаются внутри полосы, коррекция не требуется. Однако, когда прогнозируемое положение (учитывая скорость v и интервал τ) может выйти за пределы полосы, активируется корректирующее управление $u = \hat{x} - y(t_i)$, и робот начинает двигаться прямо к цели.

Согласно Теореме 4, такой алгоритм гарантирует, что истинное положение робота $z(t)$ всегда удовлетворяет условию $|z_2(t)| < 2\varepsilon$, что обеспечивает попадание в окрестность цели.

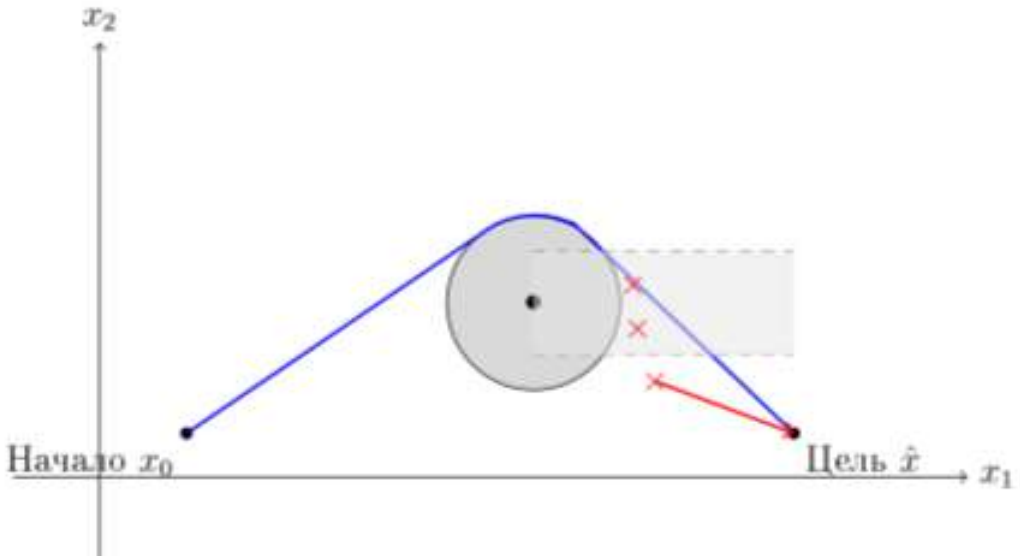

Рис. 3.6: Пример оптимальной траектории при наличии одного фазового ограничения

Результаты численных экспериментов с методом разделения траекторий

В данном разделе приведены результаты тестирования алгоритма метода разделения траекторий. На рис.3.7–3.9 видно изменение траектории в зависимости от расположения препятствий и выбора начальной точки.

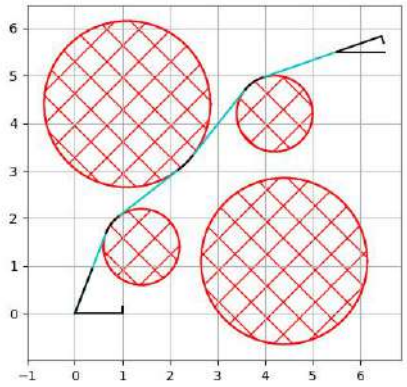

Рис. 3.7: Траектория, полученная методом разделения траекторий

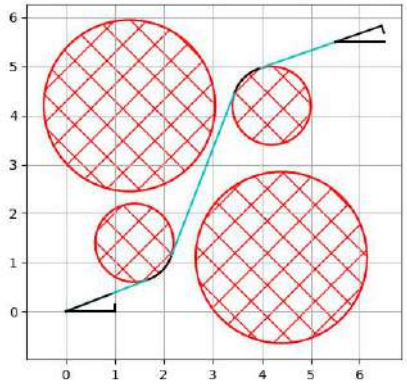

Рис. 3.8: Траектория, полученная методом разделения траекторий при приближении одной из окружностей с меньшим радиусом к окружности с большим радиусом

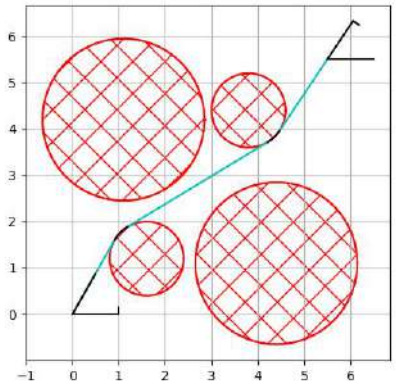

Рис. 3.9: Траектория, полученная методом разделения траекторий при приближении другой окружности с меньшим радиусом к окружности с большим радиусом

В следующем разделе описан предложенный в диссертационной работе метод поиска касательных для использования принципа разделения траекторий.

3.3 Метод поиска касательных в задаче быстродействия для колесного мобильного робота

Описание модели робота и постановка задачи

Кинематическая модель колесного робота В настоящей работе рассматривается движение по горизонтальной плоскости робота с дифференциальным приводом, т.е. с двумя ведущими колесами, которые управляются независимо друг от друга. Схема робота представлена на рис. 3.10, где 1, 2 – правое и левое ведущие колеса робота, 3 – корпус робота, 4 – пассивное рояльное колесо, выполняющее функцию дополнительной опоры для того, чтобы робот имел три точки опоры. Координаты x, y – декартовы координаты центра колесной пары робота, θ – курсовой угол робота; переменные x, y, θ образуют фазовый вектор $\mathbf{x} = (x, y, \theta)^T \in \mathbb{R}^3$. Кинематическая схема, представленная на рис. 3.10, описывается одним геометрическим параметром – расстоянием между ведущими колесами робота b. Значение этого параметра влияет на соотношение усилий, которые требуется приложить к роботу для его продольного движения и для

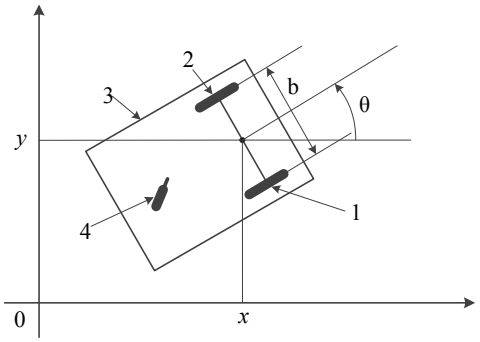

Рис. 3.10: Схема двухколесного робота: 1, 2 – колеса робота, 3 – корпус, 4 – пассивное (рояльное) колесо-подпорка. x и y – декартовы координаты центра колесной пары; θ – курсовой угол робота.

его поворота. Математическая модель, описывающая движение такого робота, имеет следующий вид:

$$\begin{aligned} \dot{x} &= \frac{(u_1 + u_2)}{2}\cos(\theta) \\ \dot{y} &= \frac{(u_1 + u_2)}{2}\sin(\theta) \\ \dot{\theta} &= \frac{(u_1 - u_2)}{b} \end{aligned} \tag{3.11}$$

В качестве управляющих воздействий u_1 и u_2 выбраны скорости соответственно правого и левого колес. Управляющие воздействия образуют управляющий вектор $\mathbf{u} = (u_1, u_2)^T \in \mathbf{U} \subseteq \mathbb{R}^2$, принадлежащий компактной области двумерного пространства, которая определяет ограничения на управляющие воздействия:

$$|u_i| \leq u_{\max}, \quad i = 1, 2. \tag{3.12}$$

Описанная модель часто применяется в задачах, связанных с движением колесных роботов, когда не требуется учитывать переходные процессы, связанные с динамикой робота. В кинематической постановке корпус робота и его колеса считаются невесомыми, и принимается допущение о том, что можно напрямую управлять скоростями вращения колес (т.е. линейными скоростями их центров). Геометрическими размерами робота пренебрегаем, считая робота математической точкой.

Замечание 9 *В тех случаях, когда необходимо учитывать реальные размеры робота, можно для простоты считать, что робот сам ограничен в горизонтальных размерах окружностью некоторого постоянного радиуса r. Все рассуждения, приведенные в данной статье, остаются в силе, если увеличить радиус препятствий на величину r, а робота по-прежнему считать точкой. В случае некруговых препятствий можно расширить границы этих препятствий также на величину r. Такое расширение является частным случаем сложения Минковского для геометрических фигур, представляющих препятствия, и круга, представляющего робота.*

Постановка задачи быстродействия Рассмотрим классическую задачу быстродействия. Пусть математическая модель движения робота задана как (3.11) с ограничениями на управляющие воздействия (3.12).

В начальный момент времени $t^0 = 0$ заданы значения фазовых координат робота:

$$x(0) = x_0; \quad y(0) = y_0; \quad \theta(0) = \theta_0. \tag{3.13}$$

Задано терминальное условие:

$$x(t^f) = x^f; \quad y(t^f) = y^f; \quad \theta(t^f) = \theta^f, \tag{3.14}$$

где t^f – неизвестное заранее время достижения терминального состояния.

Движение робота по плоскости с препятствиями в общем виде можно исследовать только численными методами. Будем считать, что все препятствия в нашей задаче являются круговыми и непересекающимися. Это определяет следующие фазовые ограничения:

$$\|(x(t), y(t)) - \mathbf{C}^k\| \leq r^k, \quad k = 1, \ldots, K, \tag{3.15}$$

где K - количество препятствий. Ограничения (3.15), определяющие область недопустимых значений фазовых переменных, означают, что траектория робота ни в какой момент времени не должна пересекать круги радиусов r^k с центрами $\mathbf{C}^k = (x_c^k, y_c^k)^T$.

Требуется достичь терминального состояния за минимально возможное время с учетом фазовых ограничений (3.15), т.е. критерий качества имеет вид

$$J = \int_0^{t^f} 1dt = t^f \to \min_{\mathbf{u}\in\mathbf{U}} \tag{3.16}$$

Требуется найти функцию $\mathbf{u} = \mathbf{u}(t) \in \mathbf{U}$, которая при ее подстановке в систему (3.11) обеспечивает переход этой системы из начального состояния (3.13) в терминальное (3.14) за минимальное время t^f с учетом фазовых ограничений (3.15).

Ограничения (3.15) означают, что область допустимых траекторий системы не является выпуклой, что сильно затрудняет поиск оптимального решения аналитическими методами или методами математического программирования.

Метод поиска касательных В описанной постановке возможно аналитически найти оптимальную по быстродействию траекторию, которая, как будет показано ниже, совпадает в данном случае с кратчайшим в геометрическом смысле путем из начальной точки в терминальную с учетом ориентации робота в начальном положении и заданной ориентацией этого робота в терминальном положении, причем при оптимальном движении робот будет совершать повороты на месте только в начальной и в терминальной точках. Для поиска такой оптимальной траектории будем использовать метод, описанный далее. Для реализации метода необходимо автоматизировать приведенные ниже вычисления. Чтобы не загромождать формулы, будем использовать одинаковые обозначения для точек и их радиус-векторов. Например, A может обозначать также и радиус-вектор точки A.

В описанной постановке возможно аналитически найти оптимальную по быстродействию траекторию, которая, как будет показано ниже, совпадает в данном случае с кратчайшим в геометрическом смысле путем из начальной точки в терминальную с учетом ориентации робота в начальном положении и заданной ориентацией этого робота в терминальном положении, причем при оптимальном движении робот будет совершать повороты на месте только в начальной и в терминальной точках. Для поиска такой оптимальной траектории будем использовать метод, описанный далее. Для реализации метода необходимо автоматизировать приведенные ниже вычисления. Чтобы не загромождать формулы, будем использовать одинаковые обозначения для точек и их радиус-векторов. Например, A может обозначать также и радиус-вектор точки A.

Нахождение отрезков касательных Для нахождения касательных к окружности, проходящих через заданную точку, воспользуемся следующим построением (рис. 3.11). Считаем, что точка P лежит вне круга радиуса r с центром в точке C. Тогда существует две касательных, проходящих через точку P и касающихся окружности в точках A и B, симметричных относительно прямой, соединяющей центр круга C и точку P. Поскольку треугольник PCA прямоугольный, можно определить косинус и синус угла φ при вершине C как:

$$\begin{aligned} \cos\varphi &= \frac{r}{d}, \quad d = \|PC\| \\ \sin\varphi &= \sqrt{1-\cos^2\varphi} \end{aligned} \tag{3.17}$$

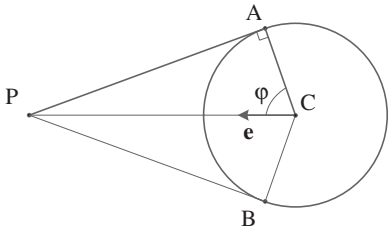

Рис. 3.11: Нахождение касательных к окружности, проходящих через заданную точку. Точки A и B находятся как вершины прямоугольных треугольников PAC и PBC.

Составим матрицу поворота:

$$\mathbf{M} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$$

Тогда точки A и B можно найти как

$$A = C + r\mathbf{M}^T\vec{\mathbf{e}}, \quad B = C + r\mathbf{M}\vec{\mathbf{e}},$$

где вектор $\vec{\mathbf{e}} = \frac{P-C}{d}$.

Аналогичным образом найдем общие касательные к двум окружностям с радиусами r_1, r_2 и центрами в точках C_1, C_2. Считаем, что эти окружности не пересекаются (рис. 3.12). Из построений, показанных на рис. 3.12, видно, что косинусы и синусы углов φ_1 и φ_2 можно найти как

$$\begin{aligned}\cos\varphi_1 &= \frac{r_1 - r_2}{d}, \quad d = \|C_1C_2\| \\ \cos\varphi_2 &= \frac{r_1 + r_2}{d} \\ \sin\varphi_i &= \sqrt{1 - \cos^2\varphi_i}, \quad i = 1, 2.\end{aligned} \tag{3.18}$$

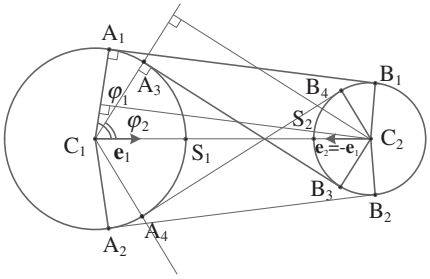

Рис. 3.12: Общие касательные к двум окружностям. Точки A_i и B_i находятся путем определения углов φ_1 и φ_2

Составим матрицы поворотов:

$$\mathbf{M}_i = \begin{bmatrix}\cos\varphi_i & -\sin\varphi_i \\ \sin\varphi_i & \cos\varphi_i\end{bmatrix}$$

и найдем точки A_j, B_j как

$$\begin{aligned} A_1 &= C_1 + r_1\mathbf{M}_1\vec{\mathbf{e}}_1, & B_1 &= C_2 + r_2\mathbf{M}_1\vec{\mathbf{e}}_1, \\ A_2 &= C_1 + r_1\mathbf{M}_1^T\vec{\mathbf{e}}_1, & B_2 &= C_2 + r_2\mathbf{M}_1^T\vec{\mathbf{e}}_1, \\ A_3 &= C_1 + r_1\mathbf{M}_2\vec{\mathbf{e}}_1, & B_3 &= C_2 - r_2\mathbf{M}_2\vec{\mathbf{e}}_1, \\ A_4 &= C_1 + r_1\mathbf{M}_2^T\vec{\mathbf{e}}_1, & B_4 &= C_2 - r_2\mathbf{M}_2^T\vec{\mathbf{e}}_1, \end{aligned}$$

где вектор $\vec{\mathbf{e}}_1 = \frac{C_2 - C_1}{d}$.

Исключение заблокированных отрезков При построении пути исключаются такие отрезки прямых, которые пересекаются с какими-либо препятствиями. Для определения пересечения отрезка прямой P_1P_2 с кругом радиуса r с центром в точке C будем использовать следующий критерий (рис. 3.13)

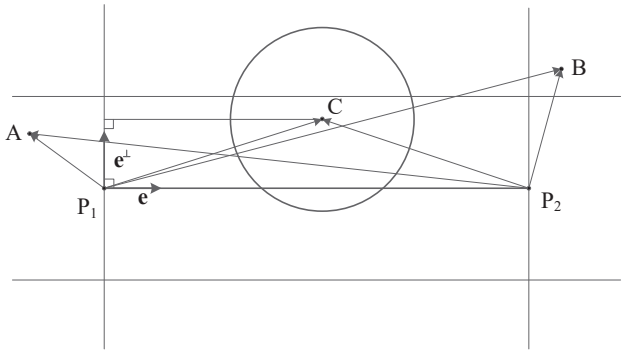

Рис. 3.13: Определение пересечения отрезка с кругом. При попадании центра круга C в заштрихованную область, являющуюся пересечением заштрихованных горизонтальной и вертикальной полос отрезок P_1P_2 считается заблокированным кругом.

Найдем единичный вектор $\vec{\mathbf{e}}$ вдоль отрезка P_1P_2 и ортогональный ему вектор $\vec{\mathbf{e}}^{\perp}$ как

$$\vec{\mathbf{e}} = (e_1, e_2)^T = \frac{\overrightarrow{P_2 - P_1}}{\|\overrightarrow{P_2 - P_1}\|}, \quad \vec{\mathbf{e}}^{\perp} = (-e_2, e_1)^T$$

Условием того, что расстояние от центра круга до прямой, соединяющей точки P_1 и P_2, меньше или равно радиусу круга r, является

$$d = \|\overrightarrow{(C - P_1)} \cdot \vec{\mathbf{e}}^{\perp}\| \leq r, \tag{3.19}$$

это условие означает, что точка C лежит внутри горизонтальной полосы шириной $2r$ (рис. 3.13). При этом проекция центра круга C на отрезок P_1P_2 должна лежать на отрезке P_1P_2, что эквивалентно условию

$$\left(\overrightarrow{(C - P_1)} \cdot \vec{\mathbf{e}}\right) \cdot \left(\overrightarrow{(C - P_2)} \cdot \vec{\mathbf{e}}\right) \leq 0, \tag{3.20}$$

что означает, что точка C лежит внутри вертикальной полосы, ширина которой равна длине отрезка P_1P_2 (рис. 3.13). Так для точки A оба угла – между $\vec{\mathbf{e}}$ и P_1A и между $\vec{\mathbf{e}}$ и P_2A – являются тупыми, поэтому оба скалярных произведения в (3.20) отрицательны. Для точки B углы между $\vec{\mathbf{e}}$ и P_1B и между $\vec{\mathbf{e}}$ и P_2B острые,

поэтому оба скалярных произведения положительны. Для точки C угол между $\vec{\mathbf{e}}$ и P_1C острый, а между $\vec{\mathbf{e}}$ и P_2C – тупой, поэтому одно скалярное произведение положительно, другое – отрицательно.

Равенство нулю допускается как в (3.19), так и в (3.20). В первом случае оно означает касание круга и прямой P_1P_2, во втором – то, что центр круга C лежит на прямой, перпендикулярной P_1P_2 и проходящей через конец отрезка.

Отметим, что на самом деле отрезок и круг не пересекаются в том и только в том случае, если центр круга лежит за пределами области, заштрихованной на рис. 3.14, которая состоит из прямоугольной полосы шириной $2r$ и двух полукругов радиуса r с центрами в концах отрезка. Эта область порождает дополнительную проверку расстояния от центра круга до начала и конца отрезка P_1P_2. Однако, в нашей задаче P_1P_2 – это отрезок касательной к одному из кругов, обозначающих препятствия, поэтому, если центр круга C находится внутри одного из полукругов (включая его внешнюю границу), значит, круговые препятствия пересекаются (рис. 3.14), что по условию не допускается.

Таким образом, проверка условий (3.19) и (3.20) в нашей задаче достаточна.

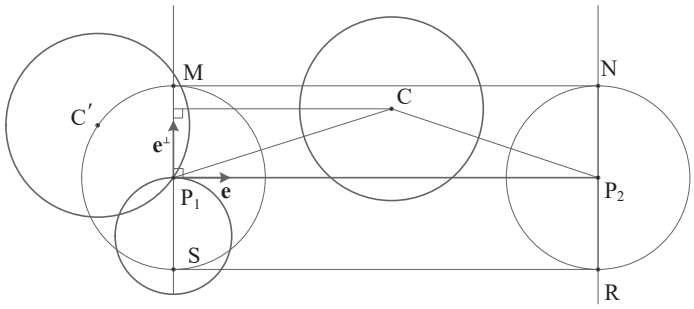

Рис. 3.14: Условие заблокированности отрезка кругом. При попадании центра круга в заштрихованную область круг пересекается с отрезком P_1P_2, и отрезок является заблокированным.

Вычисление максимальной скорости на участках пути В соответствии с моделью, на прямых участках пути максимальная скорость движения $v_{\max} = u_{\max}$. Время движения по соответствующему отрезку составляет

$$t_{\min} = \frac{l}{u_{\max}} \tag{3.21}$$

На дугах окружностей максимальная скорость определяется следующим образом. Пусть робот движется по окружности против часовой стрелки. Тогда максимальное значение управляющего воздействия u_1 равно $u_{\max}$. При этом угловая скорость робота равна угловой скорости движения его центра по соответствующей окружности, при этом

$$\begin{aligned} v &= \frac{(u_1 + u_2)}{2} \\ \omega &= \frac{(u_1 - u_2)}{b} \\ v &= \omega r \end{aligned} \tag{3.22}$$

Отсюда следует, что при $u_1 = u_{\max}$

$$u_2 = u_{\max}\frac{2r - b}{2r + b}$$

и максимальная скорость при движении по дуге составляет

$$v_{\max} = u_{\max}\frac{2r}{2r + b}.$$

Соответственно, время движения по дуге окружности радиуса r с углом φ составит

$$t_{\min} = \frac{l}{v_{\max}} = \frac{r\varphi}{v_{\max}} = \frac{r\varphi(2r + b)}{2ru_{\max}} = \frac{\varphi(2r + b)}{2u_{\max}} \tag{3.23}$$

Время, которое робот тратит для поворота на месте на угол φ, можно определить из последнего соотношения в (3.22). Максимальная возможная угловая скорость поворота

$$\omega_{\max} = \frac{2u_{\max}}{b}$$

и, таким образом, минимальное время такого поворота составит

$$t_{\min} = \frac{\varphi b}{2u_{\max}}, \tag{3.24}$$

что, если сравнить с (3.23), соответствует движению по окружности нулевого радиуса.

Эти соотношения используются при расчете времени движения по ребрам графа.

Обратим внимание, что для рассматриваемой кинематической модели время движения как по прямым участкам, так и по дугам окружностей прямо пропорционально длине пути и обратно пропорционально максимальному значению, ограничивающему управляющие воздействия в системе. Это означает, что, за исключением поворотов на месте, самый быстрый путь из начальной точки в конечную в случае, если точки лежат на границах препятствий, будет совпадать с одним из локальных минимумов в задаче поиска кратчайшего пути в геометрическом смысле. Выбор такого локального минимума зависит от геометрического параметра b. Докажем следующее предложение:

Предложение 4 *Для кинематической модели движения двухколесного робота оптимальным по быстродействию путем будет локально кратчайший в геометрическом смысле путь с добавлением поворотов в начальной и конечной точке.*

Доказательство.Рассмотрим две окружности 1 и 2 с центрами в точках C_1 и C_2 и радиусами r_1 и r_2, соответственно. Рассмотрим путь из точки D, принадлежащей окружности 1, до точки B, принадлежащей окружности 2. Пусть для определенности точки C_1 и C_2 лежат на горизонтальной прямой. Этот путь проходит от точки D по дуге первой окружности до точки A, в которой общая касательная к двум окружностям AB касается первой из них, а затем – по отрезку AB до касания со второй окружностью. Пусть для определенности точки C_1 и C_2 лежат на горизонтальной прямой, расстояние межу центрами окружностей $\|C_1C_2\|$ равно d, а угол между касательными к окружности 1 в точке D и в точке A равен ψ (рис. 3.15).

Такой путь является кратчайшим с геометрической точки зрения. Как было отмечено выше, этот путь, проходящий по дуге окружности и отрезку прямой, также является и быстрейшим, поскольку среди локальных минимумов по длине пути время, затрачиваемое на этот путь, пропорционально длине дуги, проходимой по окружности 1. Время движения по такому пути равно сумме времен движения по дуге t_a и по отрезку t_l и составляет, в соответствии с (3.23) и (3.21),

$$t = t_a + t_l = \frac{\psi(2r_1 + b)}{2u_{\max}} + \frac{\sqrt{d^2 - (r_2 - r_1)^2}}{u_{\max}} \tag{3.25}$$

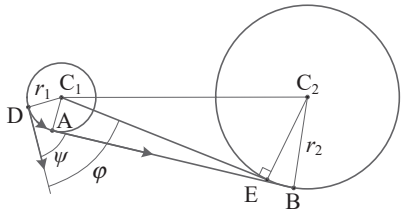

Рис. 3.15: К доказательству предложения 1. При стремлении радиуса r_1 к нулю время движения стремится к времени, затрачиваемому на поворот и на прямолинейное движение к точке касания.

Величина $\sqrt{d^2-(r_2-r_1)^2}$ является длиной отрезка AB; для получения этого выражения обратимся к рис. 3.12. Угол дуги DA равен ψ.

В выражении (3.25) перейдем к пределу при $r_1 \to 0$. Точки A и D при этом будут стремиться к точке C_1, точка B – к точке E, где C_1E – касательная к окружности 2, а угол ψ – к углу φ (рис. 3.15) между начальным направлением и направлением BE. Первое слагаемое при этом обратится в

$$t_a \to \frac{\varphi b}{2u_{\max}}, \tag{3.26}$$

а второе – в

$$t_l \to \frac{\sqrt{l^2-r_2^2}}{u_{\max}} \tag{3.27}$$

При этом, поскольку суммарное время $t = t_a + t_l$ является минимальным при любом радиусе r_1, то в пределе при $r_1 = 0$ оно также останется минимальным для всех возможных движений.

Время (3.26) является временем поворота на месте в точке C_1 на угол φ, а время (3.27) – временем движения по касательной из точки C_1 до точки E, принадлежащей окружности 2. Таким образом, мы доказали, что оптимальным по быстродействию движением из точки с произвольнной ориентацией корпуса робота до точки касания на окружности является поворот на месте до направления движения по касательной к окружности, а затем прямолинейное движение по этой касательной.

Аналогичным образом можно доказать оптимальность составного движения, которое включает в себя повороты на месте и прямолинейный участок, для движения из одной точки в другую без препятствий между ними, перейдя в выражении (3.25) к пределу при $r_1, r_2 \to 0$.

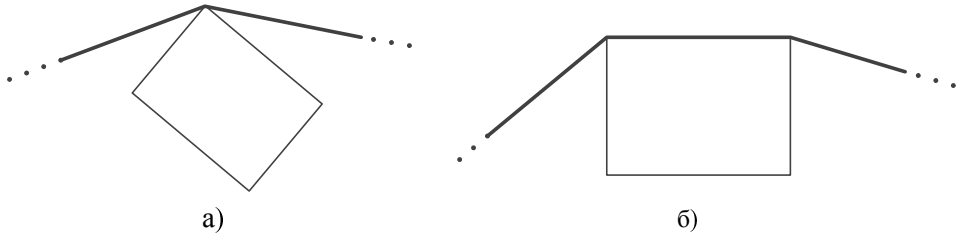

Рис. 3.16: Кратчайший путь, обходящий полигональное препятствие. Время движения складывается из времени движения по левому участку, по правому участку и времени поворота.

В случае, если допускаются полигональные препятствия на пути робота, при этом геометрически кратчайший путь проходит через угол полигона (рис. 3.16, а) или по грани полигона (рис. 3.16, б), верно следующее

Следствие 3 *В случае полигональных препятствий кратчайший путь также является является одним из локальных минимумов в задаче поиска кратчайшего пути.*

Для *доказательства* достаточно разбить путь на участки, проходящие между углами полигональных препятствий, через которые проходит кратчайший путь. В силу доказанного выше утверждения, каждый такой участок будет оптимальным с точки зрения быстродействия при движении робота от начала участка к его концу, при этом в конечной точке производится поворот на такой угол, при котором новое направление движения робота совпадает с продолжением движения по следующему участку. Этот поворот производится вокруг вершины полигонального препятствия.

Рис. 3.17: Кратчайший путь, обходящий гладкое выпуклое препятствие. Время его обхода складывается из времени движения по прямолинейному отрезку, длина которого равна длине участка границы препятствия, и времени поворота из начальной ориентации к конечной.

Следствие 4 *В случае препятствий, заданных в виде произвольных гладких выпуклых фигур, кратчайший путь также является одним из локальных минимумов в задаче поиска кратчайшего пути.*

Для *доказательства* этого следствия рассмотрим полигональное приближение произвольного выпуклого препятствия. Оптимальная траектория будет проходить по углам и ребрам такой фигуры (рис. 3.17, а). При увеличении числа углов полигона, приближающего гладкое препятствие, такая траектория остается оптимальной. Следовательно, в предельном переходе при количестве углов приближающего полигона, стремящемся к бесконечности, оптимальная траектория будет стремиться к геометрически кратчайшей траектории, обходящей гладкое препятствие (рис. 3.17, б).

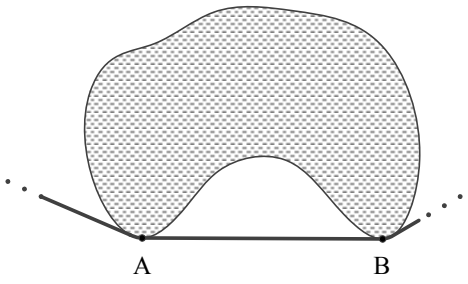

Рис. 3.18: Кратчайший путь, обходящий невыпуклое препятствие. Время обхода считается так же, как если бы это препятствие было заменено на его выпуклую оболочку.

Что касается невыпуклых препятствий, рассмотрим, например, локально кратчайший геометрический путь, обходящий невыпуклую часть препятствия на рис. 3.18. Точки A и B – крайние точки границы препятствия, между которыми путь идет по прямой. В отсутствие препятствия этот участок, как было отмечено, является оптимальным с точки зрения быстродействия. Наложение любых дополнительных фазовых ограничений не улучшает оптимальных траекторий, поэтому при наличии препятствия участок AB остается оптимальным. Следовательно, и в случае невыпуклых препятствий кратчайший путь является быстрейшим, а все невыпуклые препятствия можно в некоторых случаях для поиска оптимальной траектории заменить на их выпуклые оболочки. Такая замена однако, невозможна, если существуют препятствия, которые не пересекаются друг с другом, при этом их выпуклые оболочки пересекаются. Типичным примером является лабиринт.

Таким образом, доказано, что для кинематической модели движения робота при наличии произвольных препятствий оптимальным по быстродействию является один из локальных минимумов в задаче поиска кратчайшего пути, при этом в точках перегиба пути оптимальным движением является поворот на месте вокруг точки перегиба. Выбор локального минимума зависит от геометрического параметра b робота, при этом быстрейший путь может не совпадать с геометрически кратчайшим, однако принцип поиска такого оптимального пути сохраняется. Следовательно, мы можем определить быстрейший путь методом перебора возможных путей на графе.

Построение графа переходов Оптимальный путь, соединяющий две заданные точки и обходящий непересекающиеся круговые препятствия, состоит из отрезков прямых и дуг окружностей, при этом отрезки и дуги чередуются. В начале и в конце траектории производятся повороты на месте.

Для поиска оптимального пути составим граф Г переходов между точками, лежащими на возможных оптимальных траекториях. Этими точками являются начало пути и конец пути, а также точки, лежащие на окружностях, которые ограничивают препятствия.

Все переходы следует добавлять с проверкой, является ли данный переход свободным или заблокированным каким-либо препятствием.

Переход между точкой старта и точкой финиша с учетом поворота робота до необходимого направления из изначально заданного, а также конечного поворота в точке финиша до заданной терминальной ориентации добавим как путь из трех вершин. Обозначим точку старта через S; точку финиша – через F; вершину, соответствующую точке старта после поворота до направления движения к финишу – через S_F; вершину, соответствующую концу этой прямой до поворота к терминальному направлению – через F_S (рис. 3.19).

Ребра графа, соответствующие поворотам на месте, имеют нулевую длину, однако время, требующееся для их прохождения (вес соответствующего ребра) определяется соотношением (3.24). При этом самым быстрым будет такой поворот в том направлении из двух возможных – по часовой стрелке или против часовой стрелки, – для которого угол между начальной и конечной ориентацией меньше.

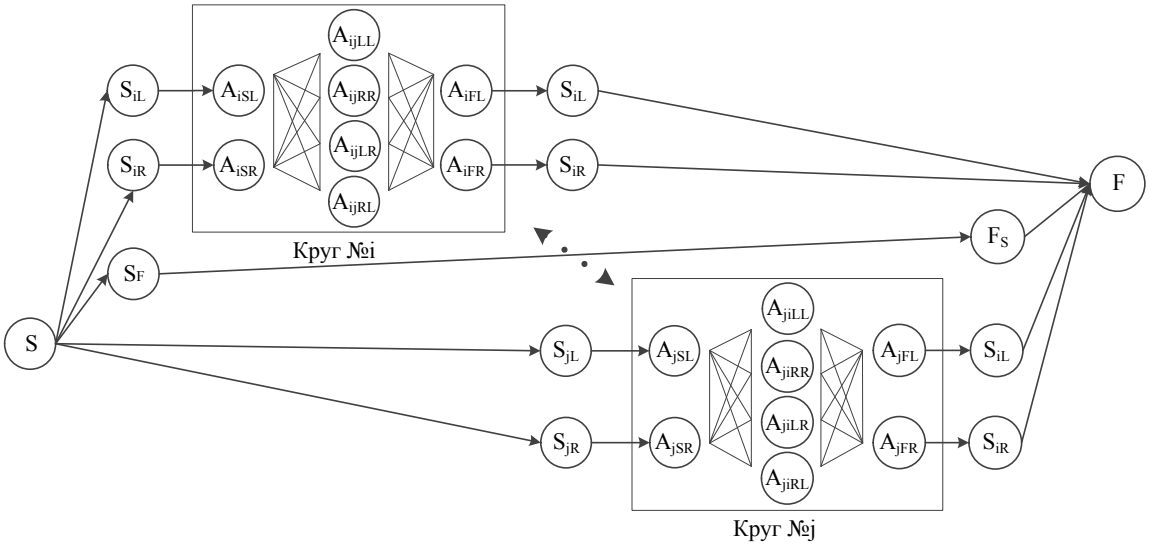

Рис. 3.19: Граф переходов между начальной и конечной точкой и окружностями. Показаны узлы графов, схематично отмечены соединения вершин, относящиеся к одной и той же окружности, а также соединения между окружностями.

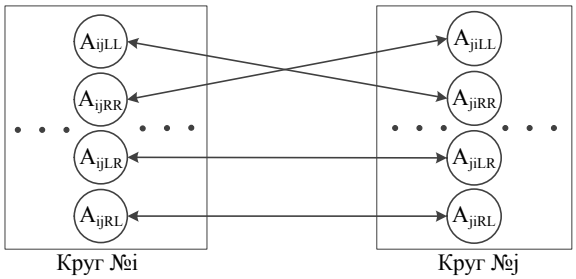

Рис. 3.20: Участки графа переходов между двумя окружностями.

Переходы между началом пути и окружностями определим как прямые, соединяющие точку старта и точки, в которых касательные к соответствующему кругу и проходящие через точку старта, касаются этого круга. Обозначим соответствующие точки на окружностях через A_{iSL} и A_{iSR} и зададим одноименные вершины графа. В этих обозначениях i задает номер окружности, индекс S обозначает, что прямая соединяется с точкой старта, индексы L и R – что прямые подходят к окружности соответственно слева и справа, если смотреть со стороны окружности на точку старта. Аналогичным образом определим прямые,

соединяющие окружности и точку финиша, обозначив их A_{iFL} и A_{iFR}. При этом необходимо учитывать, что поворот робота до его движения по соответствующей прямой, а также время, затрачиваемое на этот поворот, зависит от направления движения по этой прямой, поэтому необходимо добавить вершины графа Г, соответствующие поворотам из начальной ориентации робота до соответствующего направления (рис. 3.19). Эти вершины обозначим через S_{iL} и S_{iR} для точки старта, и через F_{iL} и F_{iR} для точки финиша. Будем обозначать левую и правую сторону для точки старта и точки финиша, смотря на точку со стороны окружности.

Переходы между точками двух различных окружностей определяются общими касательными к этим окружностям. Без исключения заблокированных отрезков, каждая пара окружностей добавляет четыре вершины, соответствующих точкам касания на одной из этих окружностей, и четыре – на другой. Точки касания (и вершины графа Г) будем обозначать как A_{ijLL}, A_{ijRR}, A_{ijLR}, A_{ijRL}, где индексы i, j обозначают порядковые номера окружностей в списке, а L и R – стороны (левую и правую), с которой касательные подходят к одной и к другой окружности. Эти точки соединены четырьмя отрезками касательных, которые задают ребра графа Г (рис. 3.20).

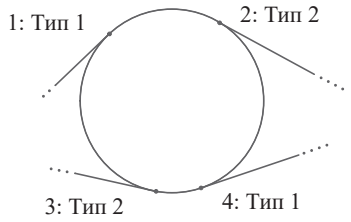

Рис. 3.21: Два типа точек касания на окружности: Тип 1 - касательная подходит «слева», тип 2 - касательная подходит «справа».

После определения всех точек касания, принадлежащих окружностям, для каждой из этих окружностей зададим переходы между всеми такими точками. Точки на каждой окружности можно разделить на два типа (рис. 3.21). В точках первого типа касательная подходит слева, если смотреть из центра окружности (точки с номерами 1 и 4 на рис. 3.21). В точках второго типа (2 и 3) касательная подходит справа. При этом, если траектория робота приходит на окружность в точке типа 1, то она должна выйти с окружности в точке типа 2, и наоборот. Траектории, содержащие переход из одной точки в другую точку такого же типа заведомо не являются оптимальными, поскольку содержат точки разворота (рис. 3.22, путь 3). Оптимальная траектория не должна содержать

разворотов, поэтому движение по окружности из точки типа 1 в точку типа 2 должно производиться по часовой стрелке, а из точки типа 2 в точку типа 1 – против часовой стрелки (рис. 3.22, путь 2). Однако, в некоторых случаях, а именно, когда угол дуги, которую необходимо пройти по окружности, больше π, неоптимальная траектория с разворотом оказывается короче (и быстрее), см. путь 1 на рис. 3.22. Это означает, что, если от точки входа на окружность до точки выхода с нее необходимо пройти угол, больший π, то такая траектория заведомо не является оптимальной, и при составлении графа переходов такие дуги можно не включать в граф.

Отметим, что в принятых обозначениях точками типа 1 являются все точки, принадлежащие окружностям, которые имеют в третьей позиции индекса литеру R, а точками типа 2 – те, которые в этом индексе имеют литеру L.

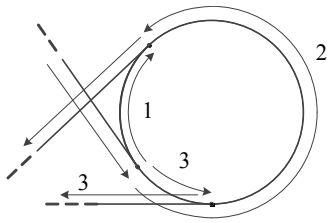

Рис. 3.22: Возможные участки траектории, заведомо не являющиеся оптимальными. Траектория подходит из левого верхнего угла и уходит в левом нижнем углу. Путь 1 через точки A и C и путь 3 через точки A и B содержат развороты, путь 2 длиннее, чем путь 1.

Итак, опишем общий вид алгоритма для построения графа переходов.

Алгоритм

Шаг 1. Проверить, заблокирован ли отрезок SF.

Если отрезок SF не заблокирован:

- Добавить точки S_F и F_S.
- Добавить переход типа «поворот» от S до S_F.
- Добавить переход типа «прямая» от S_F до F_S.
- Добавить переход типа «поворот» от F_S до F.

■ Конец.

Иначе перейти к **Шагу 2**.

Шаг 2. Для каждой окружности №i из списка:

- Найти касательные к окружности №i, проходящие через точку S.
- Для каждого из отрезков SA_{iSR}, SA_{iSL} найденных левой и правой касательных проверить, заблокирован ли этот отрезок.

  Если отрезок не заблокирован:

  - Добавить к графу вершину S_{iR} или S_{iL} для левой или правой касательной, соответственно
  - Добавить соответствующую вершину A_{iSR} или A_{iSL}.
  - Добавить переход типа «поворот» от S до S_{iR} или S_{iL}
  - Добавить переход типа «прямая» от S_{iR} до A_{iSR} или от S_{iL} до A_{iSL}, соответственно.

Шаг 3. Для каждой окружности №i из списка:

- Найти касательные к окружности №i, проходящие через точку F.
- Для каждого из отрезков $A_{iSR}F$, $A_{iSL}F$ найденных левой и правой касательных проверить, заблокирован ли этот отрезок.

  Если отрезок не заблокирован:

  - Добавить к графу вершину F_{iR} или F_{iL} для левой или правой касательной, соответственно.
  - Добавить соответствующую вершину A_{iFR} или A_{iFL}.
  - Добавить переход типа «поворот» от F_{iR} или F_{iL} до F.
  - Добавить переход типа «прямая» от A_{iFR} до F_{iR} или от A_{iFL} до F_{iL}, соответственно.

Шаг 4. Для каждой окружности №i из списка:

Для каждой окружности №j из списка, $j > i$:

- Найти все общие касательные между окружностями №i и №j.
- Для каждой найденной касательной проверить, заблокирован ли отрезок этой касательной между точками касания с окружностями.

  Если отрезок не заблокирован:

- Добавить в граф вершину, соответствующую точке касания на окружности №i: A_{ijLL}, A_{ijRR}, A_{ijLR} или A_{ijRL}.
- Добавить в граф вершину, соответствующую точке касания на окружности №j: A_{jiRR}, A_{jiLL}, A_{jiLR}, A_{jiRL}.
- Добавить переход типа «прямая» между добавленными точками.

Шаг 5. Для каждой окружности №i из списка:

- Найти в списке вершин графа все вершины, соответствующие точкам на данной окружности (т.е. у которых первый индекс равен i).
- Разделить найденные точки на два типа – тип 1 и тип 2. К первому типу отнести все точки со значением индекса R (RR, RL), ко второму – со значением индекса L (LR, LL).
- Для каждой точки A из множества точек первого типа:

  Для каждой точки B из множества точек второго типа проверить угол при движении по дуге AB по часовой стрелке. Если этот угол меньше π:
  - Добавить переход типа «дуга» из точки A в точку B с движением по часовой стрелке.
  - Добавить переход типа «дуга» из точки B в точку A с движением против часовой стрелки.

■ Конец.

В алгоритме в первую очередь проверяется возможность перейти из точки старта непосредственно в точку финица. Если этот участок пути не заблокирован препятствиями, значит, это и будет кратчайший путь, и дальнейших построений не требуется. В противном случае для определения оптимальной траектории необходимо построить весь граф.

Нахождение оптимальной траектории Для нахождения оптимальной траектории с помощью построенного графа Г будем использовать алгоритм Дейкстры поиска кратчайших путей на графе из заданной начальной вершины в его модификации, когда требуется определить кратчайший путь до другой заданной вершины.

Приведем краткое описание алгоритма [121]. Для всех вершин v_i, кроме стартовой, длины пути устанавливаются в $+\infty$ или большое положительное число,

заведомо большее любой возможной длины пути. Вершинам присваивается отметка «посещенная» или «непосещенная», причем в начале работы все вершины имеют отметку «непосещенная». Из всех непосещенных вершин выбирается та v_i, для которой длина рассчитанного пути наименьшая, и для каждой соседней с ней вершины v_j рассчитывается длина пути, складывающаяся из рассчитанной длины для данной вершины $l(v_i)$ и веса соответствующего ребра, соединяющего v_i и v_j. Если для v_j рассчитанная таким образом длина меньше, чем сохраненная, то сохраняется новая длина, а «предшествующей» v_j вершиной назначается v_i. Вершина получает отметку «посещенная», когда для нее проверены таким образом все ее соседние непосещенные вершины. Алгоритм прерывается, когда заданная финишная вершина становится посещенной.

Кратчайший путь при этом можно восстановить, пройдя от финишной вершины к стартовой в обратном порядке, используя записанные индексы предшествующих вершин. В качестве длин можно выбрать геометрическую длину соответствующего участка, либо время, затрачиваемое на прохождение этого участка. В первом случае получим кратчайшую траекторию, соединяющую начальную и конечную точки, и ее длину; во втором – минимальное время, необходимое для перемещения из начальной точки в конечную.

Вычислительная сложность данного алгоритма для графа Г, имеющего N вершин, оценивается в $O(N^2)$ операций. Для графа, построенного в нашем случае, при числе окружностей M число вершин графа будет равно (без учета блокирования отрезков, при котором вершины не добавляются в граф) $2 + 2$ (точка старта и точка финиша, а также точки поворота), $+\ 2M + 2M$ точек касания прямых, проходящих через точку старта, с соответствующими точками поворота $+\ 2M + 2M$ аналогичных точек для финиша, $+\ 4M(M-1)$ общих точек касания попарно для всех касательных к парам окружностей. Таким образом, граф Г имеет максимум

$$N = 4 + 8M + 4M(M-1) = O(4M^2)$$

вершин, а вычислительная сложность пропорциональна $O(16M^4)$.

Проверка достижимости заданной конечной точки графа производится параллельно с определением кратчайшего пути. Если в какой-то момент оказывается, что минимальная длина для непосещенной вершины равна $+\infty$, значит, все достижимые из точки старта вершины уже посещены, и заданная точка финиша недостижима. На практике это означает, что, если при нахождении кратчайшего пути на графе в результате работы алгоритма Дейкстры длина пути до заданной точки равна $+\infty$, значит, заданная точка недостижима. При описанном

способе построения графа переходов для непересекающихся препятствий и если начальная и конечная точки не находятся внутри препятствия, такая ситуация невозможна.

Для поиска кратчайшего пути на графе, помимо алгоритма Дейкстры, можно пользоваться другими методами, например, набирающим популярность методом A* [122], который может находить кратчайший путь несколько быстрее, чем метод Дейкстры.

3.4 Вычислительный эксперимент с методом касательных

Для реализации описанного алгоритма поиска кратчайшего пути была написана программа на языке Python, автоматически считывающая задаваемый файл настроек с заданием начальной и конечной точек, а также окружностей, ограничивающих препятствия. В файле настроек также можно задавать ограничение на управляющие воздействия $u_{\max}$. Программа автоматически рассчитывает все необходимые касательные, определяет заблокированные отрезки и строит граф переходов, добавляя в него необходимые вершины и ребра.

Все точки в программе задаются тремя значениями. Первые два соответствуют декартовым координатам точки на плоскости. Третье значение определяет угол ориентации робота при его прохождении через данную точку. Так, точка старта в качестве третьего значения будет иметь начальную ориентацию робота, точка финиша – конечную ориентацию.

На выходе программа сохраняет файл, в котором описывается путь в виде последовательности точек, а также производится расшифровка – указывается тип ребра (поворот, дуга окружности, отрезок прямой), необходимые параметры, определяющие соответствующую этому ребру геометрическую фигуру (для поворота – начальная точка, угол и направление поворота; для дуги – координаты центра и радиус окружности, координаты начальной и конечной точек, угловое направление движения; для отрезка прямой – координаты начала и конца отрезка), а также время движения по каждому ребру. Вычисляется и сохраняется также общее (оптимальное) время движения из точки старта в точку финища.

В программе также можно указать режим поиска – кратчайшего пути или быстрейшего пути. Как уже было сказано выше, эти пути совпадают, за исключением начального и конечного поворота, которые при определении длины дают нулевой вклад. В режиме поиска оптимальной длины выдается длина пути без учета ограничений $u_{\max}$.

При индексации точек в качестве обозначений индексов, соответствующих старту и финишу, литеры S и F повторяются для того, чтобы длины текстовых идентификаторов точек были одинаковыми.

Пример работы программы приведен на рис. 3.23. Заданы следующие настройки:

- Точка старта: $(x = 0,\ y = 0,\ \varphi = 0)$
- Точка финиша: $(x = 5,\ y = 5,\ \varphi = 0)$
- Окружность 1: $(x_c = 1.5,\ y_c = 1.5,\ r = 0.8)$
- Окружность 2: $(x_c = 3.5,\ y_c = 3.5,\ r = 0.8)$
- Окружность 3: $(x_c = 1,\ y_c = 4,\ r = 1.5)$
- Окружность 4: $(x_c = 4,\ y_c = 1,\ r = 1.5)$
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

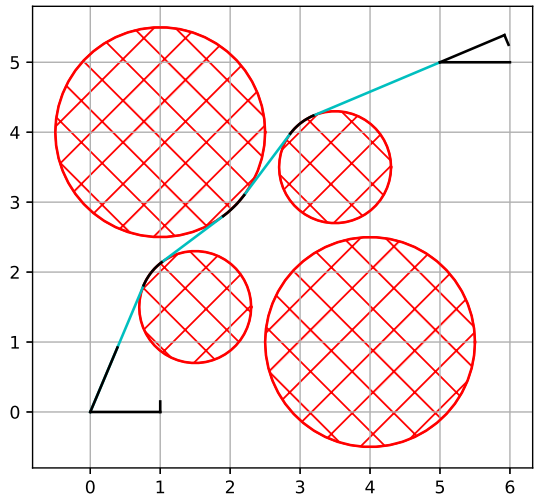

Рис. 3.23: Оптимальная траектория для четырех препятствий.

В результате расчетов получены следующие значения для оптимальной траектории:

Оптимальный путь определен как $S \to S_{0L} \to A_{0SL} \to A_{02LR} \to A_{20LR} \to A_{21RL} \to A_{12RL} \to A_{1FL} \to F_{1L} \to F$. Участки движения при жтом следующие:

1. Поворот из точки (0, 0, 0) на угол 1,172 против часовой стрелки, время движения $0,586$;

2. Прямая из точки $(0;\ 0;\ 1,172)$ в точку $(0,762;\ 1,810;\ 1,172)$, время движения $1,964$;

3. Дуга окружности с параметрами $(1,5;\ 1,5,\ 0,8)$ из точки $(0,762;\ 1,810;$ 1.172) в точку $(1,02;\ 2,14;\ 0,643)$, по часовой стрелке, время движения $0,687$;

4. Прямая из точки $(1,02;\ 2,14;\ 0,643)$ в точку $(1,9;\ 2,8;\ 0,643)$, время движения $1,1$;

5. Дуга окружности с параметрами $(1,0,\ 4,0,\ 1,5)$ из точки $(1,9;\ 2,8;\ 0,643)$ в точку $(2,2;\ 3,1;\ 0,927)$, против часовой стрелки, время движения $0,567$;

6. Прямая из точки $(2,2;\ 3,1;\ 0,927)$ в точку $(2,86;\ 3,98;$ 0. 927), время движения $1,1$;

7. Дуга окружности с параметрами $(3,5;\ 3,5;\ 0,8)$ из точки $(2,86;\ 3,98;\ 0,927)$ в точку $(3,189;\ 4,237;\ 0,398)$, по часовой стрелке, время движения $0,687$;

8. Прямая из точки $(3,189;\ 4,237;\ 0,398)$ в точку $(5,0;\ 5,0;\ 0,398)$, время движения $1,964$;

9. Поворот из точки $(5,0;\ 5,0;\ 0,398)$ на угол 0,398 по часовой стрелке, время движения $0,199$;

Общее время движения – $8,856$ единиц.

Также в качестве примера приведем построение оптимального пути для большего количества препятствий. Найденный оптимальный путь приведен на рис. 3.24.

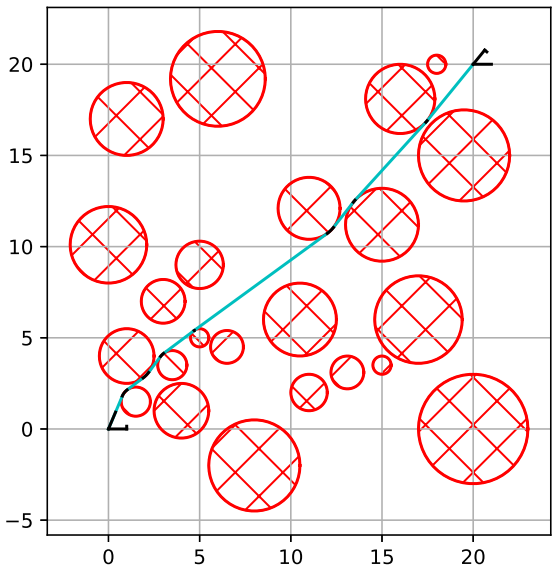

Рис. 3.24: Оптимальная траектория для 23 препятствий.

Для наглядности приведены результаты экспериментов с различным количеством препятствий и различными начальными и конечными точками.

Оптимальные траектории для двух препятствий.

- Точка старта: $(x = -1.5,\ y = 0.5,\ \varphi = 0)$
- Точка финиша: $(x = 8,\ y = -3,\ \varphi = 0)$
- Окружность 1: $(x_c = 2,\ y_c = -1.5,\ r = 1.2)$
- Окружность 2: $(x_c = 5,\ y_c = -1,\ r = 0.8)$
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

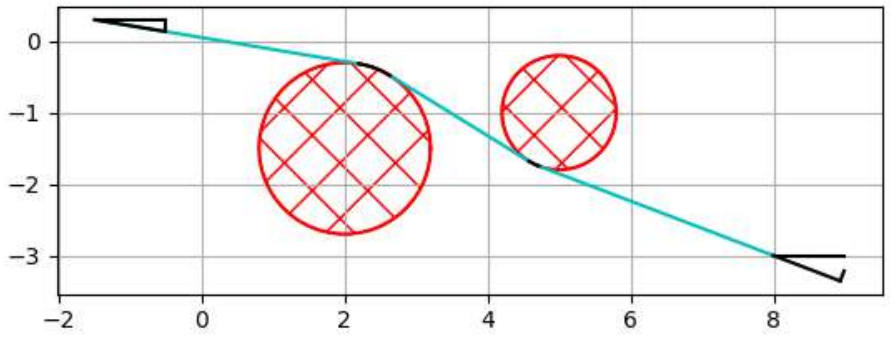

Рис. 3.25: Оптимальная траектория для двух препятствий.

Далее, положение препятствий не меняется, а меняются начальная и конечная точки.

- Точка старта: $(x = -1,\ y = -1,\ \varphi = 0)$
- Точка финиша: $(x = 8,\ y = -1,\ \varphi = 0)$
- Окружность 1: $(x_c = 2,\ y_c = -1.5,\ r = 1.2)$
- Окружность 2: $(x_c = 5,\ y_c = -1,\ r = 0.8)$
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

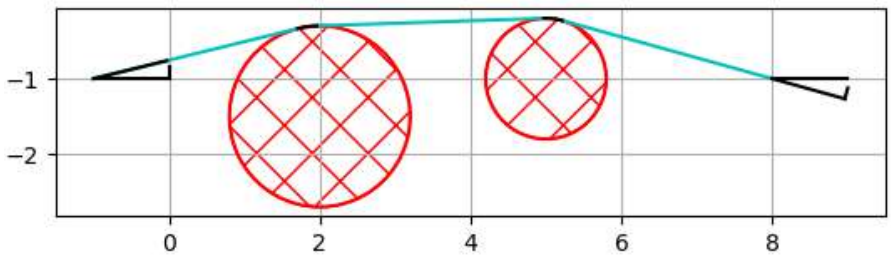

Рис. 3.26: Оптимальная траектория для двух препятствий.

Оптимальные траектории для трех препятствий. Во всех экспериментах с тремя препятствиями точки старта и финиша будут одинаковыми.

- Точка старта: $(x = -1,\ y = 2.5,\ \varphi = 0)$
- Точка финиша: $(x = 8,\ y = -3,\ \varphi = 0)$
- Окружность 1: $(x_c = 2,\ y_c = 1,\ r = 1.5)$
- Окружность 2: $(x_c = 4.2,\ y_c = -2.2,\ r = 1.2)$
- Окружность 3: $(x_c = 6,\ y_c = 1,\ r = 0.8)$
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

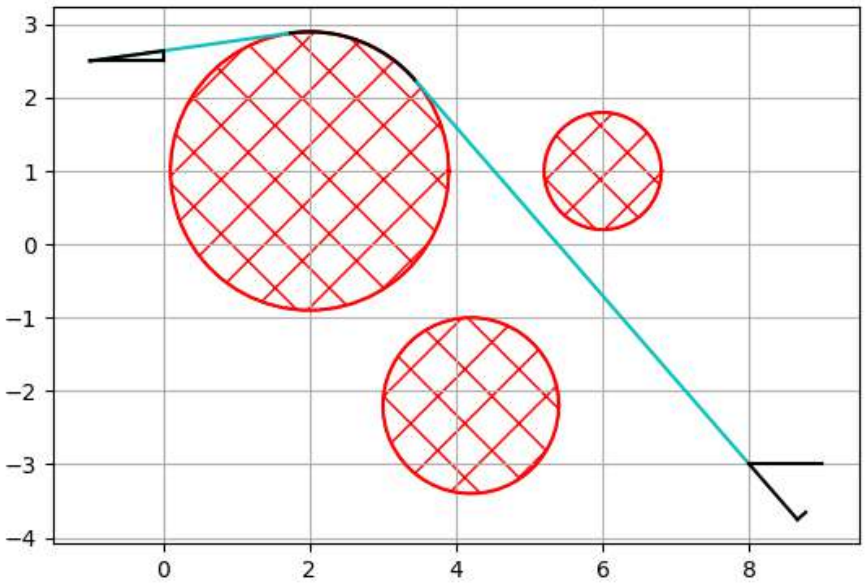

Рис. 3.27: Оптимальная траектория для трех препятствий.

Изменим положение двух окружностей с меньшим радиусом.

- Точка старта: $(x = -1,\ y = 2.5,\ \varphi = 0)$

- Точка финиша: $(x = 8,\ y = -3,\ \varphi = 0)$
- Окружность 1: $(x_c = 2,\ y_c = 1,\ r = 1.5)$
- Окружность 2: $(x_c = 3,\ y_c = -2.2,\ r = 1.2)$
- Окружность 3: $(x_c = 6,\ y_c = -2,\ r = 0.8)$
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

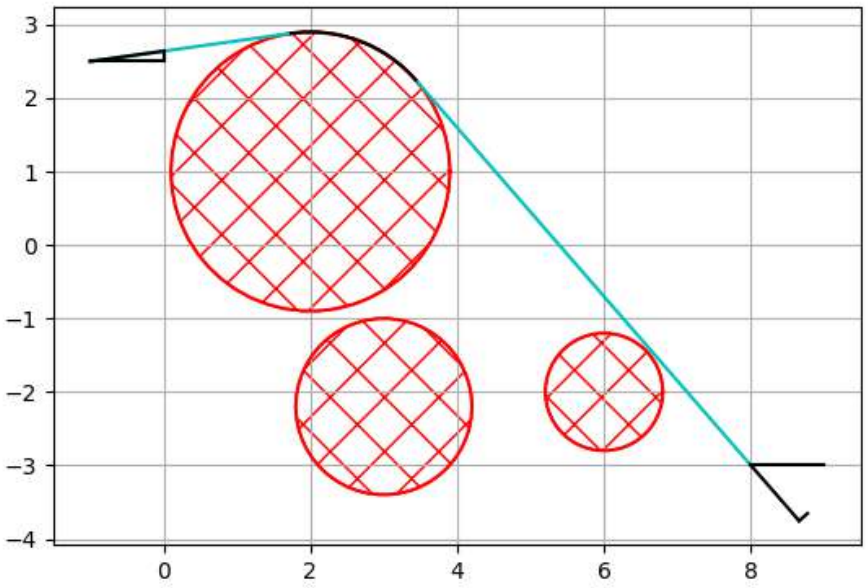

Рис. 3.28: Оптимальная траектория для трех препятствий.

И, наконец, передвинем окружность с самым маленьким радиусом.

- Точка старта: $(x = -1,\ y = 2.5,\ \varphi = 0)$
- Точка финиша: $(x = 8,\ y = -3,\ \varphi = 0)$
- Окружность 1: $(x_c = 2,\ y_c = 1,\ r = 1.5)$
- Окружность 2: $(x_c = 3,\ y_c = -2.2,\ r = 1.2)$

- Окружность 3: ($x_c = 6$, $y_c = 0$, $r = 0.8$))
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

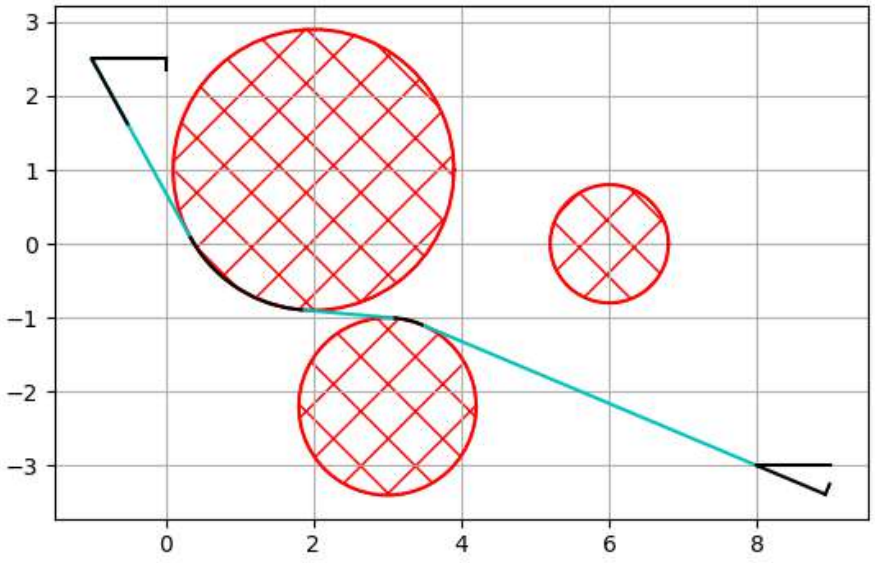

Рис. 3.29: Оптимальная траектория для трех препятствий.

Оптимальные траектории для четырех препятствий.

- Точка старта: ($x = -1$, $y = 2.5$, $\varphi = 0$)
- Точка финиша: ($x = 10$, $y = 0$, $\varphi = 0$)
- Окружность 1: ($x_c = 7$, $y_c = 1.5$, $r = 0.8$)
- Окружность 2: ($x_c = 9$, $y_c = 0$, $r = 0.8$)
- Окружность 3: ($x_c = 3$, $y_c = 1$, $r = 1.5$)
- Окружность 4: ($x_c = 6.5$, $y_c = -1$, $r = 1.5$)
- Ограничение $u_{\max} = 1$

- Расстояние между колесами $b = 1$

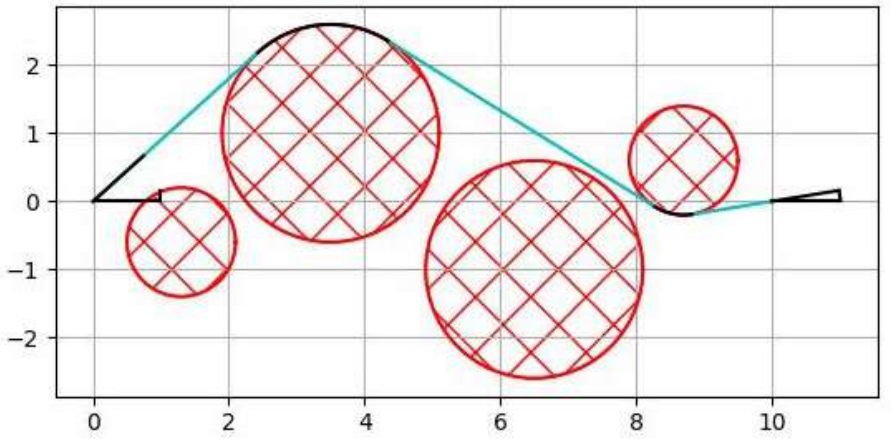

Рис. 3.30: Оптимальная траектория для четырех препятствий.

Изменим конечную точку, подняв ее на 1 единицу вверх.

- Точка старта: ($x = -1$, $y = 2.5$, $\varphi = 0$)
- Точка финиша: ($x = 10$, $y = 1$, $\varphi = 0$)
- Окружность 1: ($x_c = 7$, $y_c = 1.5$, $r = 0.8$)
- Окружность 2: ($x_c = 9$, $y_c = 0$, $r = 0.8$)
- Окружность 3: ($x_c = 3$, $y_c = 1$, $r = 1.5$)
- Окружность 4: ($x_c = 6.5$, $y_c = -1$, $r = 1.5$)
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

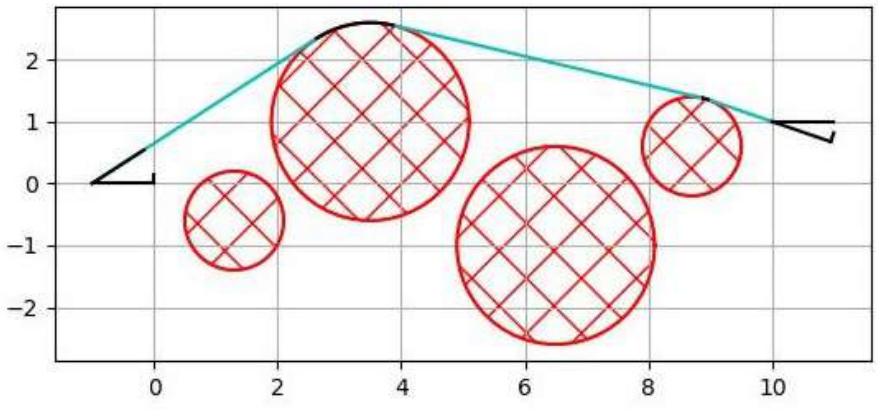

Рис. 3.31: Оптимальная траектория для четырех препятствий.

Оптимальные траектории для пяти препятствий.

- Точка старта: $(x = -1,\ y = 2.5,\ \varphi = 0)$
- Точка финиша: $(x = 10,\ y = -3,\ \varphi = 0)$
- Окружность 1: $(x_c = 2,\ y_c = 1,\ r = 1.5)$
- Окружность 2: $(x_c = 3,\ y_c = -2,\ r = 1.0)$
- Окружность 3: $(x_c = 6.5,\ y_c = 0.5,\ r = 0.8)$
- Окружность 4: $(x_c = 5,\ y_c = -0,5,\ r = 0.6)$
- Окружность 5: $(x_c = 8,\ y_c = -0.8,\ r = 0.4)$
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

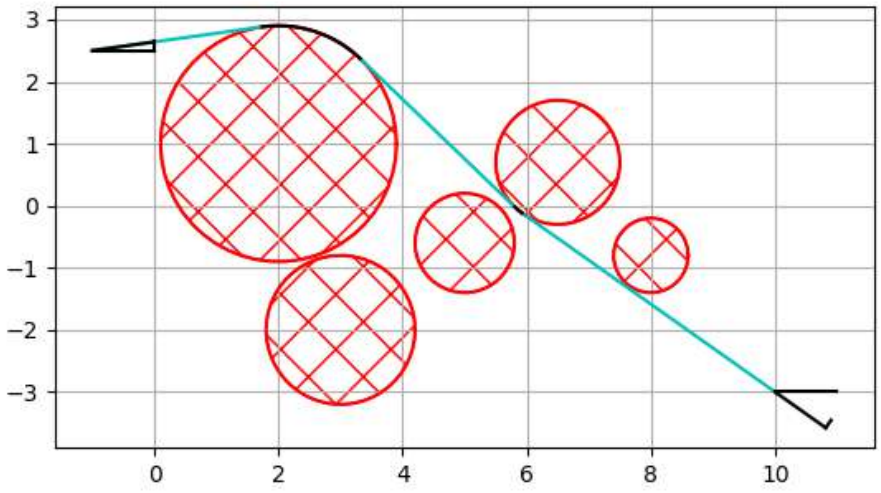

Рис. 3.32: Оптимальная траектория для пяти препятствий.

Меняем положение окружности с самым маленьким радиусом.

- Точка старта: $(x = -1,\ y = 2.5,\ \varphi = 0)$
- Точка финиша: $(x = 10,\ y = -3,\ \varphi = 0)$
- Окружность 1: $(x_c = 2,\ y_c = 1,\ r = 1.5)$
- Окружность 2: $(x_c = 3,\ y_c = -2,\ r = 1.0)$
- Окружность 3: $(x_c = 6.5,\ y_c = 0.5,\ r = 0.8)$
- Окружность 4: $(x_c = 5,\ y_c = -0,5,\ r = 0.6)$
- Окружность 5: $(x_c = 8,\ y_c = -2,\ r = 0.4)$
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

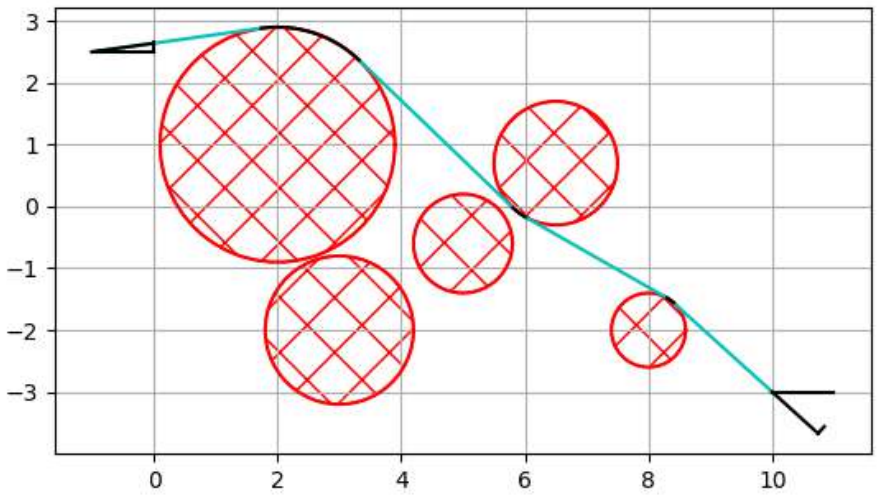

Рис. 3.33: Оптимальная траектория для пяти препятствий.

Теперь меняем начальную и конечную точки.

- Точка старта: $(x = -1,\ y = -2,\ \varphi = 0)$
- Точка финиша: $(x = 10,\ y = 2,\ \varphi = 0)$
- Окружность 1: $(x_c = 2,\ y_c = 1,\ r = 1.5)$
- Окружность 2: $(x_c = 3,\ y_c = -2,\ r = 1.0)$
- Окружность 3: $(x_c = 6.5,\ y_c = 0.5,\ r = 0.8)$
- Окружность 4: $(x_c = 5,\ y_c = -0,5,\ r = 0.6)$
- Окружность 5: $(x_c = 8,\ y_c = -2,\ r = 0.4)$
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

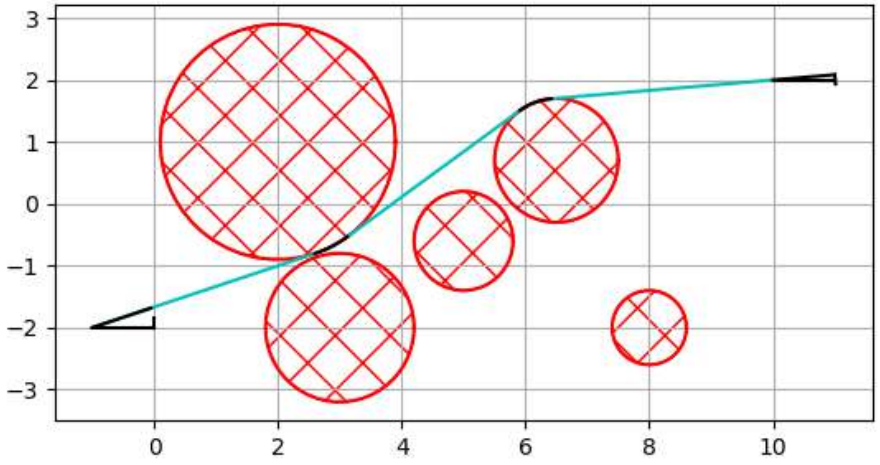

Рис. 3.34: Оптимальная траектория для пяти препятствий.

Оптимальные траектории для шести препятствий. Во всех экспериментах с тремя препятствиями точки старта и финиша будут одинаковыми.

- Точка старта: $(x = -1,\ y = 0,\ \varphi = 0)$
- Точка финиша: $(x = 13,\ y = 0,\ \varphi = 0)$
- Окружность 1: $(x_c = 2,\ y_c = 1,\ r = 1.5)$
- Окружность 2: $(x_c = 11,\ y_c = 2,\ r = 1.2)$
- Окружность 3: $(x_c = 3,\ y_c = -2,\ r = 1.0)$
- Окружность 4: $(x_c = 7,\ y_c = 0.5,\ r = 0.8)$
- Окружность 5: $(x_c = 5,\ y_c = -0.5,\ r = 0.6)$
- Окружность 6: $(x_c = 8,\ y_c = -0.7,\ r = 0.4)$
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

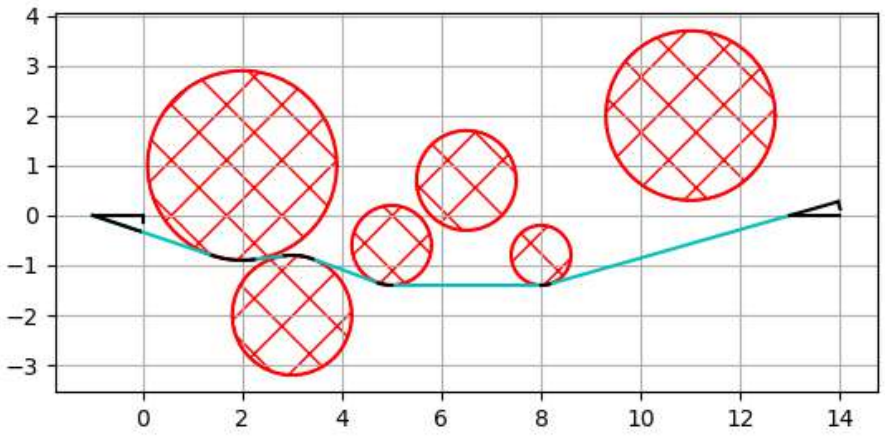

Рис. 3.35: Оптимальная траектория для шести препятствий.

Поднимем выше на $0,5$ единиц окружность с радиусом $0,8$.

- Точка старта: $(x = -1,\ y = 0,\ \varphi = 0)$
- Точка финиша: $(x = 13,\ y = 0,\ \varphi = 0)$
- Окружность 1: $(x_c = 2,\ y_c = 1,\ r = 1.5)$
- Окружность 2: $(x_c = 11,\ y_c = 2,\ r = 1.2)$
- Окружность 3: $(x_c = 3,\ y_c = -2,\ r = 1.0)$
- Окружность 4: $(x_c = 7,\ y_c = 1,\ r = 0.8)$
- Окружность 5: $(x_c = 5,\ y_c = -0.5,\ r = 0.6)$
- Окружность 6: $(x_c = 8,\ y_c = -0.7,\ r = 0.4)$
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

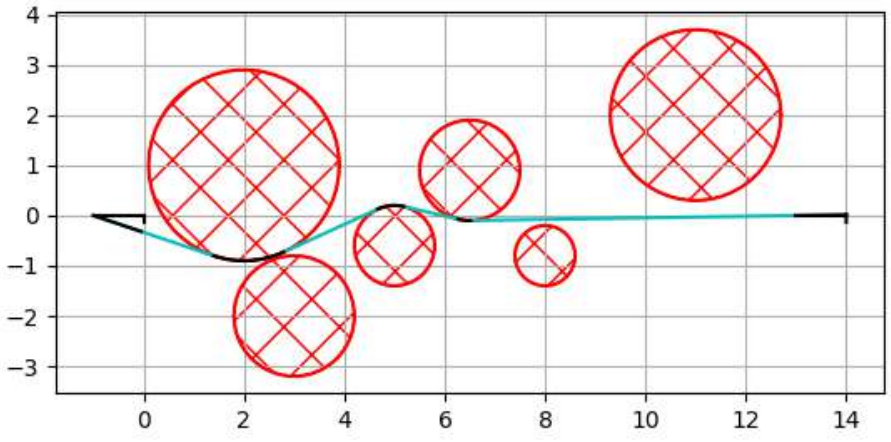

Рис. 3.36: Оптимальная траектория для шести препятствий.

Теперь опустим на 1 единицу окружность с радиусом $1,2$.

- Точка старта: $(x = -1,\ y = 0,\ \varphi = 0)$
- Точка финиша: $(x = 13,\ y = 0,\ \varphi = 0)$
- Окружность 1: $(x_c = 2,\ y_c = 1,\ r = 1.5)$
- Окружность 2: $(x_c = 11,\ y_c = 2,\ r = 1.2)$
- Окружность 3: $(x_c = 3,\ y_c = -2,\ r = 1.0)$
- Окружность 4: $(x_c = 7,\ y_c = 1,\ r = 0.8)$
- Окружность 5: $(x_c = 5,\ y_c = -0.5,\ r = 0.6)$
- Окружность 6: $(x_c = 8,\ y_c = -0.7,\ r = 0.4)$
- Ограничение $u_{\max} = 1$
- Расстояние между колесами $b = 1$

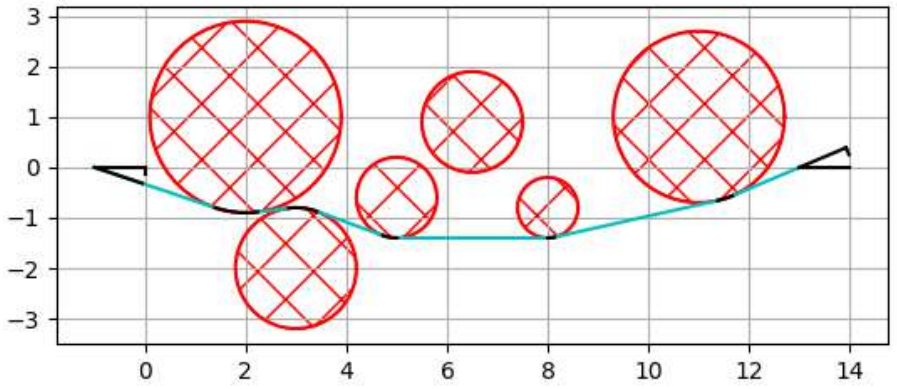

Рис. 3.37: Оптимальная траектория для шести препятствий.

Оптимальные траектории для двенадцати препятствий. В связи с большим количеством препятствий здесь и далее приведем только координаты начальных и конечных точек.

- Точка старта: $(x = 0,\ y = 0,\ \varphi = 0)$
- Точка финиша: $(x = 20,\ y = 0,\ \varphi = 0)$

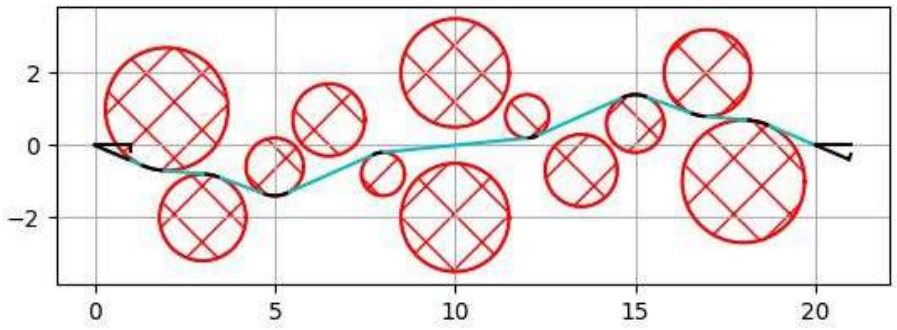

Рис. 3.38: Оптимальная траектория для двенадцати препятствий.

- Точка старта: $(x = 0,\ y = -1,\ \varphi = 0)$
- Точка финиша: $(x = 20,\ y = 1,\ \varphi = 0)$

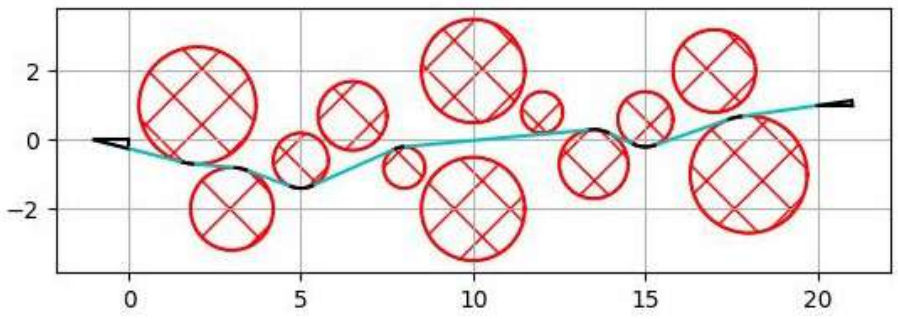

Рис. 3.39: Оптимальная траектория для двенадцати препятствий.

- Точка старта: $(x = -1,\ y = -1,\ \varphi = 0)$
- Точка финиша: $(x = 20,\ y = 1,\ \varphi = 0)$

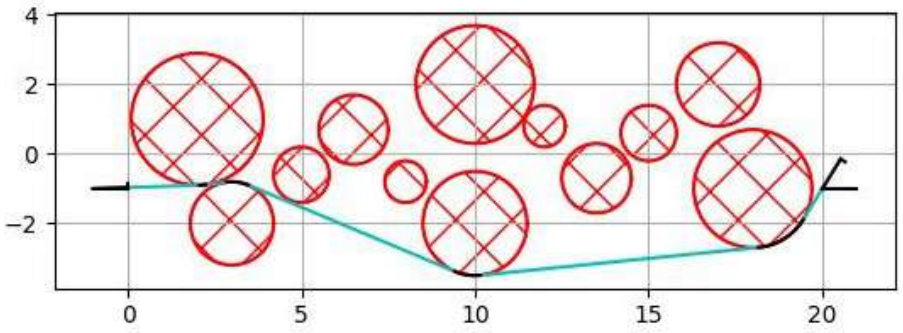

Рис. 3.40: Оптимальная траектория для двенадцати препятствий.

- Точка старта: $(x = -1,\ y = 2.5,\ \varphi = 0)$
- Точка финиша: $(x = 20,\ y = -1,\ \varphi = 0)$

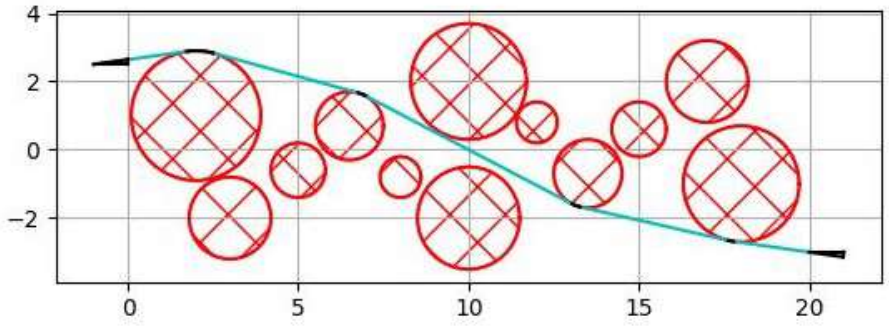

Рис. 3.41: Оптимальная траектория для двенадцати препятствий.

Оптимальные траектории для шестнадцати препятствий.

- Точка старта: $(x = -1,\ y = 3,\ \varphi = 0)$
- Точка финиша: $(x = 20,\ y = -3,\ \varphi = 0)$

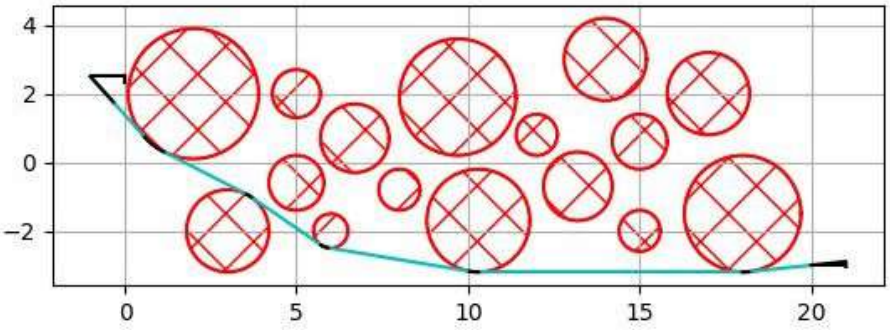

Рис. 3.42: Оптимальная траектория для шестнадцати препятствий.

- Точка старта: $(x = -1,\ y = 2.5,\ \varphi = 0)$
- Точка финиша: $(x = 20,\ y = -3,\ \varphi = 0)$

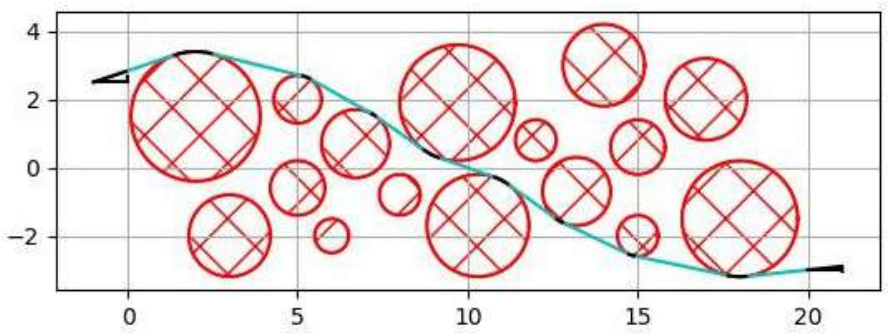

Рис. 3.43: Оптимальная траектория для шестнадцати препятствий.

- Точка старта: $(x = -1,\ y = 1,\ \varphi = 0)$
- Точка финиша: $(x = 22,\ y = -1,\ \varphi = 0)$

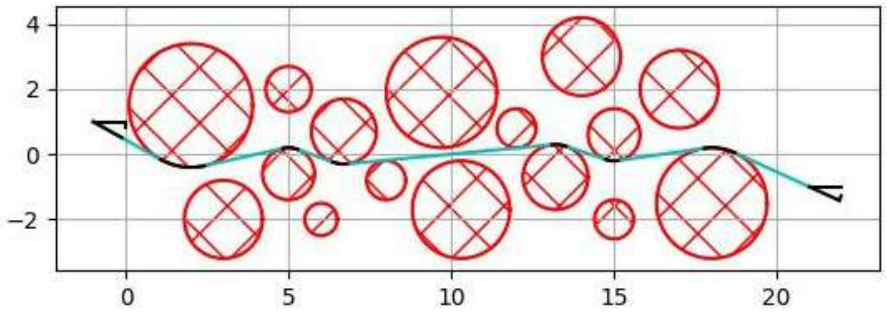

Рис. 3.44: Оптимальная траектория для шестнадцати препятствий.

- Точка старта: $(x = -1,\ y = -1,\ \varphi = 0)$
- Точка финиша: $(x = 23,\ y = 4,\ \varphi = 0)$

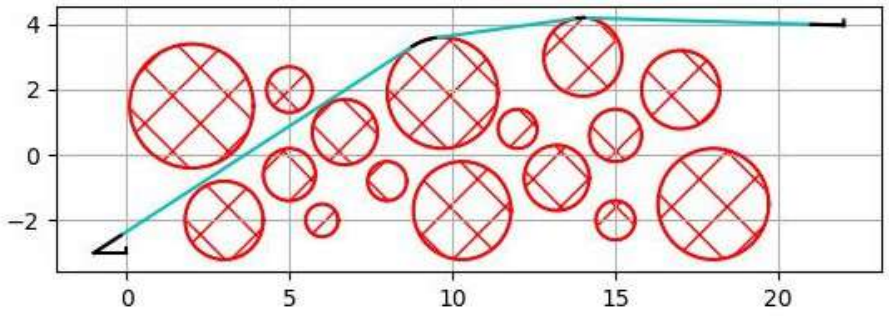

Рис. 3.45: Оптимальная траектория для шестнадцати препятствий.

- Точка старта: $(x = -1,\ y = -1,\ \varphi = 0)$
- Точка финиша: $(x = 22,\ y = 3,\ \varphi = 0)$

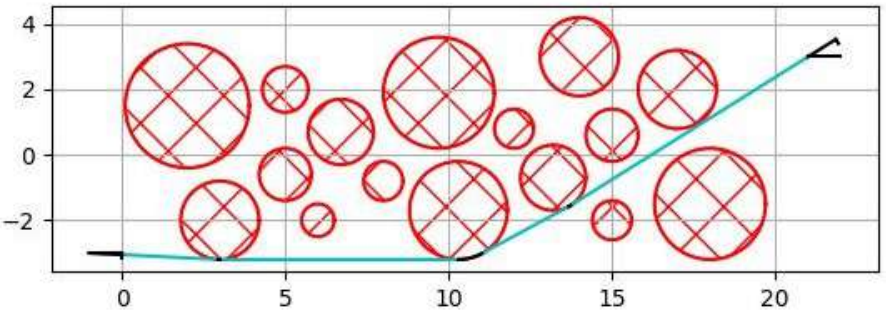

Рис. 3.46: Оптимальная траектория для шестнадцати препятствий.

3.5 Заключение

В данном разделе рассмотрена задача оптимального управления мобильным роботом при наличии фазовых координатв предположении, что робот периодически получает данные о местоположении от системы позиционирования в пределах заданной системы координат. Предполагается, что эти координаты подвержены погрешностям аппроксимации с известными границами. Основываясь на теории синтеза оптимального управления для систем второго порядка и методе

нахождения кратчайших путей на связном графе, предложен алгоритм коррекции траектории движения робота в режиме реального времени. Этот алгоритм гарантирует, что робот достигнет заданной точки с заданной точностью. Реализация и анализ численных экспериментов, основанных на этом подходе, также рассмотены в данном разделе.

Также в данном разделе диссертационной работы описан созданный автором и реализованный в виде законченной программы алгоритм автоматического поиска оптимального пути между двумя заданными точками с набором фазовых ограничений в виде кругов различного радиуса, которые не пересекаются между собой.

Исследование кинематической модели показывает, что, несмотря на то, что в общем виде кратчайший с геометрической точки зрения путь не совпадает с быстрейшим путем при движении из одной заданной точки в другую, тем не менее, он совпадает с одним из локальных минимумов в мультимодальной (имеющей несколько локальных минимумов) задаче поиска кратчайшего пути. Это означает, что для поиска быстрейшего движения здесь можно пользоваться теми же методами, что и для поиска кратчайшего пути, а именно, поиском оптимального пути на графе возможных переходов, с той разницей, что для оценки «длины» этого пути необходимо вычислять время движения по соответствующим отрезкам.

Данное свойство связано со структурой модели и определением управляющих воздействий в ней. Такая модель является достаточно сильно упрощенной, поскольку использует управление скоростями колес и не учитывает динамику переходных процессов, а управляющие воздействия кусочно-постоянны. Тем не менее, во многих задачах робототехники такая упрощенная модель оказывается полезной, например, при исследовании движения и планировании пути для гусеничных роботов.

Дальнейшее развитие алгоритма позволит учитывать препятствия произвольной формы, в том числе с негладкой границей и невыпуклые. Похожие алгоритмы в настоящее время используются в компьютерных играх, однако, там используется поиск кратчайшего пути (в том числе с обходом полигональных препятствий), и не уделяется внимания быстродействию.

Кроме того, в связи с возросшим интересом к применению методов численной оптимизации к задачам оптимального управления, результаты данной работы могут быть интересны с точки зрения оценки качества тех или иных методов оптимизации, поскольку позволяют найти абсолютный минимум времени движения и сравнить его с результатами, получаемыми численно, на такой модельной задаче.

Глава 4

Определение управления мобильным роботом при движении в режиме реального времени

В этой главе рассмотрены вопросы управления реальным мобильным роботом в реальном времени. Для решения задачи необходимо учитывать как статические, так и динамические ограничения. Решение только классическими методами, затруднено большим количеством вычислений, которые требуется проводить на борту робота, поэтому они не могут быть реализованы на беспилотном объекте в процессе его эксплуатации. Однако использование классических методов совместно с эвристическими методами или нейросетями позволяет свести задачу поиска управления и траектории движения робота к задаче, требующей меньшего числа вычислений, и реализовать поиск решения в режиме онлайн.

Другим важным вопросом, связанным с реальными роботами, является вопрос навигации мобильного робота. Этот вопрос также рассмотрен в данной работе, результаты приведены в дальнейших разделах.

В этой главе представлены новые методы поиска управления и траектории, а также навигации мобильного робота.

4.1 О практических задачах робототехники

Процесс поиска управления на реальном мобильном роботе включает несколько этапов, от сбора данных до финального исполнения команд. Этот процесс должен учитывать ограничения динамики, неопределенность в состоянии и окружающей среде, а также вычислительные возможности аппаратной системы. Поиск

управления на реальном роботе – это комплексная задача, требующая интеграции сенсоров, локализации, планирования и оптимального управления. Современные методы, такие как модель прогностического управления (Model predictive control, MPC) [176, 186–196], модель прогностического интегрального управления (Model Predictive Path Integral, MPPI) [197–199] и алгоритм быстрорастущего случайного дерева (Rapidly exploring random tree, RRT) [200–202], позволяют эффективно решать эту задачу, но требуют корректной модели динамики и надёжной аппаратной реализации.

Основные этапы поиска управления на реальном роботе следующие:

1. сбор данных с сенсоров;
2. локализация и построение карты;
3. планирование пути;
4. генерация траектории;
5. вычисление управляющих воздействий;
6. подача управления на приводы.

На реальном роботе управление реализуется следующим образом:

- на бортовой компьютер (например, Jetson Nano, Raspberry Pi, NVIDIA Orin) поступают данные от датчиков;
- управление вычисляется с помощью одного из методов (например, MPC, MPPI, ПИД-регулятора и т.д.);
- команды управления отправляются на низкоуровневый контроллер (например, Teensy, ESC и др.);
- контроллер преобразует команды в сигналы для двигателей и руля.

Целью управления беспилотного транспортного средства (БТС) является его перевод из начального состояния в заданное конечное состояние или осуществление (отслеживание) заданного программного движения БТС. Генерируемые законы управления должны обеспечивать требуемые показатели качества (точность, быстродействие и т.п.) по всем управляемым координатам с учётом заданных ограничений на управления и состояния БТС. Кроме того, практически важно, чтобы эти законы управления были оптимальными по отношению

к заданному функционалу качества. Однако в ряде случаев этого недостаточно и требуется генерировать законы управления, обеспечивающие достижение цели управления в широком классе неопределённости модели динамики БТС. С другой стороны система управления должна обеспечить управление с учетом динамических ограничений в реальном времени, параметры которых заранее не известны.

Автоматизация колесных мобильных роботов была предметом многих исследований [203–205]. Много работ посвящено планированию траекторий для мобильного робота. Среди них следует отметить методы планирования на основе теории графов [206–209], методы на основе оптимизации [210–214], методы с использованием искусственного интеллекта [215–218], методы на основе потенциальных полей [156, 212]. Но, несмотря на фундаментальные и прикладные исследования, задачи по управлению ими до конца не решены. Задачи изучения траекторий мобильного робота и расчета управления актуальны в связи с возрастающими требованиями к точности таких систем, необходимостью нахождения оптимального управления, обеспечения плавных динамических перемещений.

Чтобы добраться до пункта назначения, избегая препятствий, транспортное средство – мобильный робот должен двигаться к конечной точке под определенным углом в системе глобального позиционирования для выполнения полезных действий, например, проведения обследования, размещения груза и т. д. Расчет оптимальной траектории в сложных средах со статическими и динамическими ограничениями требует оценки всего пространства возможных состояний и поиска наилучшего решения.

В работе рассматривается задача оптимизации управления мобильным роботом в реальном времени. Обычно управление в реальном времени использует подход прогнозирования траектории. Сравнение известных методов на конкретной задаче приведено в разделе 4.5. Решаются терминальные задачи и получаются траектории на небольших участках, среди которых выбираются наиболее подходящие для решения задачи.

Основная трудность этой задачи состоит в том, что в процессе нахождения оптимального управления часто не известны некоторые дополнительные параметры, обеспечивающие оптимальное качество прохождения траектории. В результате учета реальных параметров, которые могут быть определены при автономной работе объекта, расчетная оптимальная траектория, как правило, не совпадает с реальной. Для формального описания этой проблемы вводится вектор дополнительных параметров, значения которых измеряются объектом управления в процессе управления. На основе этих дополнительных пара-

метров формируются штрафные функции, определяющие выбор траекторий из нескольких решаемых терминальных задач. Для выбора параметров используется специальная функция выбора.

Наиболее общий подход к решению задач управления в реальном времени основан на использовании метода предсказания траектории, обеспечивающей возможность добиться прогресса на пути достижения цели [220]. Основная проблема реализации этого подхода состоит в том, что при моделировании необходимо знать не только информацию о модели, но и информацию об окружающей среде. В работе [221] предлагается генерировать множество траекторий с различным управлением и выбирать их по дополнительному функционалу, который учитывает внешние факторы, в том числе и динамические фазовые ограничения. Данный функционал определяется разработчиком системы управления и ставится на борт объекта до реализации алгоритма управления. Построение такой функции требует существенного предварительного анализа внешней среды и сопряжено с возможностью внесения корректировки при выборе траектории. В результате учета реальных параметров статических и динамических ограничений, которые могут быть определены при автономной эксплуатации объекта, рассчитанная траектория оптимизации, как правило, не совпадает с реальной.

Известные подходы к предотвращению столкновений для мобильных роботов можно условно разделить на две категории: глобальные и локальные. Преимущество глобальных методов, таких как дорожная карта, декомпозиция клеток и методов потенциального поля [222] заключаются в том, что полная траектория от начальной точки до целевой точки может быть вычислена в реальном времени, но они не подходят для быстрого предотвращения препятствий.

Локальные или реактивные подходы [221, 223–226], используют лишь небольшую часть пространства состояния. Это приводит к очевидному недостатку, что они не могут вычислять оптимальные решения для всей траектории. Большинство этих локальных подходов генерируют команды движения для робота на двух отдельных этапах [223], [226]. На первом этапе определяется желаемое направление движения. На втором этапе генерируются команды управления, приводящие к движению в нужном направлении. Вычисление направления движения должно соответствовать глобальному направлению с учетом ограничений. При синтезе управления БТС при таком подходе, необходимо решить, на сколько достаточны ресурсы по управлению для достижения тех или иных областей пространства из текущего или заданного состояния, и какой вид имеет траектория движения, если такие области достижимы.

В диссертационной работе рассматривается подход, при котором функция для отбора траекторий определяется эволюционными методами символьной ре-

грессии [227–229]. Каждая функция управления рассчитана на свой собственный вектор состояния и решает задачу достижения своей цели управления, обеспечивая общую цель и общий критерий качества управления. В результате необходимо выбрать функцию управления с учетом новых параметров, например, динамических ограничений (другого робота), рассчитываемых в режиме реального времени на борту робота. Для управления используется глубокое сверточное и трансферное обучение для определения характеристик дороги и принятия правильных решений по управлению. Сверточная сеть, объединенная с сетью прямого распространения, отображает характеристики дороги и характеристики маневров, в результате выход нейросетевого алгоритма, получает решение по управлению по всем управляемым координатам с учётом заданных ограничений на управления и состояния в реальном времени с учетом оптимальности по отношению к заданному функционалу качества.

Тем не менее, все эти подходы основаны на очень точной позиции из внешнего источника (либо GPS, либо захват движения). Рассматривается автономное вождение в сложных условиях с активным вождением, используя только внутренние датчики, такие как камеры и инерциальный измерительный блок. Есть несколько способов решения этой проблемы. Существует много подходов, в которых для обеспечения точного положения используются камеры [179, 231], LIDAR [230] или другие комбинации датчиков [233]. Эти системы обычно обеспечивают положение относительно сгенерированной карты. Эти методы имеют тенденцию быть вычислительно дорогими. Альтернативный метод обеспечения абсолютной позиции использует глубокие нейронные сети для прямой регрессии позиции оценки в районе, посещенном ранее [234]. Однако этот метод локализации еще недостаточно точен, чтобы его можно было напрямую использовать для контроля.

В данной работе применяется модель прогностического интегрального управления. Метод основан на оптимизации функции стоимости, которая определяет, где на поверхности пути должно двигаться транспортному средству. Поэтому поверхность должна кодировать текущее и будущее положение дороги, препятствий, пешеходов, и другие транспортные средства. Сеть обучена так, чтобы стоимость была самая низкая в центре дорожки, и выше от центра. Эта карта стоимости может затем непосредственно вводиться в алгоритм прогнозного управления моделью. Прогнозное управление моделью работает путем чередования оптимизации и выполнения: сначала оптимизируется последовательность управления с разомкнутым контуром (генерируются тысячи траекторий после чего считается их стоимость на основе функционала, конкретный вид и параметры которого зависят от ручной настройки и личного опыта разработчика),

затем первое управление в этой последовательности выполняется транспортным средством, и затем принимается обратная связь о состоянии, и весь процесс повторяется.

Параметры оператора стоимости в алгоритме MPPI настраиваются вручную, а перспективным подходом является применение методов обучения с подкреплением. Большинство предыдущих работ с MPC [146, 176, 186–196] фокусируется на задачах стабилизации или отслеживания траектории. Ключевая разница между классическими MPC и MPC для обучения с подкреплением, является то, что задачи обучения с подкреплением являются сложными задачами. Сложность целей в задачах обучения с подкреплением заключается в том, что увеличиваются вычислительные затраты на оптимизацию, так как оптимизация должна происходить также в режиме реального времени. В основном обучение с подкреплением используется при экспериментах в виртуальной среде [235–238].

Алгоритм MPPI позволяет в реальном времени обрабатывать сложные нелинейные функции динамики и стоимости (тысячи траекторий), но, как и классические MPC алгоритм страдает от ухудшения устойчивости, когда моделируемая динамика отличается от истинной динамики транспортного средства. Стратегией борьбы с этой проблемой может быть настройка параметров модели объекта управления и параметров функции стоимости с помощью эволюционного метода обучения с подкреплением, а также применения глубокой сверточной нейронной сети для понимания сцены в реальном времени.

Проблема реализации решения задачи оптимального управления [17] состоит не только в том, что ее решение представляет собой функцию времени и непосредственная реализация этого решения в реальном объекте приводит к разомкнутой системе управления, но и в том, что при решении задачи оптимального управления, часто с целью облегчения процесса оптимизации, используются упрощенные математические модели объекта управления. Основная причина использования упрощенных моделей заключается в желании уменьшить размерность задачи и снизить трудоемкости вывода и учета всех особенностей объекта управления. К тому же очевидно, что ни одна математическая модель не описывает точно реальный объект управления. Например, математические модели автомобилеподобного робота практически никогда не учитывают пробуксовку колес, различное трение у колес, движение юзом и т.п., а эти особенности движения достаточно существенно влияют на точность моделирования реального объекта.

Считается, что решение задачи оптимального управления приводит к получению программного оптимального управления и программной оптимальной траектории. Для реализации оптимального управления необходимо построить

систему стабилизации движения объекта по программной траектории. Эта система управления за счет обратной связи и должна компенсировать неточности модели и особенности ее движения в окружающей среде. Следует отметить, что в этом случае в математической модели, используемой при решении задачи оптимального управления, не учитывалась система стабилизации движения объекта по программной траектории, поэтому оптимальная траектория не должна являться оптимальной для объекта с системой стабилизации. Из сказанного следует, что точная математическая модель объекта управления является важной особенностью решения задачи оптимального управления.

В последнее время для различных задач аппроксимации и распознавания используются искусственные нейронные сети (ИНС). Он представляют собой универсальную математическую модель некоторой функции, с заданной структурой и большим количеством настраиваемых (обучаемых) параметров [239]. Различные типы ИНС способны аппроксимировать любую функцию по множеству ее значений и значений аргументов, совокупность которых называется обучающей выборкой. В настоящей работе рассматривается решение задачи оптимального управления для математической модели объекта, описываемой искусственной нейронной сетью.

Использование нейронных сетей в качестве математических моделей объектов управления сегодня целесообразно из-за бурного развития робототехнических устройств. Разнообразие конструкций всевозможных роботов требует непрерывной разработки математических моделей для них, что, в свою очередь требует существенных затрат времени. Применение ИНС позволит автоматизировать процесс получения математической модели робота. В данном случае модель на основе ИНС в задачах управления заменяет реальный объект, поэтому ее применение избавляет разработчиков от субъективных решений по выбору модели объекта и возможности внесения ошибок при аналитических преобразованиях.

Другой необходимостью использования ИНС в качестве математических модели объектов управления является проблема навигации. В настоящий период основные навигационные системы автономных роботов построены на основе данных, получаемых с датчиков, расположенных на борту робота, это ультразвуковой дальномер, лазерный датчик лидар и совокупность видеокамер. Проверка навигационных расчетов на борту не производится, а может только корректироваться внешним оборудованием. К сожалению, данная система датчиков очень чувствительна к шумам и не зависит от работоспособности устройств, поэтому системы дублирования не имеют смысла. Одним из способов решения проблемы является использование на борту автономного робота его точной математиче-

ской модели. Тогда существенное разногласие между данными навигационной системы и результатами моделирования позволяют установить наличие ошибок навигации. При точной модели, полученной на основе аппроксимации экспериментальных данных, автономный робот может использовать систему навигации (датчики) для анализа внешней окружающей среды и уточнения одометрии от модели.

Идентификация мобильного робота — это процесс определения его состояния, параметров и характеристик на основе данных, полученных от сенсоров и других источников информации. Эта задача является ключевой для обеспечения автономности и эффективности работы роботов в различных средах.

Идентификация мобильного робота является важной областью исследований и разработок в робототехнике. С развитием технологий сенсоров и алгоритмов обработки данных возможности идентификации становятся все более точными и эффективными. Это открывает новые горизонты для применения мобильных роботов в различных сферах, от промышленности до повседневной жизни.

Существует несколько методов идентификации мобильных роботов, среди которых можно выделить:

- в *моделях на основе физических законов* используются динамические модели для предсказания поведения робота, они учитывают физические параметры, такие как масса, инерция и силы, действующие на робот;
- *методы фильтрации*, например, фильтр Калмана [240] и его расширенные версии, позволяют оценивать состояние системы на основе наблюдаемых данных с учетом шума и неопределенности;
- *сенсорные данные* [241] используют информацию от различных сенсоров (лидаров, камер, ультразвуковых датчиков) для определения положения и ориентации робота в пространстве;
- *системы машинного обучения* [242] применяют алгоритмы машинного обучения для анализа данных и предсказания состояния робота, они включают в себя нейронные сети или методы обучения с подкреплением.

Для реализации идентификации используются различные технологии и инструменты:

- лидары, камеры, инерциальные измерительные устройства (IMU), GPS и другие сенсоры, которые собирают данные о положении и окружающей среде;

- платформы для разработки, которые предоставляют инструменты и библиотеки для обработки данных и реализации алгоритмов идентификации, например, Robot Operating System (ROS) – широко используемая платформа для разработки робототехнических приложений, включая навигацию, или Gazebo – симулятор, который позволяет тестировать навигационные алгоритмы в виртуальной среде;

- разработка алгоритмов для фильтрации, обработки и анализа данных, получаемых от сенсоров.

В диссертационной работе полученная модель, разработанного метода идентификации, используется для тестирования новых алгоритмов и технологий.

Навигация мобильных роботов включает в себя методы и технологии, позволяющие роботам перемещаться в окружающей среде. Навигация должна быть интегрирована с системами управления роботом для обеспечения плавного и эффективного выполнения задач. Основные аспекты навигации мобильных роботов включают в себя: локализацию на карте, планирование пути, избежание препятствий, интеграция с системами управления следования траектории, программное обеспечение и платформы.

Локализация — это процесс определения позиции робота относительно окружающей среды. Существуют различные методы локализации, например, *одометрия* – используются данные от колесных датчиков для оценки перемещения робота; *фильтр Калмана* – это статистический метод, который комбинирует данные от различных сенсоров для более точной оценки положения;*метод одновременной локализации и картографирования* (Simultaneous Localization and Mapping, SLAM) позволяет роботу одновременно строить карту окружающей среды и определять свое местоположение на этой карте.

В диссертационной работе предложен новый метод на основе метода ASLAM (раздел 1.6). В данном методе применяется функция стоимости, с помощью которой убираются предсказанные траектории на основе лучшей вероятности локализации робота с помощью датчиков на борту и близости к целевой точке, далее выбирается траектория, которая гарантирует достижение цели и локализацию мобильного робота на карте. Основное преимущество метода состоит в том, что с его помощью одновременно решается задача управления и задача навигации.

Планирование пути включает в себя вычисление оптимального маршрута от начальной точки до целевой с учетом препятствий и других факторов. Здесь можно отметить алгоритмы поиска (A* [122, 243], Dijkstra [121] и другие алгоритмы, которые помогают находить кратчайший путь) и метод динамического

планирования, который учитывает изменения в окружающей среде в реальном времени (например, перемещение препятствий).

Для обучения роботов обнаруживать и избегать препятствий на своем пути используют ультразвуковые, инфракрасные или лазерные датчиков для обнаружения объектов, а также различные алгоритмы избегания, например, метод потенциальных полей или алгоритмы на основе машинного обучения.

Генерация траекторий для мобильных роботов — это важный аспект навигации, который включает в себя создание последовательности движений, позволяющей роботу достичь заданной цели. Выбор метода генерации траектории зависит от конкретной задачи, характеристик среды и требований к роботу. Комбинирование различных методов также может привести к более эффективным и адаптивным решениям в области навигации мобильных роботов. Существует несколько методов генерации траекторий, каждый из которых имеет свои преимущества и недостатки. Вот некоторые из наиболее распространенных способов:

1. Алгоритмы поиска пути (используются графы или сетки):
    - Алгоритм A* использует эвристическую функцию для нахождения кратчайшего пути. Он эффективен для планирования маршрута в статических средах.
    - Алгоритм Дейкстры находит кратчайший путь от начальной точки до всех остальных вершин графа. Он менее эффективен, чем A*, но гарантирует нахождение оптимального решения.
    - RRT строит дерево возможных путей, исследуя пространство. Хорошо подходит для высокоразмерных пространств.
2. Методы на основе потенциальных полей (используют концепцию сил, действующих на робота). Робот рассматривается как объект, подверженный влиянию сил притяжения (к цели) и отталкивания (от препятствий). Траектория генерируется как результат взаимодействия этих сил.
3. Полиномиальные и сплайн-интерполяции (используются для создания гладких траекторий):
    - Кубические сплайны используются для интерполяции между контрольными точками, создавая плавные кривые.
    - Полиномиальные траектории генерируются с помощью полиномов, которые учитывают начальные и конечные условия (позиция, скорость, ускорение).

4. Методы оптимизации (позволяют находить траектории, минимизирующие определенные функции (например, время, расстояние или потребление энергии)):
    - генетические алгоритмы используют эволюционные принципы для поиска оптимальных траекторий;
    - методы градиентного спуска оптимизируют траекторию, минимизируя заданную функцию стоимости;
5. Динамическое планирование (учитывают изменения в окружающей среде в реальном времени):
    - Подход DWA (Dynamic Window Approach) генерирует траектории на основе динамических ограничений робота и препятствий в окружающей среде.
    - Метод Teb (Timed Elastic Band) использует временные параметры для генерации гибких и адаптивных траекторий.
6. Машинное обучение (можно генерировать траектории на основе больших объемов данных):
    - нейронные сети используются для предсказания оптимальных траекторий на основе обучающих данных;
    - обучение с подкреплением позволяет роботу обучаться выбирать действия, которые максимизируют награду в процессе выполнения задач.

Функция выбора определяет, какие действия должен предпринять робот в зависимости от текущей ситуации и состояния окружающей среды. Эта функция включает: определение различных состояний робота (например, «поиск пути», «избежание препятствий», «выполнение задачи»), реакцию на события (например, обнаружение препятствия, достижение цели), логику принятия решений (использование деревьев решений для выбора действий в зависимости от состояния, использование конечных автоматов для управления состояниями робота, комбинирование различных уровней управления (например, стратегический, тактический и оперативный уровни), определение приоритетов для различных задач (например, избегание препятствий может иметь более высокий приоритет, чем выполнение задачи), интеграцию (после разработки функций управления и выбора необходимо интегрировать их в единую систему), а также проведение

тестирования в реальных или симулированных условиях для проверки эффективности разработанных функций и оптимизацию алгоритмов на основе полученных данных. Для интеграции требуется тестирование в различных сценариях, настройка параметров управления для достижения оптимальной производительности, обеспечение устойчивости системы к изменениям в окружающей среде.

Таким образом, для поиска управления мобильного робота в реальном времени необходимо провести идентификацию модели, сгенерировать траектории, выбрать оптимальную траекторию и настроить робот движению по оптимальной траектории.

4.2 Кинематическая модель мобильного робота

Для проверки теоретических данных проводились эксперименты на симуляторе, а также на реальном роботе (см. рис. 4.1) в Робототехническом центре ФИЦ ИУ РАН (см. рис. 4.2).

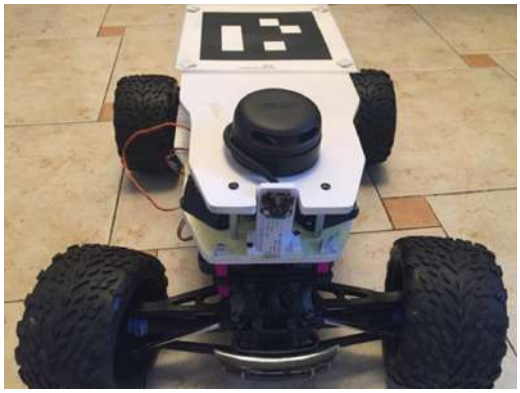

Рис. 4.1: Мобильный робот

Робот имеет шасси геометрии Аккермана [226].

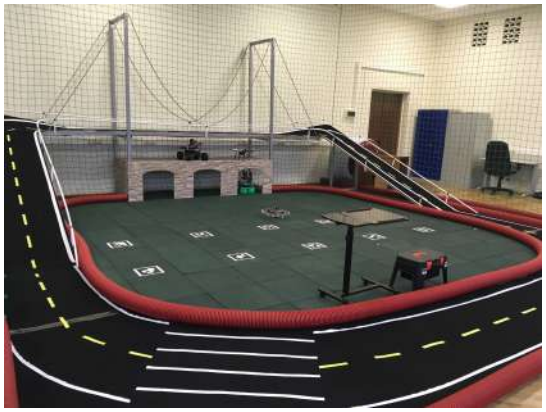

Рис. 4.2: Робототехнический центр ФИЦ ИУ РАН

Шасси с геометрией Аккермана – это конструкция передней подвески и рулевого управления автомобилей, обеспечивающая оптимальный поворот колёс при движении по дуге. Эта геометрия названа в честь английского изобретателя Рудольфа Аккермана, хотя идея была разработана ранее инженерами, а Р. Аккерман запатентовал ее применение в XIX веке. Особенность такого шасси заключается в том, что два передних колеса служат для рулевого управления, а два задних колеса – для отслеживания.

Суть геометрии Аккермана заключается в следующем: при повороте автомобиля внутреннее переднее колесо (ближе к центру поворота) должно поворачиваться под большим углом, чем внешнее переднее колесо. Это необходимо для того, чтобы оба колеса катились по концентрическим окружностям с общим центром поворота, минимизируя боковое скольжение и износ шин (рис. 4.3).

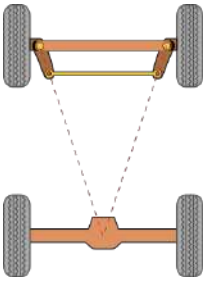

Рис. 4.3: Схематичное расположение колес у шасси геометрии Аккермана

Пусть робот поворачивает вокруг центра O. Обозначим:

- d – колёсная база (расстояние между передней и задней осями);

- l – колея (расстояние между центрами передних колёс);
- R – радиус поворота от центра O до центра задней оси;
- α_R – угол поворота внутреннего колеса;
- α_L – угол поворота внешнего колеса.

Кинематическая схема управления движением мобильного робота с шасси геометрии Аккермана показана на рис. 4.4.

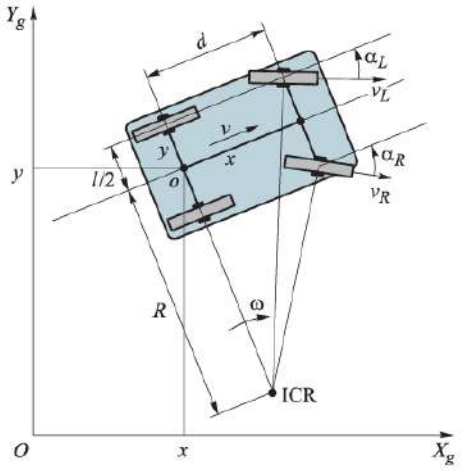

Рис. 4.4: Кинематическая схема управления движением

Согласно геометрии Аккермана, продолжения осей передних колёс должны пересекаться в одной точке – центре поворота ICR. Тогда справедливы следующие соотношения

$$\operatorname{ctg}\delta_i = \frac{R - l/2}{d}$$
$$\operatorname{ctg}\delta_o = \frac{R + l/2}{d}$$

Из этих формул следует ключевое соотношение Аккермана:

$$\operatorname{ctg}\delta_i - \operatorname{ctg}\delta_o = \frac{l}{d}$$

Это уравнение определяет идеальную геометрию Аккермана – при заданной колее и базе, углы поворота колёс должны удовлетворять этому соотношению.

Например, пусть: $d = 2.7$ м, $l = 1.5$ м, $R = 10$ м. Тогда

$$\operatorname{ctg}\delta_i = \frac{10 - 0.75}{2.7} = \frac{9.25}{2.7} \approx 3.426 \Rightarrow \delta_i \approx \operatorname{arcctg}(3.426) \approx 16.3^\circ$$
$$\operatorname{ctg}\delta_o = \frac{10 + 0.75}{2.7} = \frac{10.75}{2.7} \approx 3.981 \Rightarrow \delta_o \approx \operatorname{arcctg}(3.981) \approx 14.1^\circ$$

Таким образом, внутреннее колесо должно поворачиваться примерно на 16.3°, а внешнее – на 14.1°.

Элементы, реализующие геометрию Аккермана, – это

1. рулевая трапеция – система тяг, формирующая нужные углы поворота;
2. поворотные кулаки с рычагами под определённым углом;
3. рулевая рейка или рулевой механизм (червячная передача, шестерня-рейка и пр.).

В зависимости от настройки углов можно выделить несколько вариантов шасси Аккермана. Обзор вариантов приведен в таблице 4.1.

Таблица 4.1: Типы геометрии поворота в зависимости от настройки

Тип	**Описание**
Идеальная Аккерман	Углы удовлетворяют соотношению $\operatorname{ctg}\delta_i - \operatorname{ctg}\delta_o = l/d$. Теоретически оптимально.
Недостаточная Аккерман (Reverse Ackermann)	$\delta_o > \delta_i$ — внешнее колесо поворачивается сильнее; используется в гоночных авто для компенсации кренов и нагрева шин.
Полная Аккерман	Реализована максимально близко к теории.
Параллельный поворот	$\delta_i = \delta_o$ — неэффективно, вызывает проскальзывание.

Применяется шасси в роботах, в гражданских автомобилях для комфорта и экономичности, в гоночных машинах для улучшения сцепления на высоких скоростях, в грузовиках, спецтехнике и пр. [245, 246].

В экспериментах к данной работе рассматривалось движение мобильного робота (рис. 4.1) с шасси геометрии Акерманна, заданное следующей системой дифференциальных уравнений:

$$\begin{cases} \dot{x} = u_1 \cos\theta, \\ \dot{y} = u_1 \sin\theta, \\ \dot{\theta} = \frac{u_1}{H} \operatorname{tg} u_2, \end{cases} \tag{4.1}$$

где x, y – координаты центра задней оси мобильного робота; u_1 – его линейная скорость; θ – угол его поворота относительно оси x; u_2 – угол поворота (положительный против часовой стрелки); H – расстояние между передней и задней осями колес мобильного робота.

Кинематическая схема управления движением мобильного робота, используемого в натурных экспериментах, показана на рис. 4.5.

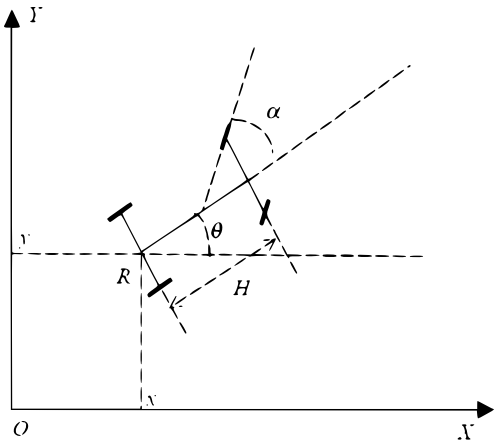

Рис. 4.5: Кинематическая схема управления движением

На рис. 4.5 α – угол поворота руля, θ – угол его поворота относительно оси x, H – расстояние между передней и задней осями колес мобильного робота, x, y – координаты.

4.3 Постановка задачи

В исследованиях, проведенных в данной работе, рассматривалась следующая постановка задачи.

Задан объект исследования, поведение которого описывается некоторой системой из n обыкновенных нелинейных дифференциальных уравнений

$$\dot{x} = f(x, u), \tag{4.2}$$

где $x \in \mathbb{R}^n$ – вектор состояния объекта управления, $u \in U \subseteq \mathbb{R}^m$ – вектор управления, $x = [x_1, x_2, \ldots, x_n]^T$; $u = [u_1, u_2, \ldots, u_m]^T$; $f(x, u) = [f_1(x, u), \ldots, f_n(x, u)]^T$; $m \leq n$.

В общем случае предполагается, что для значений времени t, принадлежащих интервалу $[0, t^+]$, заданы соответствующие начальные условия

$$x(0) = x^0 \in \mathbb{R}^n$$

и время $t_f \in [0, t^+]$ попадания в терминальные условия определяется из соотношения

$$x(t_f) = x^{t_f} \in \mathbb{R}^n. \tag{4.3}$$

Заданы статические

$$\varphi_i(x^j) \leq 0, \quad i = 1, \ldots, r, \quad j = \overline{1, n}, \quad r \leq m,$$

и динамические

$$\gamma(x^p, x^s) \leq 0, \quad p = 1, \ldots, K-1, \quad s = \overline{p+1, K}, \quad r \leq m$$

фазовые ограничения. Здесь r и K – соответствующее количество ограничений.

В рассматриваемых условиях задача состоит в том, чтобы определить такой управляющий вектор $u(x)$ и соответствующий вектор фазовых координат $x(t)$, которые в течение всей миссии достигали бы конечных условий (4.3) и обеспечивали экстремальное значение для некоторого фиксированного функционала, конкретного вида и физический смысл которых определяется природой исследуемого объекта:

$$J = \int_0^{t_f} f_0(x, u)dt \to \min. \tag{4.4}$$

В данной работе функция $f_0(x, u)$ предполагалась тождественно равной 1, то есть решалась задача быстродействия. Таким образом, функционал качества имеет вид

$$J = \int_0^{t_f} dt = t_f \to \min. \tag{4.5}$$

В натурных экспериментах использовался мобильный робот с шасси геометрии Аккермана. Напомним его математическую модель

$$\begin{cases} \dot{x} = u_1 \cos\theta, \\ \dot{y} = u_1 \sin\theta, \\ \dot{\theta} = \frac{u_1}{H} \operatorname{tg} u_2, \end{cases} \tag{4.6}$$

где x, y – координаты центра задней оси мобильного робота; u_1 – его линейная скорость; θ – угол его поворота относительно оси x; u_2 – угол поворота (положительный против часовой стрелки); H – расстояние между передней и задней осями колес мобильного робота.

Статические и динамические фазовые ограничения представлялись окружностями радиусов R_1 и R_2 соответственно:

- статические фазовые ограничения

$$\varphi_i(x, y) = R_{1i}^2 - (x_i^* - x)^2 - (y_i^* - y)^2 \leq 0, \tag{4.7}$$

  где x_i^*, y_i^* – координаты центра препятствия;

- динамические фазовые ограничения

$$\gamma(x_p, y_p, x, y) = R_2^2 - (x_p^* - x)^2 - (y_p^* - y)^2 \leq 0, \tag{4.8}$$

  где x^p, y^p – координаты другого робота.

Задавались терминальные условия x^f, y^f и θ^f, а также предельное время процесса управления t^+.

Пример статических ограничений представлен на рис. 4.6.

Рис. 4.6: Робототехнический центр ФИЦ ИУ РАН со статическими ограничениями в виде конусов

Для учета ограничений добавим к функционалу штраф за нарушение ограничений:

$$\tilde{J} = t^f + \sum_{i=1}^{r} \alpha_i \vartheta(\varphi(x, y)) + \sum_{i=1}^{K} \beta_i \vartheta(\eta(x^p, x, y)).$$

Трасса для проведения экспериментов, расположенная в Робототехническом центре ФИЦ ИУ РАН (рис. 4.2), имеет сложную конструкцию. Она состоит из

прямолинейных участков, подъема, спуска и криволинейных участков, замыкающих трассу в единый неразрывный трек.

Такая трасса сложна для преодоления роботом в связи с разноплановостью конструкции, поэтому для учета всех препятствий, встречающихся на треке, используются штрафные функции, которые замедляют процесс прохождения по трассе, и поэтому они добавляются в функционал качества.

В работе учитывались следующие штрафные функции (σ_i – весовые коэффициенты , $i = 1, \ldots, 3$):

– отклонение от первого приближения оптимальной траектории

$$y_1 = \sigma_1 \min_{\tilde{x},\tilde{y},\tilde{\theta}} |\Omega(x, y, \theta) - \Omega(\tilde{x}, \tilde{y}, \tilde{\theta})|,$$

где $\Omega(x)$ – первое приближение оптимальной траектории с учетом заданных конечного и начального состояний и статических ограничений;

– штраф за превышение допустимой скорости, на участках, где скорость ограничена

$$y_2 = \sigma_2 \vartheta(|u_1| - v^+),$$

где v^+ – допустимая скорость на данном участке траектории, ϑ – функция Хевисайда

$$\vartheta(a) = \begin{cases} 0, & a < 0, \\ 1, & a \geq 0; \end{cases}$$

– положение высоты робота над плоскостью

$$y_3 = \sigma_3 \operatorname{arctg}(\frac{z_{i+1} - z_i}{\Delta z}),$$

где Δz – шаг измерения.

Базовой функцией при синтезе функционала отбора использовалась сумма штрафов

$$F(Z_k) = y_1 + y_2 + y_3.$$

На рис. 4.7 показан вид на трассу с камеры мобильного робота.

Рис. 4.7: Вид на трассу с камеры робота в РТЦ ФИЦ ИУ РАН

4.4 Идентификация модели мобильного робота

Идентификация модели мобильного робота — это процесс определения математической модели, которая наиболее точно описывает динамику и кинематику поведения данного робота на основе экспериментальных данных. Другими словами, это процедура построения или уточнения модели робота, используя измеренные входы (например, команды управления) и выходы (например, положение, скорость, ориентация).

Основные этапы идентификации модели включают в себя [247]

- *выбор структуры модели*: кинематическая модель (например, для дифференциального привода) или динамическая (учитывает массу, моменты инерции, силы трения), линейная или нелинейная, детерминированная или стохастическая;

- *сбор данных*: проводится серия экспериментов с роботом, измеряются входы (например, напряжения на двигателях, заданные скорости) и выходы (например, реальные координаты, показания IMU, одометрия);

- *обработка данных*: фильтрация шума, синхронизация сигналов, подготовка выборки для обучения/идентификации;

- *оценку параметров модели*, которая проводится с целью найти такие параметры модели (например, коэффициенты трения, радиус колес, расстояние между колесами), которые минимизируют ошибку между предсказанным

и реальным поведением робота. Для достижения цели используются методы параметрической идентификации: метод наименьших квадратов, рекуррентные алгоритмы, фильтр Калмана, машинное обучение и т.д.;

- *валидацию модели*: проверка адекватности модели на новых данных, анализ точности предсказания, устойчивости, обобщаемости.

При решении задачи идентификации на реальном мобильном роботе идентификации подлежат, например, следующие параметры: реальный радиус колес (может отличаться от паспортных значений), база между колесами (расстояние между правым и левым колесом), коэффициент проскальзывания, параметры двигателей (постоянная времени, КПД), влияние трения, инерции, наклона поверхности и т.п.

Методы идентификации делятся на пассивные методы, использующие данные, собранные при нормальной работе робота, и активные, для которых специально формируются тестовые воздействия (например, робот движется по заданной траектории).

Задача идентификации обычно формулируется как задача оценки параметров модели системы, которая обладает существенными признаками реальной системы и характеризует ее динамические свойства в форме, удобной для синтеза управления.

Рассмотрим различные подходы к идентификации динамических моделей мобильных роботов, включая параметрические, вероятностные и машинно-обучающие методы [248].

Пусть модель движения робота имеет вид

$$\dot{x} = f(x, u; \theta) + w,$$

где $x \in \mathbb{R}^n$ – состояние робота (координаты, ориентация, скорость), $u \in \mathbb{R}^m$ – управление, $\theta \in \mathbb{R}^p$ – неизвестные параметры модели, w – шум.

Требуется найти оценку $\hat{\theta}$ по наблюдениям $D = \{(x_i, u_i)\}_{i=1}^N$, минимизируя ошибку между моделью и реальным движением.

Один из методов идентификации модели – *регрессия* – это статистический метод, позволяющий находить зависимость между входными переменными (например, управляющие сигналы) и выходными (например, положение, скорость робота).

В контексте идентификации робота

- *входы* – это команды управления (напряжения двигателей, целевые скорости),

- *выходы* – это измеренные данные (скорости, ускорения, координаты),

- *цель* заключается в нахождении параметров модели, которые связывают входы и выходы.

Линейная регрессия подходит, если динамика робота приблизительно линейна в интересующем диапазоне, нужна простая, интерпретируемая модель, нужно быстро обучить модель на реальных данных.

Пусть имеется робот с колесом, приводимым в движение двигателем. Требуется понять, как напряжение $u(t)$ влияет на угловую скорость $\omega(t)$. Для этого предположим простую модель

$$\omega(t) = a \cdot \omega(t-1) + b \cdot u(t-1) + \varepsilon,$$

здесь $\omega(t)$ – текущая угловая скорость, $\omega(t-1)$ — скорость в предыдущий момент времени, $u(t-1)$ – управляющее напряжение на двигателе, a, b – неизвестные коэффициенты, которые нужно идентифицировать, ε – случайная ошибка (шум модели).

Это – линейная модель по параметрам, и её можно обучить с помощью линейной регрессии. Для этого воспользуемся следующим алгоритмом.

Алгоритм

Шаг 1. Сбор данных

1. Подать разные напряжения $u(t)$ на двигатель.
2. Измерить скорость $\omega(t)$ (например, с помощью энкодера).
3. Записать пары данных: $\big(\omega(t-1), u(t-1)\big) \to \omega(t)$.

Шаг 2. Подготовка данных для регрессии

На основе временных рядов

$$\omega = [\omega_0, \omega_1, \ldots, \omega_T], \quad u = [u_0, u_1, \ldots, u_T].$$

Создать обучающую выборку

$$X = \begin{bmatrix} \omega_0 & u_0 \\ \omega_1 & u_1 \\ \vdots & \vdots \\ \omega_{T-1} & u_{T-1} \end{bmatrix}, \quad y = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_T \end{bmatrix}$$

Шаг 3. Обучение линейной регрессии

Решить задачу минимизации

$$\min_{a,b} \sum_{t=1}^{T} \left(\omega_t - \left(a \cdot \omega_{t-1} + b \cdot u_{t-1}\right)\right)^2$$

Шаг 4. Получение модели

После обучения модель принимает вид

$$\hat{\omega}(t) = a \cdot \omega(t-1) + b \cdot u(t-1)$$

Теперь полученную модель можно использовать для предсказания скорости вращения колеса на основе предыдущего состояния и управляющего сигнала.

Ещё один метод – это *метод наименьших квадратов* [249, 250]. Он требует полного набора данных (оффлайн), предполагает, что ошибки имеют нулевое среднее и постоянную дисперсию. Его преимущества заключаются в простоте реализации и высокой скорости. Однако есть и недостатки – он чувствителен к шуму и работает только для линейных зависимостей.

Для линейной зависимости $y = \Phi\theta + e$, где Φ – матрица регрессоров, решение

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T y.$$

Альтернативным вариантом является *рекуррентный метод наименьших квадратов* [249]. Он применяется, когда данные поступают потоком (онлайн-идентификация)

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\left[y(t) - X^T(t)\hat{\theta}(t-1)\right],$$

где $K(t)$ — матрица усиления, зависящая от ковариационной матрицы оценок.

Фильтры Калмана [240, 249] позволяют оценивать параметры в условиях нелинейности и подходят для обработки шума и обновления параметров в реальном времени. Математическая формализация имеет следующий вид:

$$p(\theta|D) \propto p(D|\theta)p(\theta)$$

Для сложных нелинейных моделей используется *обучение на основе последовательностей* [251, 252]

$$y_t = f_{\text{NN}}(u_{t-k:t}, y_{t-k:t-1})$$

Также обучение можно проводить на больших объемах данных.

Для идентификации используют *вероятностный метод* [253]

$$y_t \sim \mathcal{GP}(m(t), k(t, t')),$$

преимущества, которого заключаются в возможность интерпретации доверительных интервалов и учете неопределенности.

Возможно *сочетать физические законы и нейросети*:

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda \mathcal{L}_{\text{physics}}$$

Для устойчивости к выбросам или шуму могут применяться: Lasso-регрессия, Ridge-регрессия, M-оценки.

Для идентификации можно использовать *пары данных «управление – состояние»*, собранные в ходе эксперимента или симуляции

$$\min_{\theta} \|y_{\text{real}} - y_{\text{sim}}(\theta)\| \tag{4.9}$$

Эта задача может решаться с помощью градиентных методов, эволюционных алгоритмов или байесовских подходов.

В таблице 4.2 показано сравнение различных методов идентификации для кинетической модели мобильного робота.

Таблица 4.2: Сравнение методов идентификации модели мобильного робота

Метод	**Тип модели**	**Данные**	**Скорость**	**Время**
МНК	Линейная	Вход/выход	Быстро	реальное
Рекур. МНК	Нелинейная	Вход/выход	Медленно	нет
Фильтр Калмана	Нелинейная	Вход/выход + шум	Средне	реальное
Регрессия	Нелинейная	Вход/выход	Средне	нет
Нейросеть	Нелинейная	Физика + данные	Долго	нет

На рис. 4.8 приведено сравнение разных методов идентификации. При сравнении учитывалось снижение ошибки отклонения от реальной модели.

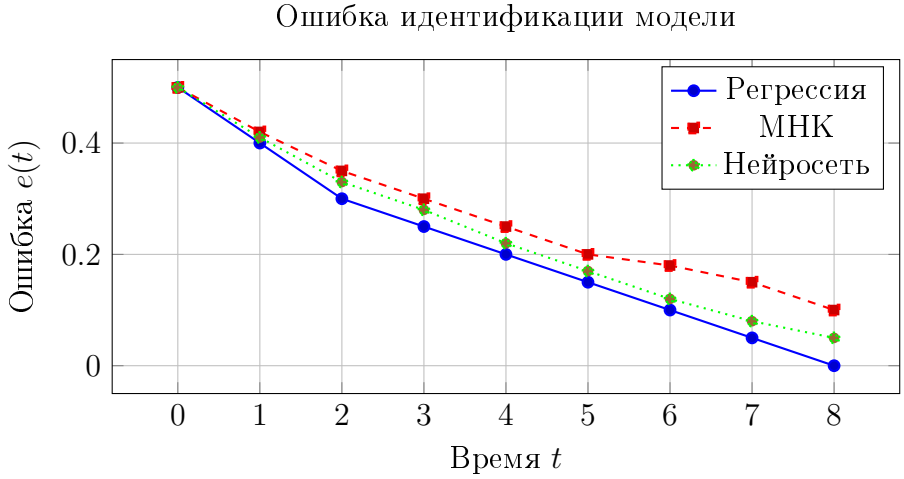

Рис. 4.8: Снижение ошибки идентификации во времени для разных методов

В работе применен традиционный метод построения моделей динамических объектов — реализация процедуры идентификации с использованием регрессионных модельных структур в сочетании с использованием нейросетевой модели. Регрессия напрямую связана с задачей прогнозирования величины выходного сигнала $y(t)$ на основе информации, полученной путем измерения других величин φ_i, $i = \overline{1,d}$, содержащих информацию о поведении системы в прошлом.

Такие модели используются из-за простоты реализации, возможности применения стандартных библиотек (NumPy, Scikit-learn, MATLAB). Они подходят для онлайн-идентификации и могут быть расширены до нелинейных моделей с использованием нелинейных регрессоров.

Однако существуют и недостатки: для реализации метода ребуется точное измерение всех необходимых переменных (например, ускорений), метод чувствителен к шуму и неточным измерениям, не все модели легко представимы в линейной регрессионной форме, для сложных систем может потребоваться большое количество данных.

Задача идентификации состоит в нахождении функции $g(\varphi)$ такой, чтобы оценка $\hat{y} = g(\varphi)$ удовлетворяла некоторому критерию и была прогнозом величины $\mathbf{y}(t)$. Функция $g(\varphi)$ – функция регрессии, должна быть некоторым методом оценена по экспериментальным данным, т.е. $g(\varphi) = g(\varphi, \mathbf{q})$, где $\mathbf{q}$ – вектор настраиваемых параметров.

Задача состоит в сборе необходимого количества экспериментальных данных во всем рабочем диапазоне системы $Z^L = \{\mathbf{u}(t), \mathbf{y}(t)\}$, $t = \overline{1, L}$. Полнота и достоверность полученных данных во многом определяют качество идентификации.

Таким образом, динамический объект может быть представлен в следующем виде

$$\mathbf{y}(t|\mathbf{g}) = g(\varphi(t, \mathbf{q}), \mathbf{q}). \tag{4.10}$$

В диссертационной работе представлен новый метод идертификации модели мобильного робота – совместное использование кинетической модели и нейросети. Управление u_1 определяется нейросетью, затем найденные значения подставляются в книетическую модель (4.1), из которой находились координаты (x, y).

Пример идентификации параметров робота с шасси Аккермана на основе линейной регрессии

В данном разделе приводится пример идентификации кинематической модели робота с шасси Аккермана (рис. 4.5) с использованием метода линейной регрессии. На основе экспериментальных данных о скоростях и углах поворота руля строится линейная модель, позволяющая оценить параметры, такие как база шасси и коэффициенты передачи.

Угловая скорость ω выражается как

$$\omega = \frac{u_1 \cdot \mathrm{tg}(\alpha)}{H}$$

Данное уравнение является нелинейным по α, однако при малых углах α можно использовать приближение $\mathrm{tg}(\alpha) \approx \alpha$, и модель становится линейной

$$\omega \approx \frac{u_1 \cdot \alpha}{H}$$

Пусть имеется набор экспериментальных данных $\mathcal{D} = \{(v_i, \delta_i, \omega_i)\}_{i=1}^{N}$, где каждое наблюдение соответствует измеренной линейной скорости, углу руля и угловой скорости.

Таблица 4.3: Экспериментальные данные

i	u_{1i} (м/с)	α_i (рад)	ω_i (рад/с)
1	1.0	0.1	0.252
2	1.5	0.1	0.378
3	1.0	0.2	0.498
4	2.0	0.1	0.504
5	1.5	0.2	0.745

Требуется оценить параметр H, используя модель

$$\omega_i = \theta \cdot (u_{1i}\alpha_i) + \varepsilon_i,$$

где $\theta = 1/H$, а ε_i — ошибка измерения.

Таким образом, задача сводится к линейной регрессии без свободного члена

$$\mathbf{y} = \mathbf{X}\theta + \boldsymbol{\varepsilon},$$

где

– $\mathbf{y} = [\omega_1, \omega_2, \ldots, \omega_N]^\top$ – вектор наблюдений угловой скорости,
– $\mathbf{X} = [v_1\delta_1, v_2\delta_2, \ldots, v_N\delta_N]^\top$ – вектор признаков,
– θ – скалярный параметр, подлежащий оценке.

Оценку параметра θ проведем методом наименьших квадратов

$$\hat{\theta} = \frac{\mathbf{X}^\top \mathbf{y}}{\mathbf{X}^\top \mathbf{X}}$$

После вычисления $\hat{\theta}$, оценка базы шасси находится как:

$$\hat{H} = \frac{1}{\hat{\theta}}$$

Для экспериментальных значений вычислим

$$\mathbf{X} = [0.1, 0.15, 0.2, 0.2, 0.3]^\top, \quad \mathbf{y} = [0.252, 0.378, 0.498, 0.504, 0.745]^\top$$

$$\mathbf{X}^\top \mathbf{y} = 0.5058, \quad \mathbf{X}^\top \mathbf{X} = 0.2025$$
$$\hat{\theta} = \frac{0.5058}{0.2025} \approx 0.476, \quad \hat{H} = \frac{1}{0.476} \approx 2.10 \text{ м}$$

$$\begin{aligned}\mathbf{X}^\top \mathbf{y} &= 0.1 \cdot 0.252 + 0.15 \cdot 0.378 + 0.2 \cdot 0.498 + 0.2 \cdot 0.504 + 0.3 \cdot 0.745 \\ &= 0.0252 + 0.0567 + 0.0996 + 0.1008 + 0.2235 = 0.5058 \\ \mathbf{X}^\top \mathbf{X} &= 0.1^2 + 0.15^2 + 0.2^2 + 0.2^2 + 0.3^2 \\ &= 0.01 + 0.0225 + 0.04 + 0.04 + 0.09 = 0.2025\end{aligned}$$

Тогда:

$$\hat{\theta} = \frac{0.5058}{0.2025} \approx 2.49(7), \quad \hat{H} = \frac{1}{0.2.49(7)} \approx 0.4004 \text{ м}$$

Таким образом, оценка базы шасси составляет около 0.4 метра.

На рис. 4.9 показана зависимость угловой скорости ω от признака $u_1\alpha$. Точками обозначены экспериментальные данные, а прямыми – результат линейной регрессии с нулевым сдвигом: $\omega = \hat{\theta} \cdot (u_1\alpha)$.

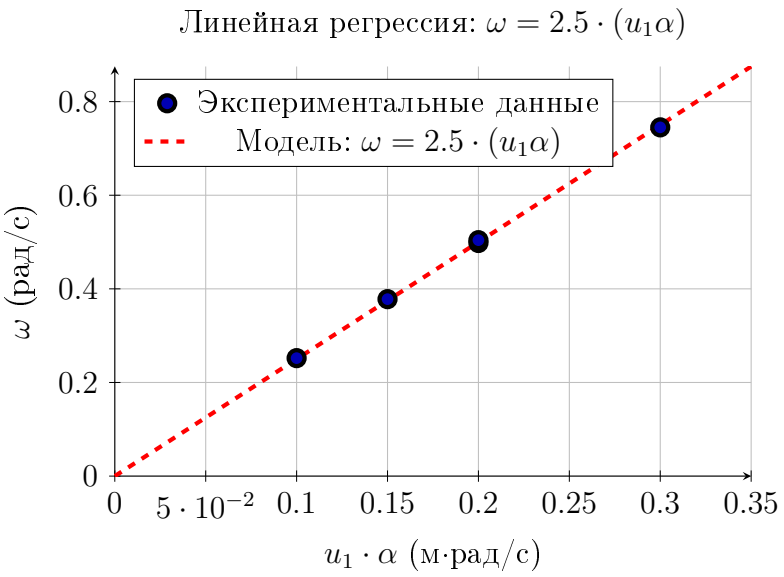

Рис. 4.9: Результаты линейной регрессии

На рис. 4.10 показаны остатки модели – разница между наблюдаемыми значениями угловой скорости ω_i и значениями, предсказанными моделью $\hat{\omega}_i = \hat{\theta} \cdot (u_{1i}\alpha_i)$.

Анализ остатков позволяет оценить качество подгонки и выявить возможные систематические ошибки.

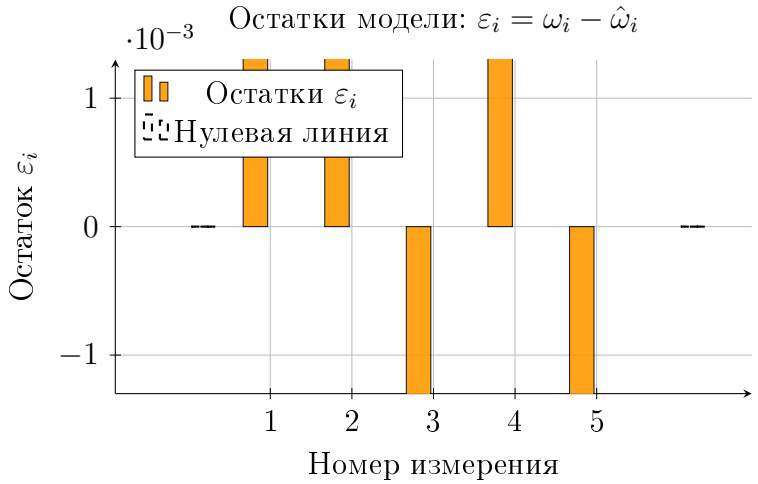

Рис. 4.10: Диаграмма остатков модели линейной регрессии

Полученная оценка базы шасси $\hat{H} \approx 0.4004$ м близка к реальному значению ($H = 0.4$ м). Остатки модели (рис. 4.10) имеют малую амплитуду (в пределах ± 0.005 рад/с), что свидетельствует о хорошем качестве аппроксимации в диапазоне малых углов. Однако заметна систематическая ошибка в точке 4 ($u_1 = 2.0$, $\alpha = 0.1$), где предсказанное значение несколько ниже наблюдаемого, что указывает на нелинейное поведение системы при повышенной скорости.

Таким образом, линейная регрессия представляет собой простой, но мощный инструмент для идентификации параметров мобильных роботов.

В данной работе предложен новый метод идентификации и построения управления — метод, использующий искусственные нейросети. Основа метода состоит в том, что правая часть системы управления рассматривается как искусственная нейронная сеть, обучая которую получаем необходимые параметры.

Математическая формализация нейронных сетей

Искусственный нейрон – это вычислительная единица, принимающая входной вектор $\mathbf{x} = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^n$, взвешивающая его с помощью весов $\mathbf{w} = (w_1, w_2, ..., w_n)^T \in \mathbb{R}^n$, добавляющая смещение $b \in \mathbb{R}$ и применяющая функцию активации $f : \mathbb{R} \to \mathbb{R}$.

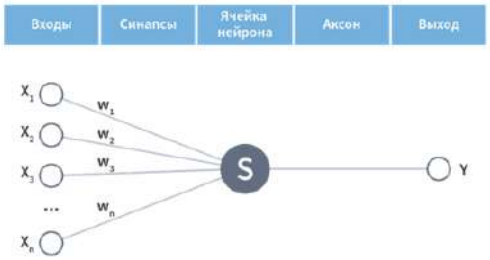

Рис. 4.11: Схематичное представление нейрона

Впервые концепция искусственного нейрона была предложена в 1943 году У. МакКаллоком и У. Питтсом [254].

Первый искусственный нейрон использовал ступенчатую активационную функцию единичного скачка – функцию Хевисайда.

Выход нейрона $y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) = f(\mathbf{w}^T\mathbf{x} + b)$. Нейронная сеть состоит из слоёв: входного, одного или нескольких скрытых и выходного.

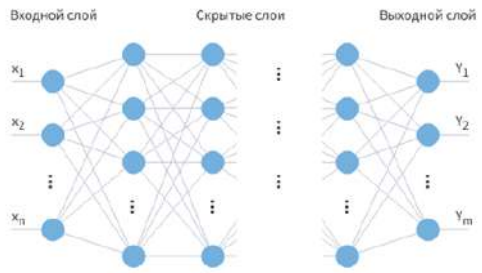

Рис. 4.12: Схематичное расположение слоев нейрона

Введем обозначения:

- N – количество слоев,
- $l^{[n]}$ – число нейронов на слое n,

- $\mathbf{a}^{[n]} \in \mathbb{R}^{l^{[n]}}$ – активации слоя n,
- $\mathbf{z}^{[n]} \in \mathbb{R}^{l^{[n]}}$ – линейные комбинации входов,
- $W^{[n]} \in \mathbb{R}^{l^{[n]} \times l^{[n-1]}}$ – матрица весов,
- $\mathbf{b}^{[n]} \in \mathbb{R}^{l^{[n]}}$ – вектор смещений,
- $f^{[n]}$ – функция активации.

В процессе обучения нейронных сетей [255] ключевую роль играют два этапа – *прямое распространение* (forward propagation) и *обратное распространение ошибки* (back propagation). Эти процессы позволяют модели делать предсказания и затем улучшать их за счёт корректировки весов на основе ошибки.

Прямое распространение – это процесс вычисления выходного значения нейронной сети на основе входных данных и текущих значений весов. Пусть на вход сети подаётся вектор $\mathbf{a}^{[0]} = \mathbf{x}$. Данные проходят через слои сети следующим образом

1. На каждом слое n вычисляется взвешенная сумма
$$\mathbf{z}^{(n)} = \mathbf{W}^{(n)}\mathbf{a}^{(n-1)} + \mathbf{b}^{(n)},$$
где $\mathbf{a}^{(n-1)}$ – выход предыдущего слоя (или вход для первого слоя).
2. Применяется функция активации
$$\mathbf{a}^{(n)} = f^{[n]}(\mathbf{z}^{(n)}),$$
где $f^{[n]}$ – например, ReLU, Sigmoid или Tanh.
3. Процесс повторяется до выходного слоя.

Целью прямого распространения является получение такого предсказания $\hat{\mathbf{y}} = \mathbf{a}^{[n]}(\mathbf{x}; \mathbf{W}, \mathbf{b})$, чтобы затем сравнить его с истинным значением $\mathbf{y}$ и вычислить функцию потерь
$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}).$$

Обратное распространение – это алгоритм вычисления градиентов функции потерь по отношению ко всем весам сети с использованием цепного правила дифференцирования. После процедуры прямого распространения вычислено значение функции потерь $\mathcal{L}$. Затем вычисляются частные производные потерь по весам
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(n)}}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(n)}}$$

для каждого слоя n, начиная с последнего и двигаясь к первому (отсюда – «обратное» распространение).

Вычисляем ошибки на выходе

$$d\mathbf{z}^{[N]} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[N]}}.$$

Для $n = N, N-1, ..., 1$ вычисляем

$$dW^{[n]} = d\mathbf{z}^{[n]}(\mathbf{a}^{[n-1]})^T,$$

$$d\mathbf{b}^{[n]} = d\mathbf{z}^{[n]},$$

$$d\mathbf{z}^{[n-1]} = (W^{[n]})^T d\mathbf{z}^{[n]} \odot f'^{[n-1]}(\mathbf{z}^{[n-1]}),$$

где $\odot$ — поэлементное умножение (Hadamard product).

Градиенты накапливаются и используются для обновления параметров

$$W^{[n]} := W^{[n]} - \eta \cdot dW^{[n]}, \quad \mathbf{b}^{[n]} := \mathbf{b}^{[n]} - \eta \cdot d\mathbf{b}^{[n]},$$

где $\eta > 0$ — скорость обучения (learning rate).

Целью обратного распространения является минимизация функции потерь, постепенно корректируя веса сети в направлении, противоположном градиенту (градиентный спуск).

Для примера рассмотрим один нейрон:
– вход: $x = 2$;
– вес: $w = 1$;
– предсказание: $\hat{y} = w \cdot x = 2$;
– истинное значение: $y = 4$;
– функция потерь: $\mathcal{L} = (\hat{y} - y)^2 = (2-4)^2 = 4$;
– активация: ReLu.

Вычислим градиент по весу

$$\frac{\partial \mathcal{L}}{\partial w} = 2(\hat{y} - y) \cdot x = 2(2-4) \cdot 2 = -8$$

Обновим вес (при $\eta = 0.1$):

$$w_{\text{new}} = w - \eta \cdot (-8) = 1 + 0.1 \cdot 8 = 1.8$$

Теперь предсказание будет ближе к правильному значению.

Таким образом, прямое распространение необходимо для получения предсказания и вычисления ошибки, а обратное позволяет эффективно вычислить градиенты и обновить веса, чтобы уменьшить ошибку. Вместе эти два процесса лежат в основе обучения нейронных сетей и позволяют им адаптироваться к данным, минимизируя ошибку предсказания.

Пример идентификации модели мобильного робота

Напомним постановку задачи. Задана математическая модель объекта управления

$$\dot{x} = f(x, u) \tag{4.11}$$

где x – вектор состояния, $x \in \mathbb{R}^n$, u – вектор управления, $u \in U \subseteq \mathbb{R}^m$, U – компактное множество, определяющее ограничение на управление.

Задано начальное состояние

$$x(0) = x^0. \tag{4.12}$$

Задано терминальное состояние

$$x(t_f) = x^{t_f}. \tag{4.13}$$

где t_f – время достижения терминального состояния, не задано, но ограничено, $t_f \leq t^+$.

Задан критерий качества

$$J = \int_0^{t_f} dt = t_f \to \min_{u \in U}. \tag{4.14}$$

В задаче необходимо найти управление, как функцию времени,

$$u = g(t) \in U, \tag{4.15}$$

такую, чтобы система

$$\dot{x} = f(x, g(t)). \tag{4.16}$$

имела частное решение из начального состояния (4.12), которое попадает в терминальное состояние (4.13) с оптимальным значением критерия качества (4.14).

Правые части системы (4.11) представляют собой искусственную нейронную сеть, на вход которой подаются текущий вектор состояния и вектор управления (x, u), а на выходе искусственной нейронной сети получаем вектор скорости движения объекта в пространстве состояний $\dot{x}$.

Прогнозирующая модель динамики модели мобильного робота $g(\varphi(t,\mathbf{q}),\mathbf{q})$ может быть реализована на нейросетевой модельной структуре, имеющей следующее математическое представление

$$g_i(\varphi(t,\mathbf{q}),\mathbf{q}) = \hat{y}_i(t|\mathbf{q}) = \hat{y}_i(t|w,W) = F_i\left(\sum_{j=1}^{n_h} W_{ij} f_j\left(\sum_{j=1}^{n_\varphi} w_{jl}\varphi_l + w_{j0}\right) + W_{i0}\right), \tag{4.17}$$

где $f_j(x) = \tanh(x)$ – активационная функция нейронов скрытого слоя; $F_i(x) = ax$, $a = const$ – активационная функция нейронов выходного слоя; n_φ – размерность регрессионного вектора (число входов ИНС); n_h – число нейронов в скрытом слое; $\mathbf{q}$ – вектор настраиваемых параметров нейронной сети, включающий весовые коэффициенты и нейронные смещения (w_{jl}, W_{ij}).

При использовании выражения (4.17) оптимизация параметров $\mathbf{q}$ представляет собой отображение множества экспериментальных данных на множество параметров модели (4.10). Традиционно используемым критерием оптимальности является среднеквадратичная ошибка прогнозирования

$$\sigma_L(\mathbf{q}, Z^L) = \frac{1}{2L}\sum_{k=1}^{L}\left(y(k) - \hat{y}(k|\mathbf{q})\right)^2. \tag{4.18}$$

Таким образом, идентификация параметров модели (4.10) состоит в нахождении вектора параметров $\mathbf{q}$, минимизирующих критерий (4.18)

$$\hat{\mathbf{q}} = argmin_{\mathbf{q}}\sigma_L(\mathbf{q}, Z^L).$$

Мы ищем управляющий вектор в виде

$$u = \begin{cases} h(x^* - x), & \text{if} \quad \|x - y^*\| > r, \\ h(x^* - x, y^* - x), & \text{иначе}, \end{cases} \tag{4.19}$$

где x^* – является координатой центральной линии, y^* – является координатой центра ограничения, r – заданный параметр.

Для описания математической модели искусственной нейронной сети введем в рассмотрения вектор входных сигналов

$$v^T = [x \vdots u]^T \tag{4.20}$$

Представим математическую модель искусственной нейронной сети типа прямого перцептрона, имеющего N слоев

$$f(x,u) = z^N \tag{4.21}$$

где

$$z^j = \alpha_j \circ y^j \tag{4.22}$$

$$y^j = A^j z^{j-1} + b^j, \tag{4.23}$$

$$\alpha_j \circ y^j = \begin{pmatrix} \alpha_j(y_1^j) \\ \dots \\ \alpha_j(y_{n_j}^j) \end{pmatrix} \tag{4.24}$$

A^j – матрица весовых коэффициентов на слое j, b^j – вектор смещения на слое j, $dim(A^j) = n_j \times n_{j-1}$, $dim(b^j) = n_j$, α_j – функция активации на слое j, $j = \overline{1, N}$.

$$z^0 = \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} \tag{4.25}$$

$n_0 = n + m$, $n_N = n$.

Для обучения искусственной нейронной сети необходимо для реального объекта провести K испытаний для K функций управления и определить состояния объекта в дискретные моменты времени t_i, $i = \overline{1, M_k}$, $k = \overline{1, K}$. Множество значений вектора управления и вектора состояний в определенные дискретные моменты времени является обучающей выборкой

$$S = \{(u(t_i), \tilde{x}(t_i) : i = \overline{1, M_1}), \dots, (u(t_i), \tilde{x}(t_i) : i = \overline{1, M_K})\} \tag{4.26}$$

Задача обучения состоит в том, чтобы подобрать количество слоев N в искусственной нейронной сети, выбрать функции активации (4.24) на каждом слое, определить количество нейронов на внутренних слоях n_j и найти такие весовые коэффициенты матриц и векторов смещения на каждом слое, чтобы минимизировать ошибку относительно обучающей выборки

$$F = \sum_{k=1}^{K} \sum_{i=1}^{M_K} ||x(t_i) - \tilde{x}(t_i)|| \to \min . \tag{4.27}$$

Далее для полученной искусственной нейронной сети решается задача оптимального управления (4.11) – (4.25). Полученное оптимальное управление проверяется на объекте, на котором была получена обучающая выборка (4.26).

Для проверки работоспособности предложенного метода проводился вычислительный эксперимент, в котором в качестве реального объекта управления

была использована математическая модель (3.11) при $b = 2$:

$$\begin{aligned} \dot{x_1} &= \frac{(u_1 + u_2)}{2} \cos(x_3), \\ \dot{x_2} &= \frac{(u_1 + u_2)}{2} \sin(x_3), \\ \dot{x_3} &= \frac{(u_1 - u_2)}{2}, \end{aligned} \tag{4.28}$$

где управление имело ограничение $-10 \leq u_i \leq 10$, $i = 1, 2$.

Обучающая выборка имела 5090526 элементов. Для аппроксимации была выбрана двухслойная искусственная нейронная сеть с одним скрытым слоем из 50 нейронов. Для обучения использовалась библиотека TensorFlow. В качестве алгоритма обучения использовался алгоритм ADAM, заданная точность обучения 0,01, число эпох 10000. В результате обучения значение функционала (4.27) составило величину 0,003918. Обучение выполнялось на компьютере Intel Core i7 2,8 GHz.

Для полученной модели решалась задача оптимального управления с фазовыми ограничениями. Заданное начальное состояние $x^0 = [10 \quad 10 \quad 0]^T$. Терминальное состояние $x^{t_f} = [0 \quad 0 \quad 0]^T$. Функционал качества включал точность достижения терминальных условий и штрафы за нарушение фазовых ограничений

$$J_1 = t_f + p_1 ||x^{t_f} - x(t_f)|| + p_2 \sum_{i=1}^{4} \int_0^{t_f} \vartheta(\varphi_i(x(t)))dt \to \min_{u \in U}, \tag{4.29}$$

где

$$t_f = \begin{cases} t, & \text{если } t < t^+ \text{ и } ||x^{t_f} - x(t_f)|| \leq \varepsilon, \\ t^+, & \text{иначе.} \end{cases} \tag{4.30}$$

$t^+ = 2,2$, $\varepsilon = 0,001$, $p_1 = p_2 = 3$, $\vartheta(a)$ – функция Хевисайда:

$$\vartheta(a) = \begin{cases} 0, & a < 0, \\ 1, & a \geq 0. \end{cases} \tag{4.31}$$

$$\varphi_i(x(t)) = r_i - \sqrt{(x_1(t) - x_{1,i})^2 + (x_2(t) - x_{2,i})^2} \tag{4.32}$$

$i = \overline{1,4}$, $r_i = 1,5$, $r_2 = 3$, $r_3 = 3$, $r_4 = 1,5$, $x_{1,1} = 2,5$, $x_{2,1} = 2,5$, $x_{1,2} = 2$, $x_{2,2} = 8$, $x_{1,3} = 8$, $x_{2,3} = 2$, $x_{1,4} = 7,5$, $x_{2,4} = 7,5$.

Управление искалось в форме кусочно-линейной функции времени.

$$u_i = \begin{cases} 10, & \text{если } \tilde{u}_i > 10, \\ -10, & \text{если } \tilde{u}_i < -10, \\ \tilde{u}_i, & \text{иначе.} \end{cases} \tag{4.33}$$

где

$$\tilde{u}_i = (q_{k+1} - q_k)\frac{t - (j-1)\Delta t}{\Delta t} + q_k, \tag{4.34}$$

$k = j + (i-1)2$, $i = 1, 2$, $(j-1)\Delta t \le t \le j\Delta t$, $j = \overline{1, K}$, K – число временных интервалов, $\Delta t = 0,22$,

$$K = \left\lfloor \frac{t^+}{\Delta t} \right\rfloor = \left\lfloor \frac{2,2}{0,22} \right\rfloor = 10. \tag{4.35}$$

С учетом того, что значения параметров вычислялись на границе интервалов, то для каждого управления u_i, $i = 1, 2$ необходимо было найти $K + 1 = 11$ параметров. Итого решением задачи является вектор, состоящий из 22 параметров, $q = [q_1 \quad \dots \quad q_{22}]^T$. Для решения задачи применялись различные эволюционные алгоритмы [108]. Более подробное исследование методов численного решения задачи оптимального управления с фазовыми ограничениями [116] показало, что данная задача относится к классу задач глобальной оптимизации, и градиентные и квази-ньютоновские методы уступают по эффективности поиска эволюционным алгоритмам.

В частности, для решения данной задачи наилучшее решение получил следующий алгоритм, названный в [110] алгоритмом «роя - частиц» (см. также [108, 116, 148]).

С помощью вычислительной машины производим поиск решения $\tilde{u}(\cdot)$ на множестве параметров $\mathbf{q}$ по дополнительному функционалу в виде определения терминальных условий.

В реальности основной задачей регулирования является максимальное приближение выхода объекта к желаемому значению, определяемому уставкой. Таким образом, более естественно выглядит критерий типа

$$J(q) = \sum_{k=1}^{N} \left(\|x(t_k) - r(t)\| \right)_{x(0)=x^{0,k}} \to \min, \tag{4.36}$$

где $r(t)$ – уставка, траектория для численного синтеза может быть в виде сплайна для дискретного набора точек количеством N, $\mathbf{q}$ – вектор идентифицируемых параметров функции стоимости, $q = [q_1, q_2, q_3, q_4]^T$ – вычисленные для оптимального управления $\tilde{u}(\cdot)$, полученного методом PSO, используя следующие

процедуры

$$q_{k+1}^i = q_k^i + v_{k+1}^i,$$
$$v_{k+1}^i = w_k v_k^i + c_1 r_1 \left(p_k^i - q_k^i\right) + c_2 r_2 \left(p_k^g - q_k^i\right),$$

где q_k^i – величина параметра, k – текущая итерация, v_k^i – величина и направление вектора скорости каждого параметра, w_k – постоянный вес, c_1, c_2 – параметры ускорения, r_1, r_2 – случайные числа от 0 до 1, p_k^i – лучшая найденная точка варианта параметра, p_k^g – лучшая найденная точка всех вариантов параметра.

После вычисления направления вектора v, параметр перемещается в точку q_{k+1}^i. В случае необходимости обновляются значения лучших точек для каждого параметра и для всех параметров в целом. После этого цикл повторяется. Так как p_k^i, p_k^g находятся после вычисления (4.36) (движение БТС по траектории), то алгоритм требует значительных ресурсов и запускается десятки тысяч итераций на симуляторе.

Для сужения области поиска используется принцип базисного решения. Согласно этому принципу [125] мы определяем начальные значения параметров, которые называем базисным. Тогда поиск решения сосредотачиваем в окрестности базисного решения. Если базисное решение выбрано удачно, то время поиска решения можно существенно сократить. В любом случаем базисное решение можно всегда заменить хорошим решением, найденным в процессе поиска. Принцип поиска решения на основе базисного удобен при решении практических задач синтеза управления, где структуру управления может задать инженер, руководствуясь здравым смыслом и опытом.

Результат работы предложенного метода следующий:

$$\tilde{q} = [11.7384\ 10.7888\ 11.7560\ 9.8858\ 6.5680\ 11.9419\ 11.9860\ 11.0948 \ldots, 0]^T. \quad (4.37)$$

Значение функционала для данной задачи составило величину $J_1 = 1,771$. Поскольку основной составляющей функционала было время достижения терминального условия, а предельное время $t^+ = 2,2$, то полученное значение функционала указывает на то, что объект управления достигает точно терминального состояния без нарушения фазовых ограничений.

На втором этапе осуществлялась проверка полученного оптимального решения (4.37). Для этой цели кусочно-линейная функция управления подставлялась без изменений в модель (4.28). Результаты моделирования показали расхождение между траекториями не более 7%, при этом расхождение увеличивалось со временем, как и для реальной разомкнутой системы управления.

4.5 Метод генерации траекторий

В данной работе предложен метод синтеза функционала для нахождения квазиоптимальной траектории движения мобильного робота. Метод состоит из трех шагов: генерация траектории, выбор квазиоптимальной траектории и обучение нейронной сети.

Идентификация робота и генерация траектории – это два ключевых аспекта в управлении роботами, особенно в контексте мобильных роботов или манипуляторов. Они тесно связаны между собой, так как точная информация о состоянии и параметрах робота (идентификация) напрямую влияет на качество планирования и выполнения траекторий.

Генерация траектории – это процесс построения желаемой траектории движения робота (координаты, скорости, ускорения) во времени, с учётом ограничений системы: физических (ограничения по скорости, ускорению), геометрических (препятствия, пути) и динамических.

К задачам генерации траекторий относятся: построение плавной траектории от точки A до точки B, учёт ограничений приводов и динамики, обеспечение минимального времени или энергопотребления.

В данной работе рассматривается задача построения траектории от точки A до точки B.

Если модель неточна (например, неизвестны реальные массы или трение), то сгенерированная траектория может быть нефизичной, т.е робот не сможет её воспроизвести, или привести к большим ошибкам слежения при управлении, или вызвать перегрузку приводов или потерю устойчивости, и т.д. Поэтому идентификация позволяет получить точную модель, которая используется при генерации траектории.

Также оптимизация траектории зависит от динамики. При оптимальном управлении (например, при использовании методов оптимального быстродействия или минимизации энергии) учитываются уравнения движения робота. Эти уравнения содержат параметры, которые должны быть точно известны. Точная модель помогает их определить.

Управление по обратной связи требует соответствия модели и реального объекта. Если модель, используемая для генерации траектории, отличается от реального робота, контроллер может давать большие ошибки. Точная модель уменьшает эти расхождения.

В некоторых системах уточнение параметров выполняется онлайн (адаптивная система), чтобы динамически корректировать параметры модели и, соответственно, траекторию. Это важно, например, если робот меняет нагрузку (берёт

груз).

Рассмотрим, например, управление двухколёсным балансирующим роботом (сигвей). Без точной идентификации массы и момента инерции корпуса невозможно правильно рассчитать управляющие сигналы для поддержания равновесия. При генерации траектории поворота или разгона необходимо знать, как быстро колёса смогут разогнаться, как повлияет инерция конструкции – всё это требует точной модели. Если модель неточна, траектория будет «нереализуемой», и робот либо опрокинется, либо не сможет выполнить заданный маневр.

Таким образом, связь между идентификацией робота и генерацией траектории можно выразить так: идентификация обеспечивает точную модель робота, а генерация траектории строится на основе этой модели. Чем точнее модель, тем более эффективной и реализуемой будет траектория.

В диссертационной работе представлен новый метод синтеза фугкционала. Он состоит из следующих шагов: генерация траекторий и определение функции выбора.

Рассмотрим каждый шаг предложенного метода отдельно (разделы 4.5.1 и 4.5.2). А также приведем алгоритм (раздел 4.5.3) и представим результаты экспериментов на реальном роботе в реальной среде (раздел 4.5.4).

4.5.1 Генерация траекторий

Генерация траекторий – это задача построения допустимой траектории движения динамической системы от начального состояния к целевому, с учётом ограничений на динамику, управляющие воздействия и окружающую среду. Эта задача является ключевой в автономных системах, робототехнике, управлении беспилотными летательными аппаратами и других приложениях.

Напомним основную постановку задачи. Пусть

– $x(t) \in \mathbb{R}^n$ – вектор состояния системы;

– $u(t) \in \mathbb{R}^m$ – вектор управлений;

– $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ – нелинейная модель динамики

$$\dot{x}(t) = f(x(t), u(t))$$

– $\varphi(x(t), u(t)) \leq 0$ – статические ограничения;

– $\eta(x(t)) \leq 0$ – динамические ограничения;

– $J(x(\cdot), u(\cdot))$ – целевая функция.

Задача состоит в нахождении такой пары $(x(\cdot), u(\cdot))$, чтобы

$$\begin{aligned} & J(x(\cdot), u(\cdot)) \to \min_{u(\cdot)} \\ \text{при условиях: } & \dot{x}(t) = f(x(t), u(t)), \quad \forall t \in [0, T] \\ & x(0) = x_0, \quad x(T) = x_T \\ & g(x(t), u(t)) \leq 0, \quad h(x(t)) \leq 0 \end{aligned}$$

Существует множество методов генерации траекторий, приведем наиболее используемые. Ниже приведены результаты сравнения рассмотренных методов и выводы о выборе метода для дальнейших исследований.

- *Линейно-квадратичный регулятор (Linear Quadratic Regulator, LQR).* Для линейной модели $\dot{x} = Ax + Bu$ и квадратичного функционала:

  $$J = \int_0^T x^T Q x + u^T R u \, dt$$

  оптимальное управление имеет вид:

  $$u(t) = -Kx(t)$$

  где K — вычисляется через решение уравнения Риккати.

- *Модель прогностического управления (MPC)* решает задачу оптимизации на каждом шаге времени t на горизонте T:

  $$\min_{u_{t:t+T}} \sum_{k=0}^{T-1} g(x_{t+k}, u_{t+k}) + g_T(x_{t+T})$$

  при ограничениях на x и u.

- *Модель прогностического интегрального управления (MPPI).* Этот алгоритм был разработан для решения задачи поиска оптимального управления для нелинейных систем. Алгоритм MPPI берет схему методов MPC. Основное отличие от MPC заключается в том, что управление строится с использованием стохастического метода оптимизации. На каждом временном шаге MPPI создает набор управляющих последоватиельностей путем семплирования из нормального распределения с параметрами λ.

  $$u_t = \sum_{k=1}^{K} w^{(k)} u_t^{(k)}, \quad w^{(k)} \propto \exp\left(-\frac{1}{\lambda} S^{(k)}\right),$$

где $S^{(k)}$ — стоимость траектории k, учитывающая целевой функционал и штрафы за нарушение ограничений.

После чего получает набор соответствующих траекторий, используя заданную модель динамической системы. Каждая траектория оценивается с использованием некоторой функции оценки качества, и на основе полученных значений строится итоговое управляющее воздействие. Таким образом, алгоритм позволяет находить управления для нелинейных динамических систем и использовать нелинейные функции оценки качества.

- *Алгоритм быстрорастущего случайного дерева (RRT) совместно с ПИД-регулятором.* Алгоритм случайного поиска в пространстве состояний, не использующий явной модели динамики. RRT строит дерево возможных состояний, случайно выбирая точки в пространстве состояний и добавляя их к дереву, если они допустимы. Алгоритм продолжается до достижения целевой области.

На графике ниже показана примерная траектория, сгенерированная методом MPPI.

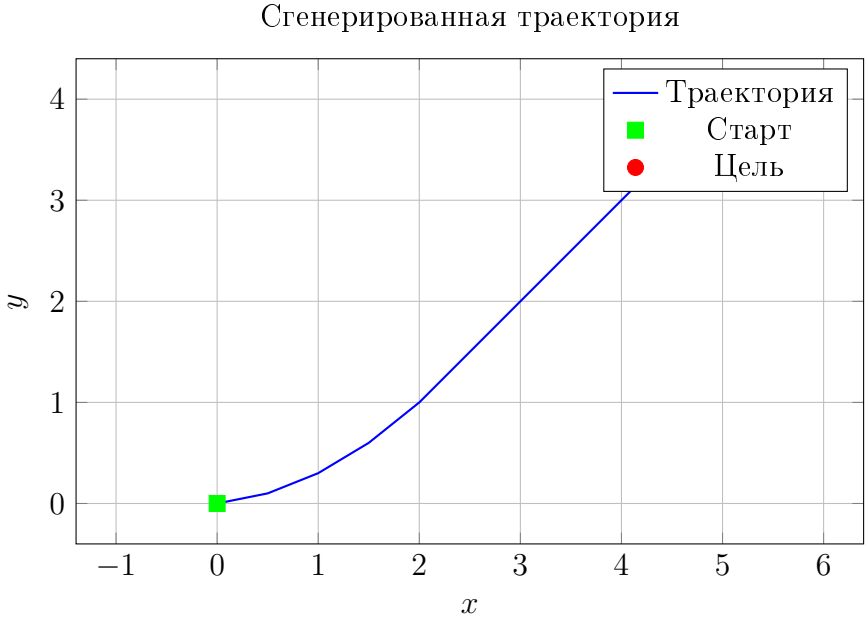

Рис. 4.13: Пример траектории, синтезированной с помощью метода MPPI

На рис. 4.14 приведены три траектории, построенные методами MPC, MPPI,

RRT. Критериями сравнения были: длина пути – чем короче, тем лучше, гладкость – оценивается через производную $\ddot{x}(t)$, вычислительная сложность – время и память, обработка ограничений – как метод справляется с препятствиями.

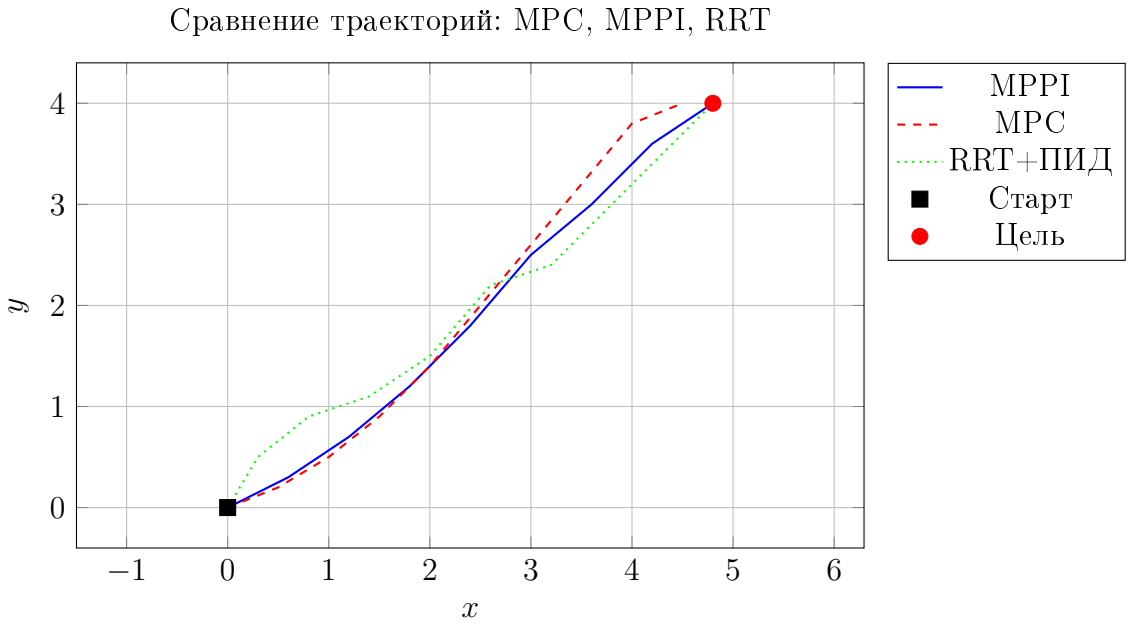

Рис. 4.14: Сравнение траекторий, построенных разными методами

На рис. 4.15 представлена ошибка $e(t)$ для каждого метода.

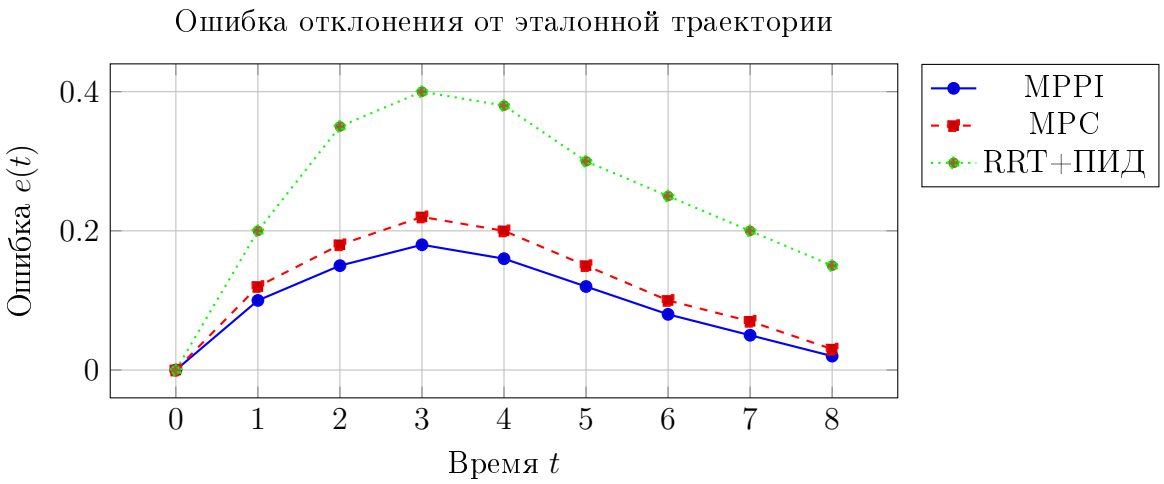

Рис. 4.15: Ошибка отклонения от эталонной траектории

В таблице приведено сравнение основных свойств методов генерации траекторий.

Таблица 4.4: Сравнение методов синтеза траекторий

Критерий	**MPC**	**MPPI**	**RRT + ПИД**
Скорость работы	Средняя	Низкая	Высокая
Точность	Высокая	Высокая	Средняя
Требования к модели	Высокие	Средние	Низкие
Обработка неопределенностей	Ограниченная	Хорошая	Отсутствует
Учет ограничений	Да	Да	Частично

Сравнение различных методов синтеза траекторий позволяет выбрать наиболее подходящий алгоритм в зависимости от задачи. По данным таблицы видно, что MPC обеспечивает гладкие траектории и точное следование ограничениям, MPPI работает эффективно в условиях неопределённости, а RRT хорошо подходит для поиска допустимого пути в сложных средах.

Решение задачи управления роботом с шасси Акерманна разными методами

В работе решена задача поиска управления для движения мобильного робота с шасси геометрии Аккермана четырьмя методами: MPC, MPPI, RRT и методом градиентного спуска со штрафами.

Для данной задачи метод MPC решает задачу оптимизации на каждом шаге t на горизонте T

$$\min_{u_{t:t+T}} \tilde{J}(x_{t:t+T}, u_{t:t+T})$$

при ограничениях:

$$x_{t+k+1} = f(x_{t+k}, u_{t+k}), \quad \forall k = 0, ..., T-1$$

$$x_t = x_{\text{current}}, \quad \varphi_i(x_{t+k}, y_{t+k}) \leq 0, \quad \eta_j(x_{t+k}, y_{t+k}, x_p, y_p) \leq 0$$

Для реализации метода градиентного спуска со штрафами было параметризовано управление как последовательность $u_{1:T}$ и минимизирован функционал

$$\min_{u_{1:T}} \tilde{J}(x_{1:T}, u_{1:T})$$

с использованием градиентного спуска. Штрафные функции обеспечили выполнение ограничений в процессе оптимизации.

В таблице 4.5 приведено сравнение тестированных методов по следующим параметрам: точность, обработка неопределенностей, учет ограничений, скорость работы, оптимальность.

Таблица 4.5: Сравнение методов управления роботом с шасси Акерманна

Критерий	**MPC**	**MPPI**	**RRT**	**Град. спуск + ПИД**
Точность	Высокая	Высокая	Средняя	Средняя
Обр. неопред.	Ограниченная	Отличная	Нет	Нет
Учет огр.	Да	Да	Частично	Да
Скорость	Средняя	Низкая	Высокая	Высокая
Оптим-ть	Локальная	Стохастическая	Не гарантирована	Локальная

На рис. 4.16 приведены графики траекторий, полученных разными методами для движения мобильного робота с геометрией шасси Аккермана.

На рис. 4.17 приведены графики изменения линейной скорости $u_1(t)$, полученных разными методами для движения мобильного робота с геометрией шасси Аккермана.

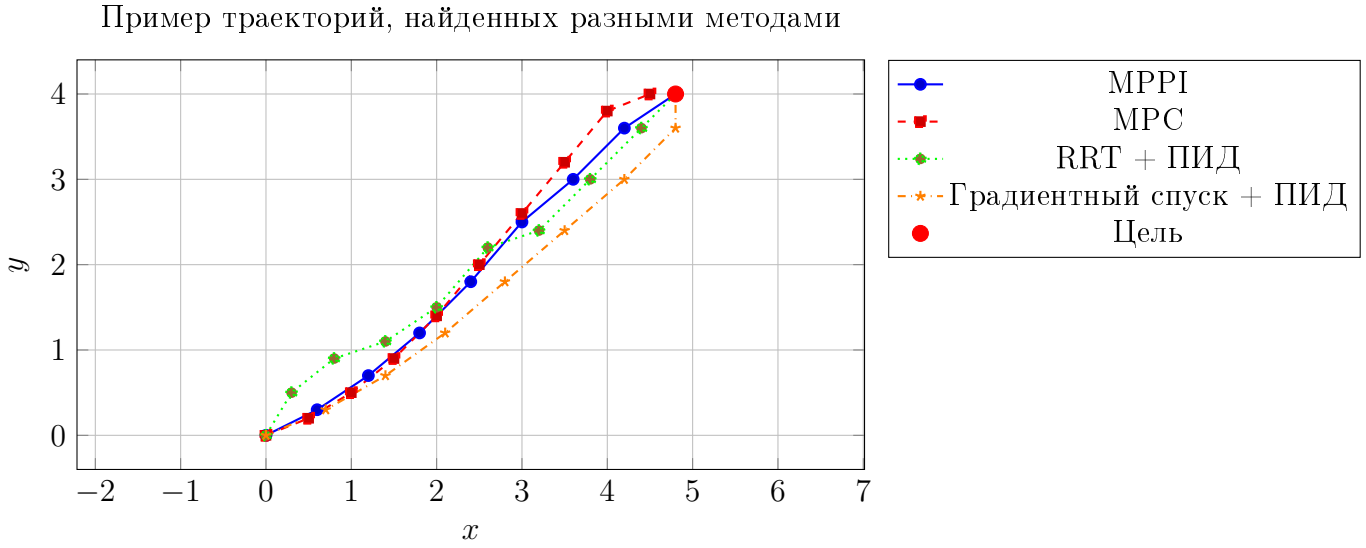

Рис. 4.16: Примеры траекторий, полученных разными методами

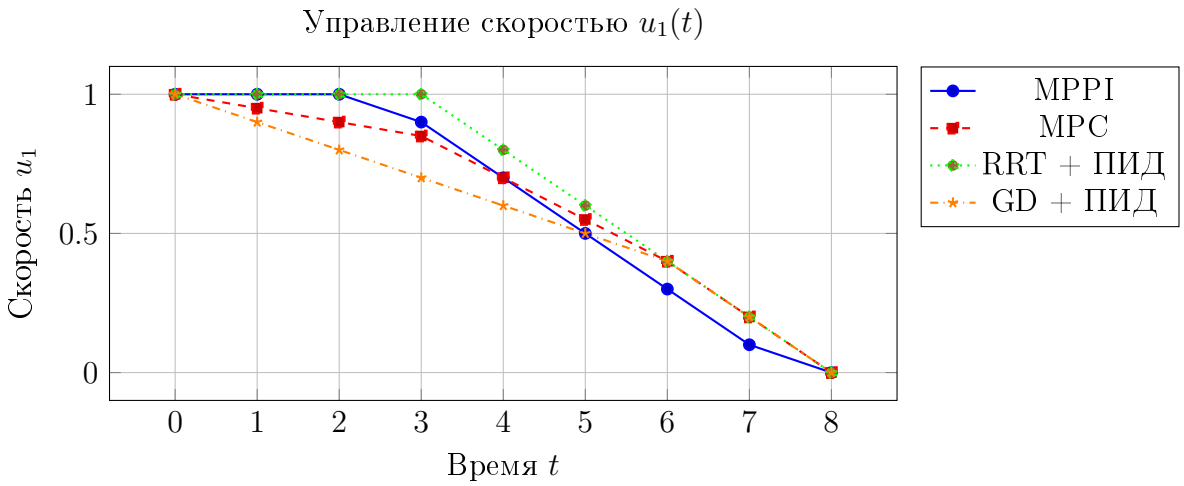

Рис. 4.17: Сравнение линейной скорости $u_1(t)$ для разных методов

На рис. 4.18 приведены графики изменения угла рулевого управления $u_2(t)$, полученных разными методами для движения мобильного робота с геометрией шасси Аккермана.

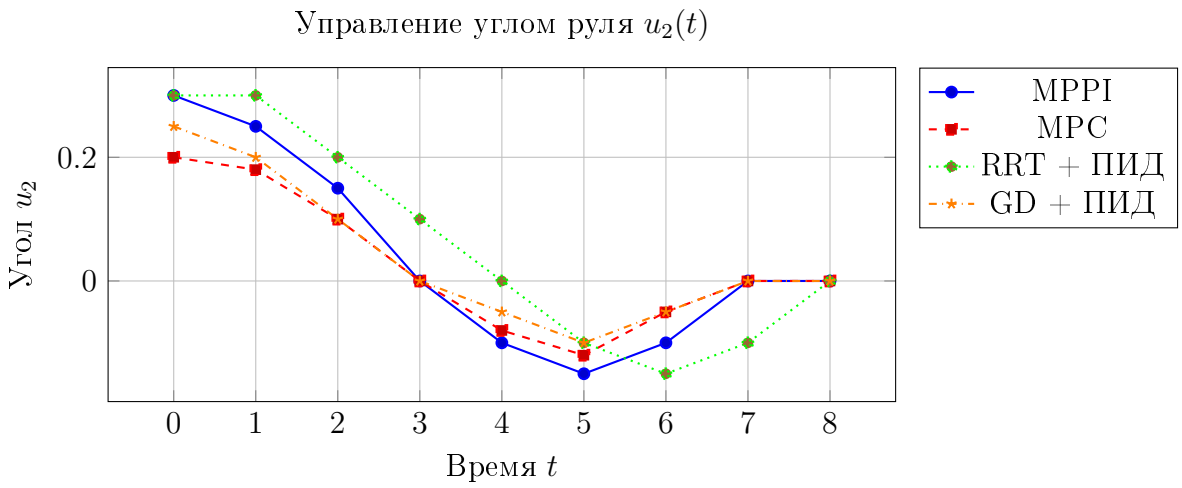

Рис. 4.18: Изменение угла рулевого управления $u_2(t)$

На рис. 4.19 приведены графики ошибки отклонения от целевой точки, полученных разными методами для движения мобильного робота с геометрией шасси Аккермана.

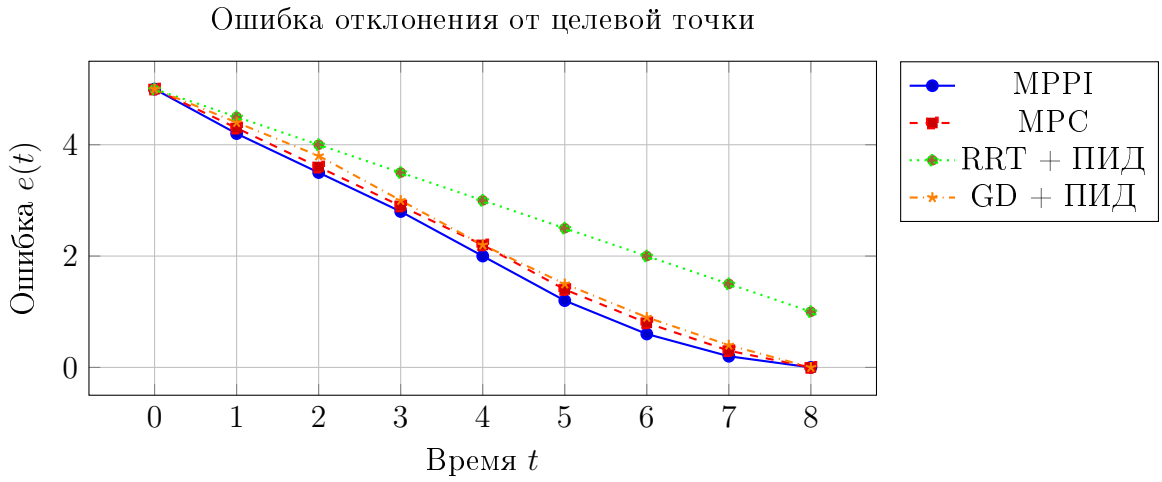

Рис. 4.19: Ошибка отклонения от целевой точки для разных методов

В таблице 4.6 приведены результаты сравнения средних значений u_1 и u_2.

Таблица 4.6: Средние значения управления по методам

Метод	**Среднее u_1**	**Среднее u_2**
MPC	0.65	0.05
MPPI	0.68	0.03
RRT + ПИД	0.60	0.08
Градиентный спуск + ПИД	0.70	0.04

В таблице 4.7 приведены результаты сравления методов по вычислительным характеристикам: время работы, число итераций, средняя ошибка.

Таблица 4.7: Сравнение методов по вычислительным характеристикам

Метод	**Время (с)**	**Число итераций**	**Средняя ошибка $\bar{e}$**
MPC	3.2	15	0.12
MPPI	7.5	50	0.10
RRT + ПИД	1.8	200	0.45
Градиентный спуск + ПИД	2.1	100	0.15

На основе большого числа экспериментов был проведен сравнительный анализ вычислительной сложности четырех методов. Полученные результаты показывают, что

1. у метода MPC наблюдалась
– высокая стоимость одной итерации (решение задачи оптимизации);
– низкое число итераций;
– подходит для систем с жёсткими требованиями к точности;
2. у метода MPPI –
– многократное моделирование ансамбля траекторий;
– высокая вычислительная нагрузка;
– отлично работает в условиях неопределённости;
3. у метода RRT –
– быстрый старт и низкая стоимость итерации;
– требует много итераций для сходимости;
– не гарантирует оптимальности;
4. у метода градиентного спуска
– низкая стоимость итерации;
– требует хорошего начального приближения;
– может застрять в локальных минимумах.

На основе проведенного в работе анализа можно сделать следующие выводы о четырех подходах к решению задачи управления мобильным роботом с шасси Акерманна:

1. метод MPC обеспечивает самую гладкую траекторию и быстрое снижение ошибки, лучший выбор при высоких требованиях к точности и допустимых временных затратах,

2. метод MPPI подходит для стохастических сред и сложных динамик в условиях неопределенности,

3. метод RRT+ПИД медленнее сходится к цели, т.к. требует дополнительных методов для пересчета траекторий и имеет большее значение ошибки, он эффективен для поиска допустимой траектории в сложных средах,

4. метод градиентного спуска+ПИД быстрый, но менее надежный в условиях неопределённости и чувствителен к начальному приближению.

Выбор метода зависит от конкретной задачи: точности, скорости, наличия неопределенностей и сложности среды. По результатам проведенных экспериментов дальнейшее исследование в работе проводится с методом MPPI.

Обзор MPPI

В данном разделе разберем более подробно модель прогностического интегрального управления (MPPI), так как именно этот метод используется в исследованиях, численных и натурных экспериментах в данной работе.

Метод MPPI – это стохастический алгоритм оптимального управления, использующий идеи интегралов по траекториям и подход модельного предиктивного управления (MPC). Он эффективен для систем с нелинейной динамикой и высокой размерностью состояния, таких как автономные роботы, летательные аппараты и бипеды. Основная идея метода заключается в том, что MPPI генерирует ансамбль возможных траекторий управления на горизонте T, вычисляет их стоимость и на основе этих стоимостей возвращает усреднённое управляющее воздействие, взвешенное экспонентой отрицательной стоимости.

Таким образом, более "дешёвые"траектории получают больший вес при формировании финального управления.

Метод MPPI [176, 186–196] основан на оптимизации функции затрат, которая определяет, где должно двигаться транспортное средство. Поэтому метод должен кодировать текущее и будущее положение дороги, препятствий, пешеходов и других транспортных средств. Функция стоимости в нашем алгоритме

получает стоимость поверхности из карты затрат. Эта стоимость самая низкая в центре трассы и выше от центра. И затем карта затрат может быть непосредственно введена в алгоритм управления прогностической моделью. Управление прогностической моделью работает путем чередования оптимизации и выполнения: сначала оптимизируется последовательность управления с разомкнутым контуром (генерируются тысячи траекторий, после чего их стоимость рассчитывается на основе функционала, конкретный тип и параметры которого зависят от выбора критерия оптимальности), затем первый контроль в этой последовательности выполняется транспортным средством, а затем поступает обратная связь о состоянии и весь процесс повторяется. Параметры оператора затрат в алгоритме MPPI настраиваются вручную, и перспективным подходом является использование методов глубокого обучения [13].

Метод реализуется согласно следующей блок-схеме

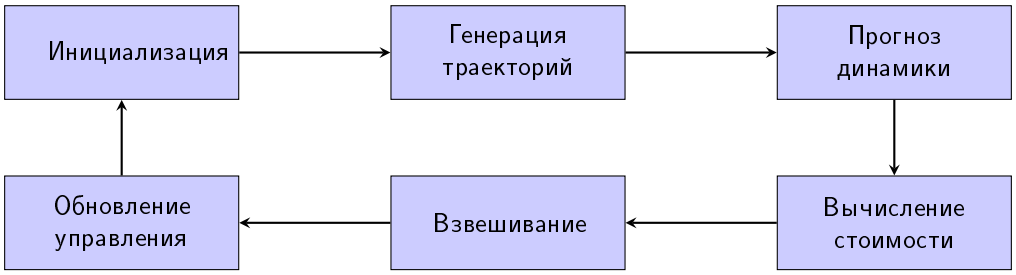

Рис. 4.20: Блок-схема работы алгоритма MPPI (Model Predictive Path Integral control)

Обозначим
– $x_t \in \mathbb{R}^n$ – состояние системы в момент времени t;
– $u_t \in \mathbb{R}^m$ – управление;
– $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ – функция перехода

$$x_{t+1} = f(x_t, u_t);$$

– $g_t(x_t, u_t)$ – мгновенная стоимость;
– $g_T(x_T)$ – терминальная стоимость;
– $\xi_{t:t+T-1} = \{\xi_t, ..., \xi_{t+T-1}\}$ – шумовые возмущения;
– K – количество траекторий в ансамбле;
– $\lambda > 0$ – параметр температуры.

Алгоритм MPPI

Шаг 1. Генерация траекторий
Для каждой из K траекторий

$$u_\tau^{(k)} = \bar{u}_\tau + \delta_\tau^{(k)}, \quad \delta_\tau^{(k)} \sim \mathcal{N}(0, \Sigma), \quad \forall \tau \in [t, t+T-1].$$

Шаг 2. Прогнозирование траекторий
Интегрируем модель динамики

$$x_{\tau+1}^{(k)} = f(x_\tau^{(k)}, u_\tau^{(k)})$$

от начального состояния x_t до горизонта T.

Шаг 3. Вычисление стоимости траекторий
Суммарная стоимость для траектории k

$$S^{(k)} = \sum_{\tau=t}^{t+T-1} g_\tau(x_\tau^{(k)}, u_\tau^{(k)}) + g_T(x_{t+T}^{(k)})$$

Шаг 4. Взвешивание траекторий
Вычисляем веса

$$w^{(k)} = \frac{\exp\left(-\frac{1}{\lambda} S^{(k)}\right)}{\sum_{j=1}^{K} \exp\left(-\frac{1}{\lambda} S^{(j)}\right)}$$

Шаг 5. Обновление управления
Обновляем последовательность управления

$$\bar{u}_\tau = \sum_{k=1}^{K} w^{(k)} u_\tau^{(k)}, \quad \forall \tau = t, ..., t+T-1.$$

Возвращаем $\bar{u}_t$ как текущее управляющее воздействие.

Ниже приведен пример псевдокода, написанного на языке Python, для реализации метода MPPI.

```
def mppi_step(state, dynamics, cost, K=100, T=20):
    costs = []
    controls = []

    for k in range(K):
        noise = np.random.normal(0, sigma, size=(T, m))
        control_seq = mean_control + noise
        state_seq = rollout(dynamics, state, control_seq)
```

```
        total_cost = compute_total_cost(cost, state_seq, control_seq)

        costs.append(total_cost)
        controls.append(control_seq)

    weights = softmax(-np.array(costs) / lambda_)
    new_mean = weighted_average(controls, weights)

    return new_mean[0]  # применяем первое действие
```

На рисунке 4.21 приведено несколько траекторий в пространстве состояний и их относительная стоимость.

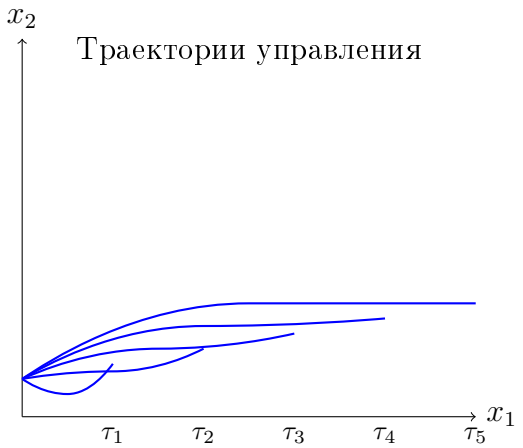

Рис. 4.21: Несколько возможных траекторий в фазовом пространстве

График 4.22 иллюстрирует, как веса распределяются между $K = 10$ траекториями при разных значениях параметра λ.

Перейдем к формализации метода для задачи управления мобильным роботом. Рассмотрим постановку задачи синтеза управления движением по траектории с дискретным временем. Задана математическая модель объекта управления

$$\dot{x}_{t+1} = f(x_t, v_t), \tag{4.38}$$

где x_t – вектор состояния объекта, v_t – фактический вектор управления, $x_t \in \mathbb{R}^n$, $v_t \sim N(u_t, \Sigma)$, $u_t \in \mathbf{U} \subseteq \mathbb{R}^m$, $\mathbf{U}$ – замкнутое ограниченное множество, $m \leq n$, $N(u_t, \Sigma)$ – Гауссов шум с математическим ожиданием u_t.

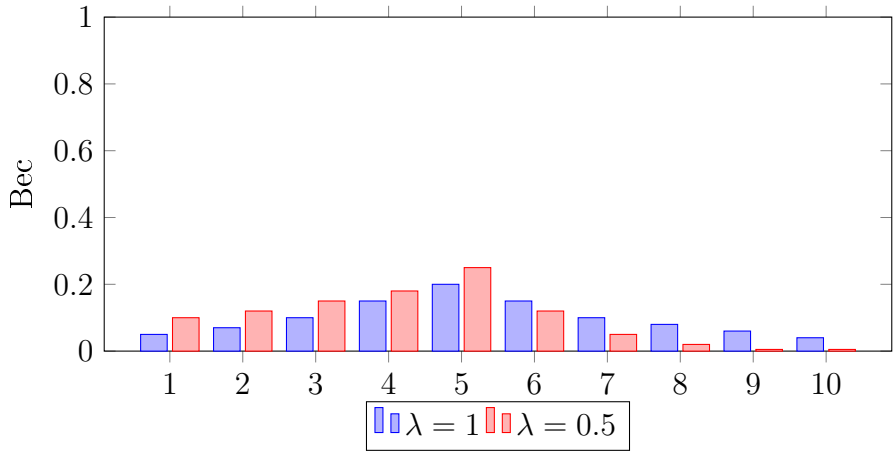

Рис. 4.22: Распределение весов по траекториям при разных значениях λ

Управление должно пройти через контроллер нижнего уровня со стохастической добавкой в виде Гауссова шума, в результате фактический вектор управления определим как $V = (v_0, v_1, \ldots, v_{T-1})$ – последовательность входов через некоторое количество шагов T. Нас интересует вычисление среднего управления $U = (u_0, u_1, \ldots, u_{T-1})$, которое минимизирует функцию стоимости

$$J(U) = E\left[q_1(x_T) + \sum_{t=0}^{T-1}\left[q_2(x_t) + \lambda u_t^T \Sigma^{-1} u_t\right]\right], \tag{4.39}$$

где $q_1(\cdot)$ – терминальная стоимость состояния, а $q_2(\cdot)$ – мгновенная стоимость состояния, $S(V^m) = q_1(x_T^m) + \sum_{t=0}^{T-1} q_2(x_m^t)$ – часть стоимости, зависящая от состояния.

Алгоритм MPPI начинается с инициализации вектора состояния x_0. Вычисление управления начинается с учета управляющей последовательности из предыдущей итерации U_{k-1}, симуляции (тысячи) траектории параллельно, каждая с различной последовательностью управления V^m, где m – реализация случайной траектории. Затраты на управление собираются для каждого развертывания и сопоставлены с весами траектории:

$$\omega(V^m) = \exp\left(-\frac{1}{\lambda}\left(S(V^m) - \sum_{t=0}^{T-1} u_{t,k-1}^T \Sigma^{-1} v_t^m - \rho\right)\right), \tag{4.40}$$

где ρ устанавливается в минимальное значение затрат среди всех выбранных траекторий, предназначен для арифметического недопущения. Квазиоптималь-

ное управление на каждом временном шаге вычисляется как:

$$u_{t,k} = u_{t,k-1} + \frac{1}{\sum_{m=1}^{M} \omega(V^m)} \sum_{m=1}^{M} \left[\omega(V^m)\xi^m\right], \tag{4.41}$$

где $\xi^m \sim N(0, \Sigma)$ – Гауссов шум с нулевым средним, M – число реализаций траекторий, q – вектор идентифицируемых параметров функции стоимости, $q = [q_1, q_2, q_3, q_4]^T$.

Методы генерации траекторий с помощью искусственных нейронных сетей

В работе предложен новый способ генерации траекторий. Этот метод использует нейронные сети и архитектуры глубокого обучения.

Известны различные подходы к нейросетевой генерации траекторий. Среди наиболее популярных

- Рекуррентные нейронные сети (RNN, LSTM, GRU) Траектории, как последовательности координат, удобно моделировать с помощью RNN и их модификаций (LSTM, GRU), которые хорошо захватывают временные зависимости и позволяют предсказывать будущие точки на основе истории движения [256]. Подходы с двунаправленными RNN (Bidirectional RNN) и механизмами внимания (attention) улучшают качество за счёт анализа как прошлого, так и будущего контекста [256].

- Генеративные модели (GAN, CNN) Для синтетической генерации траекторий применяются генеративно-состязательные сети (GAN). Новые исследования предлагают преобразовывать траектории в формат, пригодный для обработки сверточными нейросетями (CNN), что позволяет использовать достижения компьютерного зрения для генерации реалистичных пространственных паттернов траекторий [257].

- Мультимодальные и многоагентные методы Современные архитектуры, такие как ADAPT, позволяют одновременно предсказывать траектории для всех агентов в сцене, учитывая взаимодействия между ними и используя динамическое обучение весов. Такой подход обеспечивает высокую точность и эффективность предсказаний в задачах автономного транспорта [258, 259].

- Интеграция с планировщиками маршрутов В практических системах автономного вождения нейросети используют не только сенсорные данные, но и заранее сгенерированные маршруты (траектории), построенные с помощью глобальных планировщиков, что позволяет учитывать инфраструктуру и ограничения дорожной сети [260]. В данном работе рассматриваются именно такие способы генерации траекторий.

Предлагаемый метод заключается в следующем: для решения задачи синтеза системы управления движением объекта по пространственной траектории используем генератор траекторий.

Для генератора траекторий зададим вектор, состоящий из начального и конечного состояния, начальной и конечной кривизны, а также скорости: $x_l = [x, y, \psi, k, \nu]^T$, $x_F = [x, y, \psi, k, \nu]^T$, что приводит к пятимерной таблице поиска векторов параметров. Разрешение и размерность сохраненной таблицы является функцией хранения и вычислительной мощности робота и, следовательно, зависит от платформы.

Глобальный планировщик создает предпочтительный путь для робота или поле затрат, или функция навигации , кодирование оптимального расстояния до цели от каждого состояния. Как правило, управление с обратной связью применяется к компенсированию расхождений робота с моделью движения, но в целом оно не может компенсировать неосуществимые траектории. Применяя метод, который интегрирует более сложные модели транспортного средства на стадии траекторного планирования, мы можем уменьшить количество ошибок, которые контроллер обратной связи должен компенсировать, имея более точный критерий стоимости кандидата для траектории на время его действия.

Пример генерации соседних траекторий приведен на рис. 4.23.

Поэтому важно, создать функцию выбора, которая правильно считает некоторую смесь риска столкновения, потребление энергии, задержку наклона (время, проведенное на склонах) глобального планировщика, гладкости траектории, и другие «издержки», которые могут быть связаны с траекторией.

4.5.2 Определение функции выбора

В данном разделе описан новый метод нахождения функции выбора. Существуют разные способы определения этой функции:

- ручной;
- аналитический;

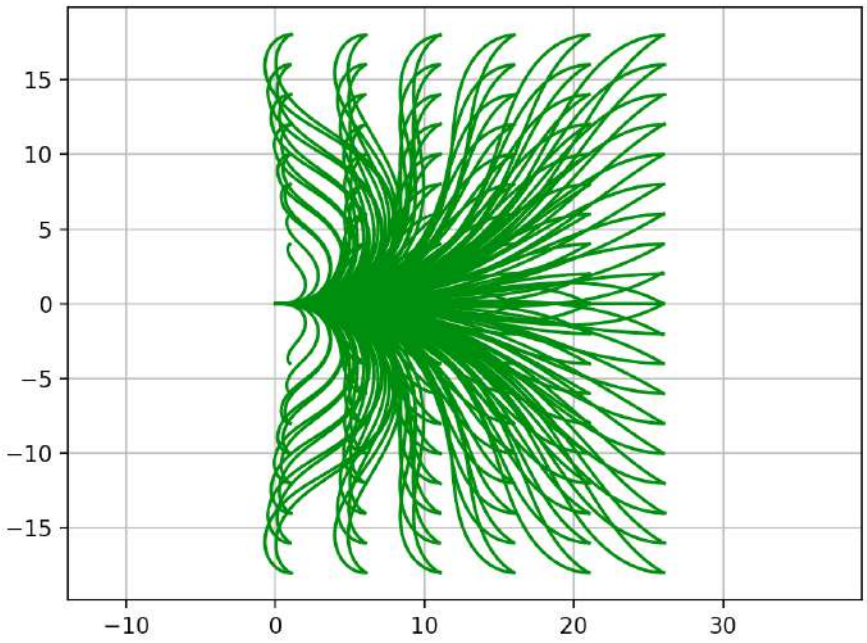

Рис. 4.23: Пример генерации соседних траекторий.

- через обучение с подкреплением;
- с помощью сетевого оперетора.

В проведенных в диссертацонной работе исследованиях применялись три из пересчисленных варианта: ручной, аналитический и с помощью сетевого оператора. При ручном выборе управлние осуществляется в ручном режиме для получения набора данных.

Остановимся подробнее на аналитическом методе.

На первом этапе с помощью методов численной оптимизации строится первое приближение оптимальной траектории $\Omega(x)$ с учетом заданных конечного и начального состояний и статических ограничений. Этот путь называется функцией навигации $\eta(x)$. Для модели объекта на борту решается несколько краевых задач по заданному функционалу и выбираются различные конечные состояния, не совпадающие с исходной задачей из условий вычислительной возможности автономного решения задачи. Желательно установить начальные предположения параметров управления близко к фактическому решению, тем меньше итераций алгоритма необходимы для страхования от попадания в неверный минимум. Исторически аппроксимации решений были найдены ручной настройкой полиномиальных функций данных из нескольких измерений. Для решения этой проблемы можно использовать метод динамического программирования [261]. Все сгенерированные траектории сохраняются для дальнейшего анализа и выбора. В реальных условиях подбирать траектории по исходному функционалу нецелесообразно, поскольку конечные условия могут привести к тупикам.

На втором этапе выполняется предсказание движения (численное интегрирование, чтобы предсказать движение).

Для этого мы считаем функционал качества равным максимальному значению функции суммы всех штрафов, полученных при достижении конечного состояния,

$$F(Z_k) \to \max, \quad Z_k = \sum_{i=1}^{K} R_i(x, u, y),$$

где y - вектор параметров, не входящих в вектор состояния, но влияющих на выбор участка траектории, например, отклонение от навигационной функции, параметры, определяющие изменение высоты траектории, допустимые скорости, расстояния до препятствий и т.д. В исследованиях использовались методы, описанные в [262], [263], в них осуществлялся выбор траекторий из множества полученных траекторий, в данном случае строится одна общая траектория.

Новизна такого метода заключается в том, что определяется адаптивный функционал, т.е. при изменении условий внешней среды он меняет параметры выбора.

Теперь рассмотрим нахождение функции выбора с помощью сетевого оператора [96, 125, 227–229]. В этом методе определялась структура функционала.

Суть метода заключается в том, что он кодирует математическое выражение в виде ориентированного графа, граф описывает состав элементарных функций с одним или двумя аргументами. В памяти компьютера граф хранится в виде целочисленной матрицы, элементами которой являются номера функций из базового набора. В процессе поиска сначала кодируется основное математическое выражение, затем поиск выполняется с использованием генетического алгоритма по набору вариантов основного решения. Критерием поиска является оценка найденного оптимального управления по критерию начальной оптимизации (3). В результате выбор оптимальной траектории осуществляется совместно с синтезом критерия выбора для конкретных траекторий.

Пример кодирования методом сетевого оператора

Например, для описания математического выражения

$$y = a \sin\left(e^{-(x+b)} \sin(ax)\right) \tag{4.42}$$

в форме сетевого оператора мы используем следующие основные наборы функций

$$\begin{aligned} F_0 &= (x_1, a, b)\,, \\ F_1 &= (f_{1,1}(z) = z, f_{1,2}(z) = -z, \\ &\qquad f_{1,3}(z) = e^z, f_{1,4}(z) = \sin z), \\ F_2 &= (f_{2,1}(z_1, z_2) = z_1 + z_2, f_{2,2}(z_1, z_2) = z_1 z_2)\,, \end{aligned}$$

где индекс i в имени набора F_i показывает количество аргументов функции.

На рис. 4.24 показан граф оператора сети для математического выражения (4.42).

В компьютере граф сетевого оператора представлен в виде целочисленной

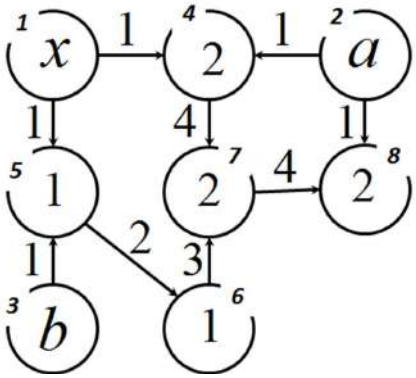

Рис. 4.24: Граф оператора сети для математического выражения (4.42)

матрицы

$$\Psi = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

С помощью вычислительного метода находим функцию выбора, обеспечивающую минимум отклонения от глобальной траектории, по которой должен двигаться мобильный робот, избегая столкновения и другие ограничения вектора состояния и управления [262].

Поскольку функция навигации обычно используется в свертке с траекторией, то лучшая траектория вычисляется функцией выбора для каждого действительного управления, который соединяет пару состояний, избегая препятствий:

$$C_N = [c_0, c_1, \ldots, c_n]^T,$$

Большинство современных методов предполагает ручную настройку операторов выбора, а перспективным подходом является применение методов машинного обучения, чтобы обучить правильному отображению действий и окружающей среды на функцию выбора.

Предложенный метод заключается в обучении нейронной сети генерировать элементы управления для различных предварительно сгенерированных траек-

торий, например, для объезда препятствий, для движения по заданной траектории и т.д. Для этого входные данные нейронной сети от камер и лазерных дальномеров объединяются с выходными данными функции выбора, рис. 1.

Контролируемое обучение нейронной сети основано на данных с камер и лазерных дальномеров и управлении обычным контроллером по заданной траектории. Переключение между заданными траекториями осуществляется с помощью функции выбора на основе поступающих данных от датчиков восприятия.

Наилучшая траектория вычисляется с помощью функции выбора $C_N(\mathbf{q})$ для каждого допустимого элемента управления за период $[0, t_f]$, которая соединяет пару состояний, начальное $x(0)$ и конечное x_f, избегая препятствий:

$$C_N(\mathbf{q}) = [\mathbf{c_0}(\mathbf{q}), \mathbf{c_1}(\mathbf{q}), \ldots, \mathbf{c_n}(\mathbf{q})]^{\mathbf{T}},$$

где $\mathbf{q}$ - вектор настраиваемых параметров.

Последние достижения в области глубокого обучения позволяют создавать современные модели нейронных сетей для решения сложных задач. Задача управления нейронной сетью мобильного робота требует многомодальных входных данных. На входные данные можно получать данные из разных источников, обрабатывая каждый тип данных с использованием различных типов нейронных слоев.

Обучение осуществляется на основе данных, полученных с помощью обычного контроллера, который отслеживает перемещения примитивов (фиксированных маневров), называемых порождающими решетками, которые приводят к регулярным графам. Такие примитивы соответствуют решению двухточечной краевой задачи. Функция выбора должна выбирать те маневры, которые соответствуют целевой глобальной оптимальной траектории по отношению к заданному функционалу качества.

Большинство современных методов предполагают ручную настройку операторов выбора, но многообещающим подходом является использование методов машинного обучения для обучения правильному сопоставлению действий и окружения с функцией выбора. В работе рассмотрен подход, в котором параметры функции выбора определяются методом оптимизации роя частиц (Particle swarm optimization, PSO) [110, 112]. Используя вычислительный метод, мы находим функцию выбора, обеспечивающую минимальное отклонение от заданной траектории, по которой должен двигаться мобильный робот, избегая столкновения и других ограничений на вектор состояния и управления.

Для проведения экспериментов используется физический движок Gazebo, с моделью, основанной на технических характеристиках реального робота, в среде, основанной на технических характеристиках реального трека Робототехниче-

ского центра ФИЦ ИУ РАН [264]. Нейронная сеть переобучается на физическом движке модели робота Gazebo. Мы получаем синтез функций управления с помощью интегрирования. Gazebo используется для генерации функции выбора.

4.5.3 Алгоритм синтеза функционала

Для получения управления на основе функции выбора и сетевого оператора предлагается использовать следующий алгоритм.

Алгоритм.

Шаг 1. Определение модели беспилотного транспортного средства.
Провести идентификацию модели с использованием регрессионных модельных структур.

Шаг 2. Выбор траектории.
Сненерировать траекторию методом MPPI. Выбрать интегральное управление по лучшей траектории.

Шаг 3. Определение функции выбора.
Считаем функционал качества равным максимальному значению функции суммы всех штрафов, полученных при достижении конечного состояния,

$$F(Z_k) \to \max, \quad Z_k = \sum_{i=1}^{K} R_i(x, u, y),$$

где y - вектор параметров, не входящих в вектор состояния, но влияющих на выбор участка траектории, например, отклонение от навигационной функции, параметры, определяющие изменение высоты траектории, допустимые скорости, расстояния до препятствий и т.д.

Определить функцию выбора.

В разделе 4.5.4 приведен пример использования алгоритма на реальном роботе.

4.5.4 Результаты применения метода генерации траекторий

В этом разделе приведены результаты применения предложенного метода генерации к мобильному роботу (рис. 4.1) с шасси геометрии Аккермана (4.1) в Робототехническом центре ФИЦ ИУ РАН (рис. 4.2).

Беспилотное транспортное средство построено с использованием автомобильного шасси масштаба 1/10, весит 1,5 кг и способен развивать скорость до 10 м/с, также в модели робота используются следующие параметры:

Параметр	значение
радиус действия робота [m]	1
максимальная скорость [m/s]	50.0 / 3.6
максимальное ускорение [m/ss]	2.0
максимальная кривизна [1/m]	1.0
максимальная ширина дороги [m]	7.0

Вычисления выполнялись с помощью комплекта NVIDIA Jetson TX2. Измерение локальных координат движения БТС проводилось в Робототехническом центре ФИЦ ИУ РАН [264].

Решение методом синтеза функционала Чтобы имитируемая динамика не отличалась от истинной динамики транспортного средства, были смоделированы различные состояния, похожие на реальные, и создана репрезентативная выборка данных для обучения нейронной сети для преобразования сцены в управление в реальном времени, рис. 4.25 слева. Чтобы изучить функцию регрессии по пикселям, требуется порядка сотен тысяч кадров обучающих данных, рис. 4.25 справа. Кадры с видеокамеры вместе с управлением были сохранены в тот момент, когда контроллер MPC генерировал управление для конечной точки. Архитектура CNN ограничена работой в режиме реального времени на NVIDIA Jetson TX2, аналогичная сеть установлена и на симуляторе.

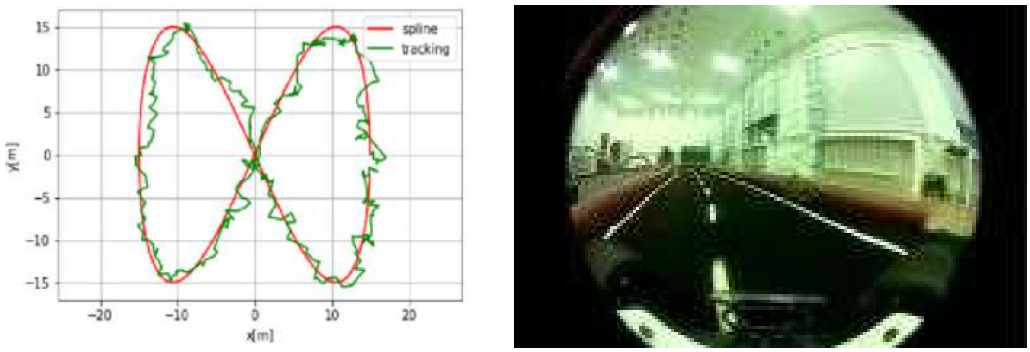

Рис. 4.25: Прохождение УФ-излучения по траектории Лиссажу с добавлением шума с нормальным распределением и стандартным отклонением 0,7 к вектору состояния.

Для того чтобы моделируемая динамика не отличалась от истинной динамики транспортного средства необходимо точно вычислять свое состояние относительно затрат на пути к цели в прогнозируемом пространстве. На прогнозируемом пространстве используется монокулярное зрение и глубокая сверточная нейронную сеть для преобразования сцены в карту затрат в реальном времени. Рассмотрено создание функции для выбора заранее определенных путей, и сеть обучается с учетом выходных данных этой функции в сети объединения, см. рис. 4.26.

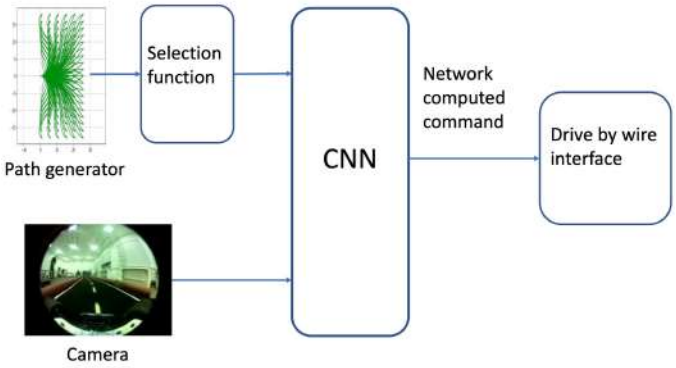

Рис. 4.26: Обученная сеть используется для генерации управляющих команд

Чтобы изучить функцию регрессии по пикселям, способную производить затраты на прохождение для каждого пикселя, данные обучения нужны порядка

сотен тысяч кадров. Маркировка всех этих данных вручную трудоемкий, медленный и склонный к ошибкам процесс. Необходимы датчики и камеры, которые могут связывать каждое изображение с полной оценкой состояния, включая ориентацию и положение в сочетании с обзорной картой трека, зарегистрированного по координатам GPS, они могут быть используется для создания сотен тысяч маркированных изображений без какой-либо ручной маркировки отдельных картинок.

Наиболее простой подход в создании обучающей маркированной выборки состоит в преобразовании реальной сцены с одной камеры с помощью библиотеки OpenCV в плотную карту затрат. Преобразование состоит из нескольких этапов. На первом этапе находятся ключевые точки, выделяются ограничения трека с помощью установленной на роботе камеры. На втором этапе необходимо произвести трансформацию изображения с помощью матрицы гомографии по четырем или более точкам (рис. 4.27). На третьем этапе происходит кодировка изображения. Ограничения кодируются числом 255, а предпочтительный путь нулем. Пиксели от 0 до 255 изменяются по градиенту. Сгенерированная методом MPPI траектория будет стоить меньше, если будет условно проходить по пикселям с меньшим числом. Таким образом, робот движется по более светлому участку пути (рис. 4.28).

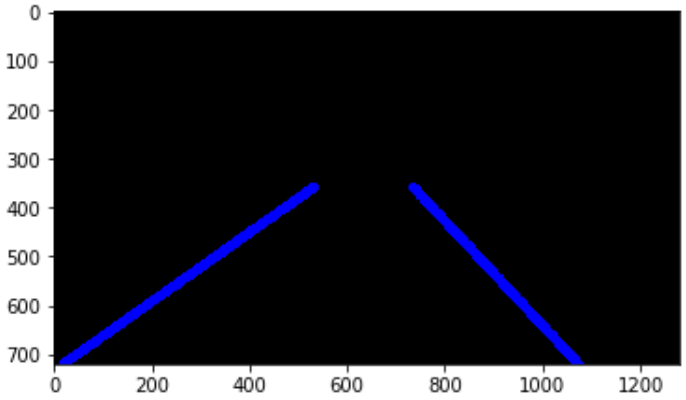

Рис. 4.27: Трансформацию изображения с помощью матрицы гомографии по четырем или более точкам

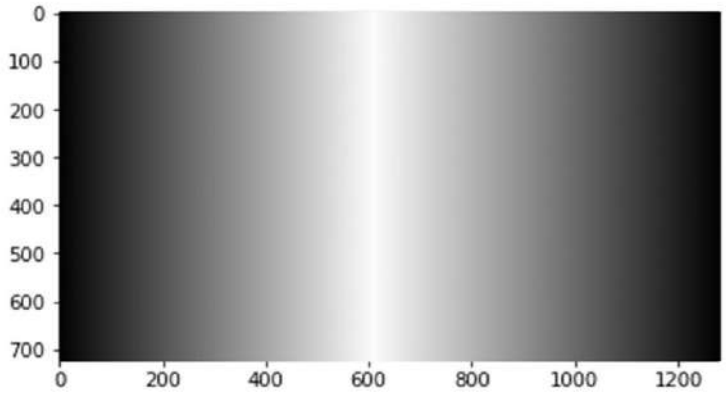

Рис. 4.28: Траектория движения робота

Кроме того, подъемы и спуски могут кодироваться определенным числом в зависимости от крутизны.

С помощью камеры БТС, который использовался в экспериментах, были сохранены 15000 последовательных изображений размером 160x120 пикселей. С помощью программы преобразования были получены соответствующие изображениям карты затрат непосредственно перед транспортным средством (в координатах транспортного средства). Это включает в себя значительные изменения в условиях освещения, ограничений и позы БТС на треке. После чего была обучена сверточная нейронная сеть. Архитектура CNN ограничена для работы в режиме реального времени на Nvidia Jetson TX2.

Сеть принимает входные изображения 160x120 и передает их через несколько фильтров свертки и 2 объединенных слоя, за которыми следует набор из 4 расширенных фильтров сверток. Наши сети используют сверточные блоки 3x3 с BatchNorm и ReLU. По сравнению с прямым методом получения карты затрат с помощью OpenCV, обученная таким образом сеть может терпеть удаление небольших областей трека из-за засвечивания и все еще производить карты стоимости, пригодные для использования.

В отличие от контролируемого обучения на трассе в ручном режиме (рис. 4.29 слева), где график управления нейроконтроллером в точности повторяет управление человека-оператора, нейроконтроллер, основанный на имитации контроллера MPC (рис. 4.29 справа), имеет видимое более плавное управление,

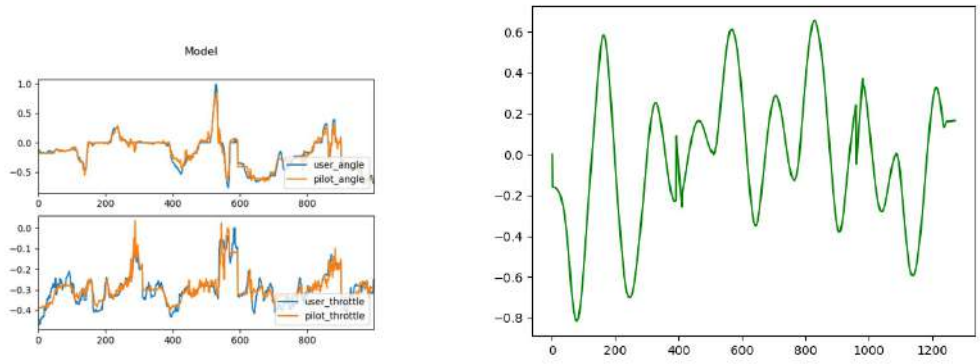

Рис. 4.29: (a) Контролируемое ручное обучение по маршруту; (b) квазиоптимальное управление $u2$.

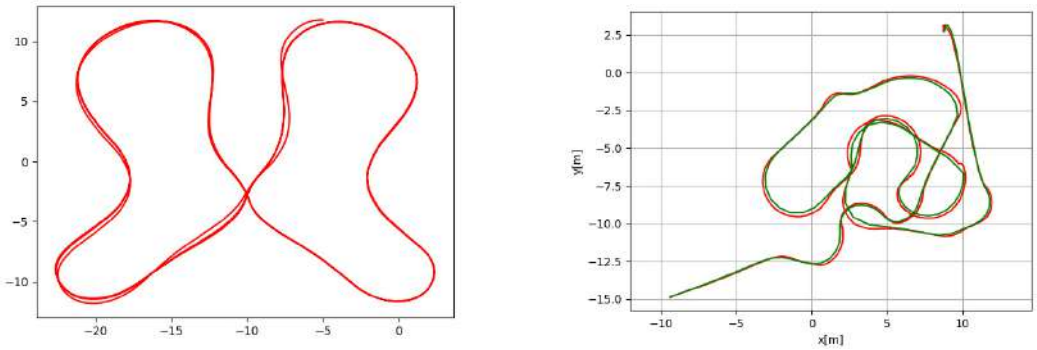

Рис. 4.30: (a) Функция, аналогичная траектории лиссажу длиной 200 м; (b) точки на этой траектории были выбраны через каждые 0,5 метра

например, через канал рулевого управления $u2$.

В качестве критерия оптимальности мы берем среднеквадратичную ошибку состояния желаемого значения $r(t)$ (9). При проведении экспериментов в качестве траектории была выбрана функция, аналогичная траектории Лиссажу (рис. 4.30) длиной 200 м и другим траекториям, а в качестве конечных точек были выбраны точки на этой траектории через каждые 0,5 метра.

Среднеквадратичные значения среднеквадратичных ошибок приведены в таблице 4.8, где 1 означает MCO, 2 - ручное управление NNC и 3 - управление NNC MPC.

Таблица 4.8: The results of numerical experiment

1	2	3
Coordinate x	0.0924	0.0016
Coordinate y	0.1029	0.0058
Orientation angle θ	0.0920	0.0368

Пример нейросетевого управления После проведения натурных экспериментов было установлено, что прямоу управление через нейросеть получается неустойчивым, поэтому в дальнейших экспериментах и исследованиях использовался только метод синтеза функционала.

В качестве встраиваемой вычислительной платформы был использован NVIDIA Jetson TX2 с 256 ядрами CUDA и четырёх ядерным процес-сором ARM Cortex-A57 это позволило распараллелить компьютерную сеть.

Функцию выбора находим методом сетевого оператора. Для поиска решения, минимизируем функционал, описывающий точность движения по траектории, используем вариационный генетический алгоритм со следующими параметрами: раз-мер начальной популяции - 50; число поколений - 10; число скрещиваемых пар в одном поколении - 30; число вариаций в одном решении - 10; число поколений между сменой базисного решения – 2; число элитарных реше-ний – 16; вероятность мутации - 7.0 ; параметр для скрещивания - 4.0.

Для решения задачи синтеза системы управления перемещением объекта по пространственной траектории $\Omega(x, y)$ используем 2 функции управления и функцию выбора, полученные методом сетевого оператора. Схема метода приведена на рис. 4.31.

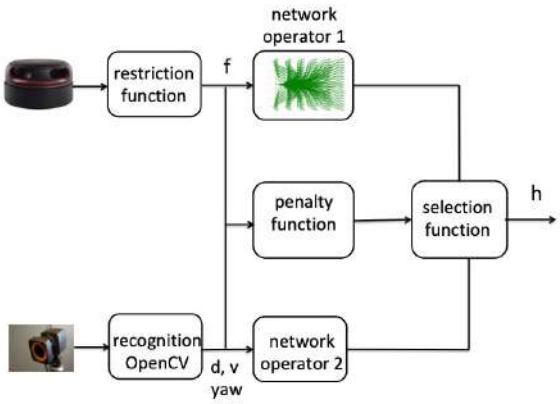

Рис. 4.31: Схема метода УФ-контроля с двумя функциями управления в виде сетевых операторов и функцией выбора между ними

Оператор сети 1 (SO1) получает f-функцию статических или динамических ограничений от rmlidar a2, установленного на мобильном роботе. В функциональном блоке ограничений облако точек преобразуется в функции прямых линий, треугольников и т.д. Синтез SO1 основан на параметрах функции f и параметрах навигационной функции (визуально это пунктирная линия на рис. 4.47). На выходе SO1 генерируется маневр обхода препятствий с учетом оставшихся ресурсов и минимизацией базового функционала.

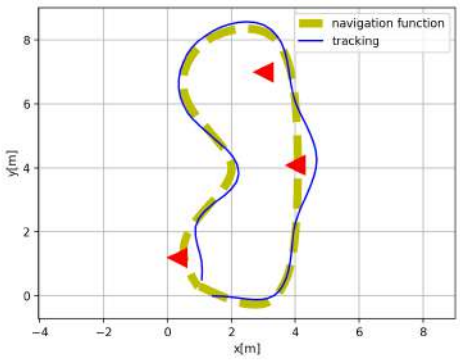

Рис. 4.32: Функция навигации и отслеживания

Оператор сети 2 (SO2) на вход получает d, который является расстоянием до ближайшей точки на желтой линии, рис. 4.47 (навигационная линия), v, которая является скоростью UV, скорость определяется с помощью оптического

потока, yaw - ориентация угол по отношению к навигационной линии в ближайшей точке. Расстояние до ближайшей точки и угол ориентации относительно навигационной линии рассчитываются в блоке распознавания OpenCV, где желтая линия распознается на видеокадрах с помощью библиотеки OpenCV, и вычисляются параметры d, yaw. Синтез SO2 основан на параметрах $d(t)$, $v(t)$, $yaw(t)$ и $d(t-1)$, $yaw(t-1)$. Выходные данные SO2 генерируют управление отслеживанием траектории.

Блок выбора в виде сетевого оператора 3, основанный на точных параметрах, переключает управление сетевыми операторами 1,2 на управление UV, гарантируя, что цель следования пути достигается оптимальным образом в отношении заданного функционала качества, принимая во внимание ограничения в реальном времени, параметры которые заранее не известны.

Для конструирования нейросети используется фреймворк глубокого обучения Keras, написанный на Python, и библиотека Tensor Flow в качестве внутреннего механизма. В качестве модели была выбрана сверточная сеть с пятью слоями рис. 4.33–4.37, за которыми следуют два плотных слоя перед выводом, за векторными данными следуют 3 плотных (полносвязанных) слоя, затем они объединяются с выходом X до 2 плотных контрольных слоев. На рис. 4.39 приведены графики ошибки обучения при обучении с помощью контроллера, копирующего ПИД-регулятор.

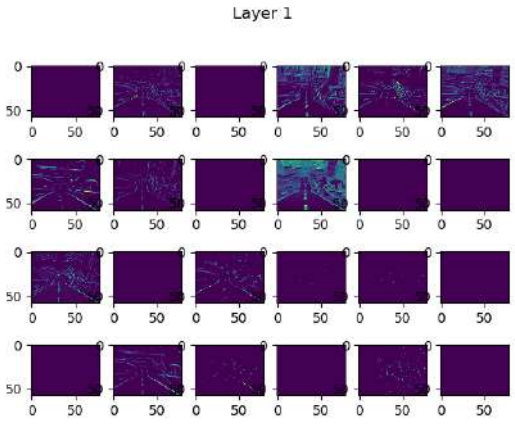

Рис. 4.33: Сверточная сеть. Первый слой

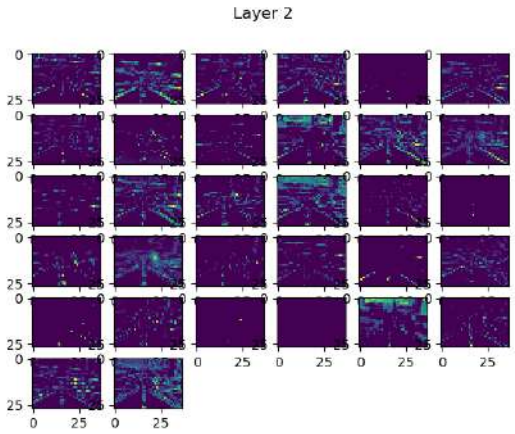

Рис. 4.34: Сверточная сеть. Второй слой

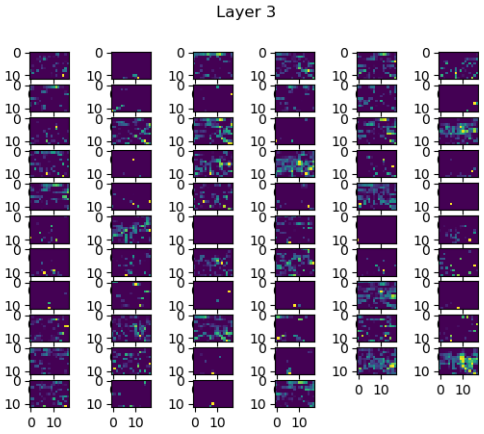

Рис. 4.35: Сверточная сеть. Третий слой

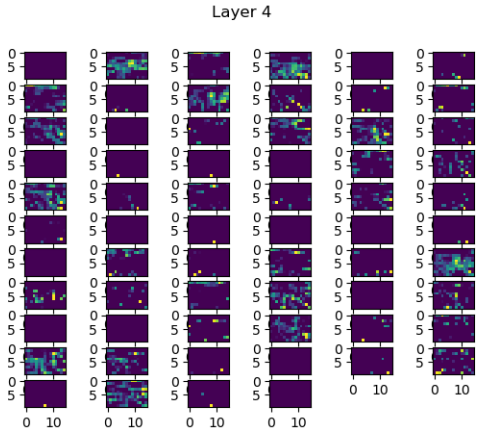

Рис. 4.36: Сверточная сеть. Четвертый слой

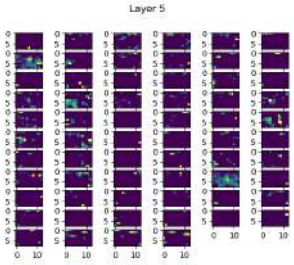

Рис. 4.37: Сверточная сеть. Пятый слой

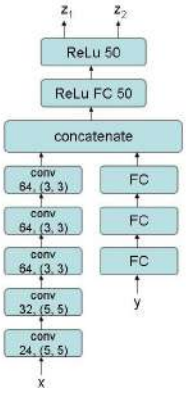

Рис. 4.38: Блок-схема метода нейросетевого управления

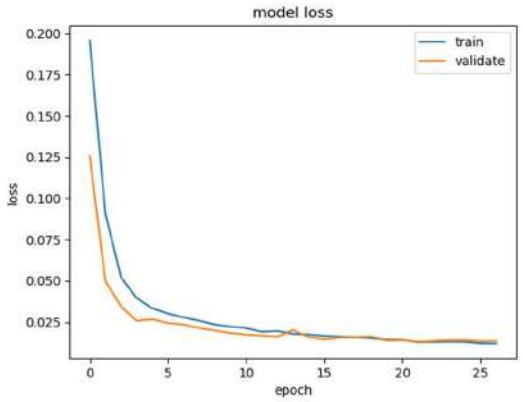

Рис. 4.39: Ошибка обучения при обучении с помощью кон-троллера, копирующего ПИД-регулятор

Вычислительные эксперименты проводились на симуляторе, написанном на языке Python [265].

В результате синтеза было получено управление - поворот колес робота. Форма сетевого оператора, в виде списка [14], для полученного контроля имеет следующий вид

$$R = ((0, 19), (2, 0), (5, 4)), ((5, 10), (3, 5), (18, 1)).$$

Мы декодируем получившийся список в символьное выражение традиционной математической формы. В результате решение, полученное после декодирования и упрощения, получается в виде

$$u_1 - -3.5 \operatorname{arctg} \theta, \quad u_2 = \text{const},$$

где θ – угол между направлением движения робота и направлением на цель.

На рис. 4.40 показаны траектории решения краевых задач вместе со схематическим изображением реальных условий.

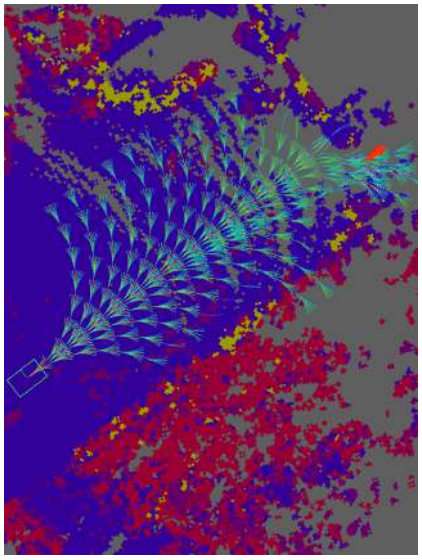

Рис. 4.40: Траектории решения краевых задач

На рис. 4.41 показаны траектории решения краевых задач вместе со схематическим изображением реальных условий вместе с оптимальной траекторией.

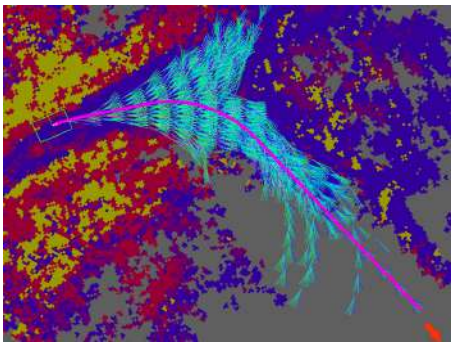

Рис. 4.41: Траектории решения краевых задач с выбранной оптимальной траекторией

4.6 Метод ASLAM-MPPI

4.6.1 О методах ASLAM-MPPI

Расчет оптимальной траектории в сложных средах со статическими и динамическими ограничениями требует оценки всего пространства возможных состояний и поиска наилучшего решения.

В работе предложено решение этой проблемы с использованием алгоритма метода активной одновременной локализации и картографирования на основе модели прогнозирующего интегрального пути для мобильных роботов ASLAM-MPPI (ASLAM, от англ. method of Active Simultaneous Localization And Mapping; MPPI, от англ. Model Predictive Path Integral method), который сочетает в себе контроль и планирование в отношении целей действий и восприятия.

Метод с моделью предсказания уже давно используется в робототехнических системах. Более гибкий метод прогнозирующего интегрального пути, основанный на выборке управления, который может быть оптимизирован в соответствии с общими критериями стоимости, выпуклыми и невыпуклыми, был впервые реализован в работе [269]. В работе расширен этот метод, проинтегрировав его с методом одновременной локализации и картографирования (SLAM, от англ. method of Simultaneous Localization And Mapping), чтобы он был применим к более широкому классу стохастических роботизированных систем с динамическими ограничениями. Различные применения метода SLAM приведены, например, в работах [164, 167, 171, 267]. Все эти подходы формулируют метод как метод решения задачи апостериорной оценки максимума и часто используют факторные графы [144] для установления взаимозависимости между переменными.

Метод активного SLAM включает интеграцию с распознаванием объектов. В то же время структура метода оптимизирует цели восприятия для надежного определения ключевых точек, замыкания цикла путем выборки траектории, максимального тестирования точек интереса кадра. Ключевой кадр должен соответствовать определенному тесту, который обычно содержит три перемещения, три поворота и один параметр масштабирования. Когда у кандидата достаточно проверок, мы уверены, что цикл найден.

Общий критерий стоимости данного метода должен учитывать как цели восприятия, так и действия по планированию движения. Построение такого функционала затруднено возможными конфликтами между целью восприятия и целью действия. Предложенный метод, учитывающий распознавание значимых ориентиров для цели восприятия, потребует выбора такого движения, чтобы сделать ориентир как можно более заметным для тестирования ключевых кадров.

Для этого в функцию затрат вводится условие перехода к целям восприятия.

4.6.2 Обзор метода SLAM

Метод одновременной локализации и картографирования (SLAM) – это задача, при которой автономный агент (робот) строит карту окружающей среды и одновременно вычисляет свою траекторию относительно этой карты. Эта задача является одной из центральных в области мобильной робототехники и компьютерного зрения.

Обозначим

– $x_t \in \mathbb{R}^n$ – состояние робота (координаты, ориентация, скорость и т.д.) в момент времени t;

– $u_t \in \mathbb{R}^m$ – управляющее воздействие (например, команды двигателя);

– $z_t \in \mathbb{R}^p$ – наблюдаемые данные с сенсоров (лидар, камера, GPS и т.д.);

– $m \in \mathbb{R}^{d \times N}$ – карта среды, состоящая из N объектов или точек.

Основная идея метода – найти

$$p(x_{0:t}, m \mid z_{0:t}, u_{0:t}),$$

т.е. оценить совместное распределение вероятностей траектории робота и карты по последовательности наблюдений и управлений.

Известна модель движения

$$x_t = f(x_{t-1}, u_t, w_t)$$

где w_t – шум процесса.

Известна модель наблюдения

$$z_t = h(x_t, m, v_t)$$

где v_t – шум измерений.

Задано апостериорное распределение

$$p(x_{0:t}, m \mid z_{0:t}, u_{0:t}) \propto p(z_t \mid x_t, m) \cdot p(x_t \mid x_{t-1}, u_t) \cdot p(x_{0:t-1}, m \mid z_{0:t-1}, u_{0:t-1})$$

Среди известных методов решения SLAM следует отметить

- Расширенный фильтр Калмана SLAM (Extended Kalman Filter SLAM, EKF-SLAM) [266]. Этот метод предполагает гауссовские шумы, карта моделируется как вектор $m = [m_1, ..., m_N]$, где $m_i \in \mathbb{R}^2$ – координаты особенности, объединённое состояние $X_t = [x_t, m]^T$, уравнения фильтра Калмана используются для обновления оценки X_t.

- Быстрый SLAM / Фильтр частиц (FastSLAM / Particle Filter) [267] использует метод частиц

$$p(x_{0:t}, m \mid z_{0:t}, u_{0:t}) = \sum_{s=1}^{S} w_t^{(s)} \delta(x_{0:t} - x_{0:t}^{(s)}) p(m \mid x_{0:t}^{(s)}, z_{0:t}),$$

где: S – количество частиц, $w_t^{(s)}$ – вес частицы s, $\delta(\cdot)$ – дельта-функция.

Метод SLAM реализуется согласно блок-схеме.

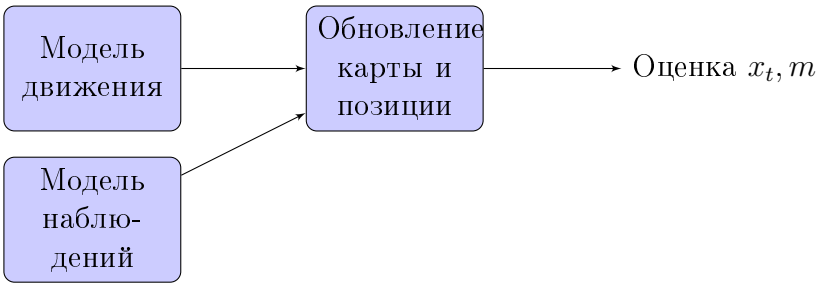

Рис. 4.42: Блок-схема работы алгоритма SLAM

Таким образом, SLAM представляет собой сложную задачу оценки состояния в условиях неопределённости, объединяющую локализацию и построение карты. Методы, такие как EKF-SLAM и FastSLAM, позволяют решать эту задачу эффективно и точно, что делает их основой современных автономных систем.

В данной работе используется следующий метод SLAM. Предположим, нужно оценить неизвестную переменную X, в методе SLAM переменная X обычно включает траекторию робота (в виде дискретного набора позиций) и положение ориентиров в окружающей среде. Нам дан набор измерений $Z = \{z_k : k = 1, \ldots, m\}$ таких, что каждое измерение может быть выражено как функция X, т.е.

$$z_k = h_k(X_k) + \varepsilon_k$$

где $X_k \subseteq X$ – подмножество переменных, $h_k(\dot{})$ – известная функция (модель измерения или наблюдения), а ε_k – случайный шум измерения.

При апостериорной оценке максимума мы оцениваем X, вычисляя присвоение переменных X^*, которое апостериори достигает максимума $p(X|Z)$:

$$X^* = \underset{X}{\operatorname{argmax}}\, p(X|Z) = \underset{X}{\operatorname{argmax}}\, p(Z|X)p(X), \tag{4.43}$$

где равенство следует из теоремы Байеса.

В (4.43) $p(Z|X)$ – это вероятность измерений Z с учетом присвоения X, а $p(X)$ – это априорная вероятность превышения X. Априорная вероятность включает в себя любую предварительную информацию о X; в отсутствие предварительной информации $p(X)$ становится постоянной (равномерное распределение), что несущественно и может быть исключено из оптимизации. В этом случае максимальная апостериорная оценка сводится к оценке максимального правдоподобия.

Предполагая, что измерения независимы (т.е. соответствующие шумы некоррелированы), задача (4.43) сводится к

$$X^* = \underset{X}{\operatorname{argmax}}\, p(X) \prod_{k=1}^{m} p(z_k|X) = \underset{X}{\operatorname{argmax}}\, p(X) \prod_{k=1}^{m} p(z_k|X_k), \tag{4.44}$$

где справа мы заметили, что z_k зависит только от подмножества переменных в X_k.

Задача (4.44) может быть интерпретирована в терминах вывода факторных суперграфов [270], где переменные соответствуют узлам факторного графа.

Вероятность измерения в (4.44) вычисляется по формуле

$$p(x_k|X_k) \propto e^{-\frac{1}{2}||h_k(X_k)-x_k||^2_{\Omega_k}}.$$

Поскольку максимизация апостериорного значения совпадает с минимизацией отрицательного логарифмического апостериорного распределения, максимальная апостериорная оценка в (4.44) принимает вид

$$X^* = \arg min_X \left[-\lg \left(p(X) \prod_{k=1}^{m} p(z_k|X_k) \right) \right], \tag{4.45}$$

что является нелинейной задачей наименьших квадратов.

Задача минимизации (4.45) обычно решается с помощью последовательной линеаризации, например, методами Гаусса-Ньютона или Левенберга-Марквардта.

Таким образом, среда в окне оптимизации должна быть статичной. Мы хотим, чтобы скорость алгоритма и работы была значительно выше скорости динамических препятствий, в этом случае среду можно считать статичной, включая динамические элементы в состав модели.

4.6.3 ASLAM-MPPI

В данной работе впервые предполагается использовать модифицированный метод SLAM а именно активный SLAM, и его объединение с методом MPPI.

Задача управления движением робота с целью минимизации неопределенности его отображения на карте и локализации обычно называется методом активного SLAM. Это определение исходит из хорошо известного активного восприятия [268]. Теоретические подходы к управлению для метода активного SLAM включают использование прогнозирующей модели управления [269, 270].

Математическая модель объекта управления имеет вид

$$\dot{x}_{i+1} = f(x_i, v_i),$$

где x_i – вектор состояния объекта, ν_i – фактический вектор управления (вектор управления включает скорости и вращение рулевого колеса), $x_i \in \mathbb{R}^n$, $v_i\ N(u_i, \Sigma)$ – гауссовский шум с математическим ожиданием u_i, $u_i \in U \subset \mathbb{R}^m$, $m \leq n$, U – замкнутое ограниченное множество.

Управление должно проходить через контроллер более низкого уровня со стохастическим добавлением в виде гауссовского шума, в результате фактический вектор управления определяется как $V = (v_0, v_1, \ldots, v_{\tau-1}$ – это последовательность входов после определенного количества шагов T. Необходимо вычислить среднее управление $U = (u_0, u_1, \ldots, u_{\tau-1})$, которое минимизирует функцию затрат

$$J(U) = E\left[q_1(x_T) + \sum_{t=0}^{T-1}\left[q_2(x_t) + \lambda u_t^T \Sigma^{-1} u_t\right]\right],$$

где $q_1(x_T)$ – конечное значение состояния, а $q_2(x_t)$ – мгновенное значение состояния.

Алгоритм ASLAM-MPPI начинается с инициализации (локализации методом SLAM) вектора состояния x_0. Расчет управления начинается с рассмотрения последовательности побега из предыдущей итерации U_{k-1}, параллельного моделирования (тысяч) траекторий, каждая из которых имеет различную последовательность управления V^m, где m – реализация случайной траектории.

Затраты на управление собираются для каждого развертывания и сопоставляются с весами траекторий

$$\Omega(V_m) = e^{\left(-\frac{1}{\lambda}\left(S(V_m) - \sum_{t=0}^{T-1}(v_t)^T \Sigma^{-1} v_t - \rho\right)\right)},$$

где ρ задает минимальное значение стоимости среди всех выбранных траекторий, предназначенных для исключения алгоритма, $S(V_m) = q_1(x_T^m) + \sum_{t=0}^{T-1} q_2(x_t^m)$ является частью стоимости, в зависимости от состояния. Квазиоптимальное управление на каждом временном шаге вычисляется как

$$U(T, K) = U(t, K-1) + \frac{\sum_{t=1}^{T-1} \Omega(V_m)\xi^m}{\sum_{t=1}^{T-1} \Omega(V_m)},$$

где ξ^m $N(0, \Sigma)$ – гауссов шум с нулевым средним значением, M – количество реализаций траектории.

Алгоритм ASLAM-MPPI позволяет в режиме реального времени обрабатывать сложную нелинейную динамику беспилотного транспортного средства (БТС) и окружающей среды на горизонте прогнозирования, но, как и классический алгоритм модельного прогнозирования, он страдает от ухудшения стабильности при моделировании динамики, отличной от истинной динамики БТС. Стратегия решения этой проблемы может заключаться в идентификации модели стохастическими методами на основе структуры нейронной сети.

С другой стороны, если прогноз состояния БТС без водителя отличается от состояния БТС без водителя, полученного с помощью метода SLAM, система принимает решение о сбоях и предоставляет механизмы восстановления.

Алгоритм отслеживает, что система распознавания местоположения сгенерировала неправильные ограничения, удаляет их при необходимости и пересчитывает оценку состояния. В частности, он принимает решение о закрытии контура на основе информации о прогнозе состояния. Алгоритм ASLAM-MPPI позволяет выбирать оптимальную траекторию на основе функции стоимости в окне прогноза продолжительностью около 1,5 секунд (время, использованное в эксперименте) в соответствии с навигационной функцией. В этом случае порог несоответствия состоянию БТС устанавливается в соответствии с моделью с состоя-нием, полученным с помощью SLAM на уровне, превышающем гауссовский шум ошибки состояния

$$\mu = \frac{1}{N} \sum_{i=1}^{N} r_i,$$

$$\sigma = \sqrt{\frac{N}{N-1} \sum_{i=1}^{N} (r_i - \mu)^2},$$

где μ – оценка параметра сдвига, σ – оценка параметра масштаба, r_i – количество состояний, N – общее количество элементов в траектории.

На рис. 4.43 показаны расхождения между состоянием БТС, т.е. стохастический характер управления без обратной связи. Три линии – это три разных запуска БТС с одинаковым управлением.

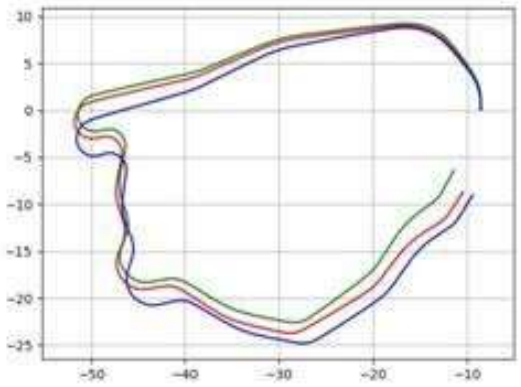

Рис. 4.43: Расхождения в состоянии БТС на уровне гауссовского шума, при одинаковом управлении без обратной связи

При глобальном планировании желательно, чтобы значимые ориентиры (получение роботом однозначных функций) появлялись в области навигационной траектории, но не являлись препятствием. В то же время глобальная траектория может быть оптимальной в смысле безопасности от сбоев.

Оптимальное планирование траектории для автономных роботов изучается уже давно. В работе [147] разработана реактивная навигационная система для обеспечения нескольких целей. Каждая цель рассматривается как поведение, и их задача состоит в том, чтобы взвесить набор дискретных элементов управления движением на основе его ожидаемой полезности. Выгоды от каждого поведения также взвешиваются центральным арбитром в зависимости от текущей миссии системы.

Чтобы иметь возможность управлять роботом в режиме реального времени, нам нужно достаточно быстро обновить карту [166]. Это очень важно для роботов с быстрой динами-кой, таких как ровер, представленный на рис. 4.44 (слева). Чтобы соответствовать этим требованиям, необходимо использовать ускорения, например, с помощью графического процессора. Для уточнения и регулировки положения и ориентации мобильного робота предлагается использовать цветные линии и маркеры (рис. 4.44 справа) при инициализации ровера и замыкании цикла. На рис. 4.44 (в центре) показан вид полигона с камеры, установленной на ровере.

Рис. 4.44: Ориентиры в виде цветных линий, ярких конусов и маркеров

Для динамических препятствий в предыдущих работах, например [244], мы использовали генератор траекторий и предварительно рассчитанные траектории, которые были выбраны с помощью функции стоимости. Алгоритм ASLAM-MPPI позволяет нам генерировать тысячи траекторий в реальном времени и выбирать ту, которая учитывает динамическое ограничение. На рис. 3 представлена карта затрат. Множество светлых линий – это траектории метода ASLAM-MPPI, визуализированные на RVIZ. Оптимальная траектория – тонкая черная линия.

Алгоритм метода ASLAM-MPPI

Алгоритм метода ASLAM-MPPI работает следующим образом:

– Используя алгоритмы ASLAM (например в [269] - [267]), мы получаем локальное состояние БТС и карту занятости с учетом динамических препятствий;

– Алгоритм A-star вычисляет глобальную траекторию. Активный SLAM включает терминальные состояния в круг значимых ориентиров, например, маркеров;

– Алгоритм MPPI моделирует (прогнозирует) из начального состояния тысячи траекторий с учетом динамики БТС в направлении глобальной траектории. Окно прогнозирования зависит от расчетной скорости динамических препятствий;

– Используя функцию стоимости, оптимальная траектория рассчитывается в соответствии с заданным критерием оптимальности;

– Применяется управление, соответствующее оптимальной траектории;

– Затем алгоритм повторяется;

– Если локализация БТС превышает порог расхождения с прогнозом локального состояния алгоритма ASLAM-MPPI, система локализации корректируется.

4.6.4 Результаты применения метода ASLAM-MPPI

Конечно-разностные уравнения модели мобильного робота с шасси геометрии Аккермана имеют следующий вид

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{pmatrix} = \begin{pmatrix} x_k + v_k \cos\theta_k \\ y_k + v_k \sin\theta_k \\ \theta_k + \frac{v_k}{H} \operatorname{tg}\omega_k \end{pmatrix},$$

где (x_k, y_k) – координаты мобильного робота; v_k – его линейная скорость; ϑ_k – угловая ориентация робота; ω_k – угловая скорость; H – расстояние между передней и задней осями колес мобильного робота.

Если $f(x) < 0$, где $f(x)$ – условие распознавания меток в алгоритме, то алгоритм переключается на другую функцию затрат, такую как

$$G(Z_k) = F(Z_k) + (1 - \vartheta(f(x)))y_5,$$

где $y_5 = \sin(x + \Delta x, y + \Delta y)$, здесь$(x, y)$ – текущие координаты, $(\Delta x, \Delta y)$ – расстояние до распознанной метки, ϑ – функция Хевисайда

$$\vartheta(a) = \begin{cases} 0, & a < 0, \\ 1, & a \geq 0. \end{cases}$$

К описанной задаче применяется метод ASLAM-MPPI.

– Используя алгоритмы ASLAM (например в [269] - [267]), получаем локальное состояние БТС и карту занятости (рис. 4.45) с учетом динамических препятствий;

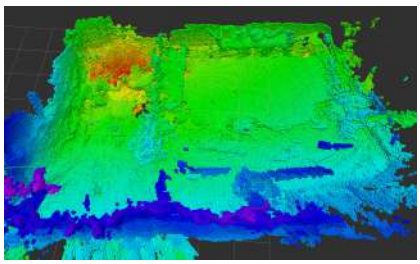

Рис. 4.45: Карта занятости полигона

Метод ASLAM-MPPI был протестирован в Робототехническом центре Российской академии наук, где развернут полигон, имитирующий промышленный объект с конструкциями различной сложности, пандусами в виде крутой горки,

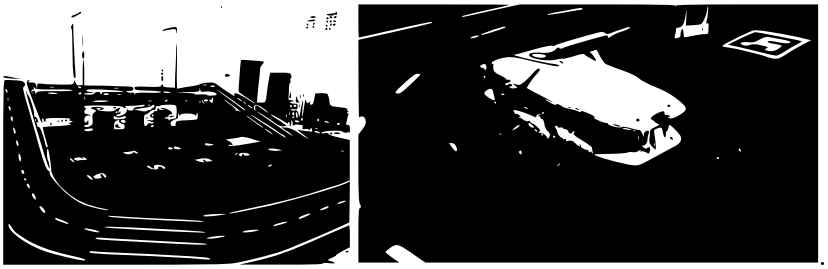

Рис. 4.46: Полигон и ровер

мостом и другими ограничениями, рис. 4.46 слева. Беспилотное транспортное средство, используемое в экспериментах показано на рис. 4.46 справа.

Динамика БТС была реализована на структуре модели нейронной сети, имеющей следующий математический вид

$$g_i(\varphi(k,\mathbf{q}),\mathbf{q}) = \hat{y}_i(k|\mathbf{q}) = \hat{y}_i(k|w,W) = \\ = F_i\left(\sum_{j=1}^{n_h} W_{ij} f_j\left(\sum_{l=1}^{n_\varphi} w_{jl}\varphi_l + w_{j0},\right) + W_{i0}\right),$$

где $\mathbf{q}$ – настраиваемые параметры нейронной сети, включая весовые коэффициенты и смеще-ния (w_{jl}, W_{ij}); $F_i(x) = ax$ – функция активации нейронов в выходном слое $a = const$; n_h – количество нейронов в скрытом слое; $f_i(x) = \tanh(x)$ – функция активации нейронов скрытого слоя; n_φ – количество нейронов в скрытом слое, входные данные.

Нейронная сеть для метода ASLAM-MPPI включает 2 скрытых слоя с двумя нелинейностями, что означает, что общая конфигурация сети составляет 6-32-32-4. Она принимает в качестве входных данных 4 переменные состояния (угол наклона, продольную скорость, поперечную скорость, курс), а также управляемое рулевое управление и управление скоростью, и они выводят производную по времени от переменных состояния.

Нейронная сеть была обучена на данных, собранных на полигоне (рис. 4.46), в режиме ручного управления БТС. Для обучения использовался метод стохастического градиентного спуска. Обучение нейронной сети выполнялось с использованием инфраструктуры Центра коллективного пользования «Высокопроизводительные вычисления и большие данные» (ЦКП «Информатика») ФИЦ ИУ РАН (г. Москва). Результаты (рис. 4.47) передавались на встроенную вычислительную платформу, в качестве которой использовалась NVIDIA Xavier NX, что позволило распараллелить нейронную сеть.

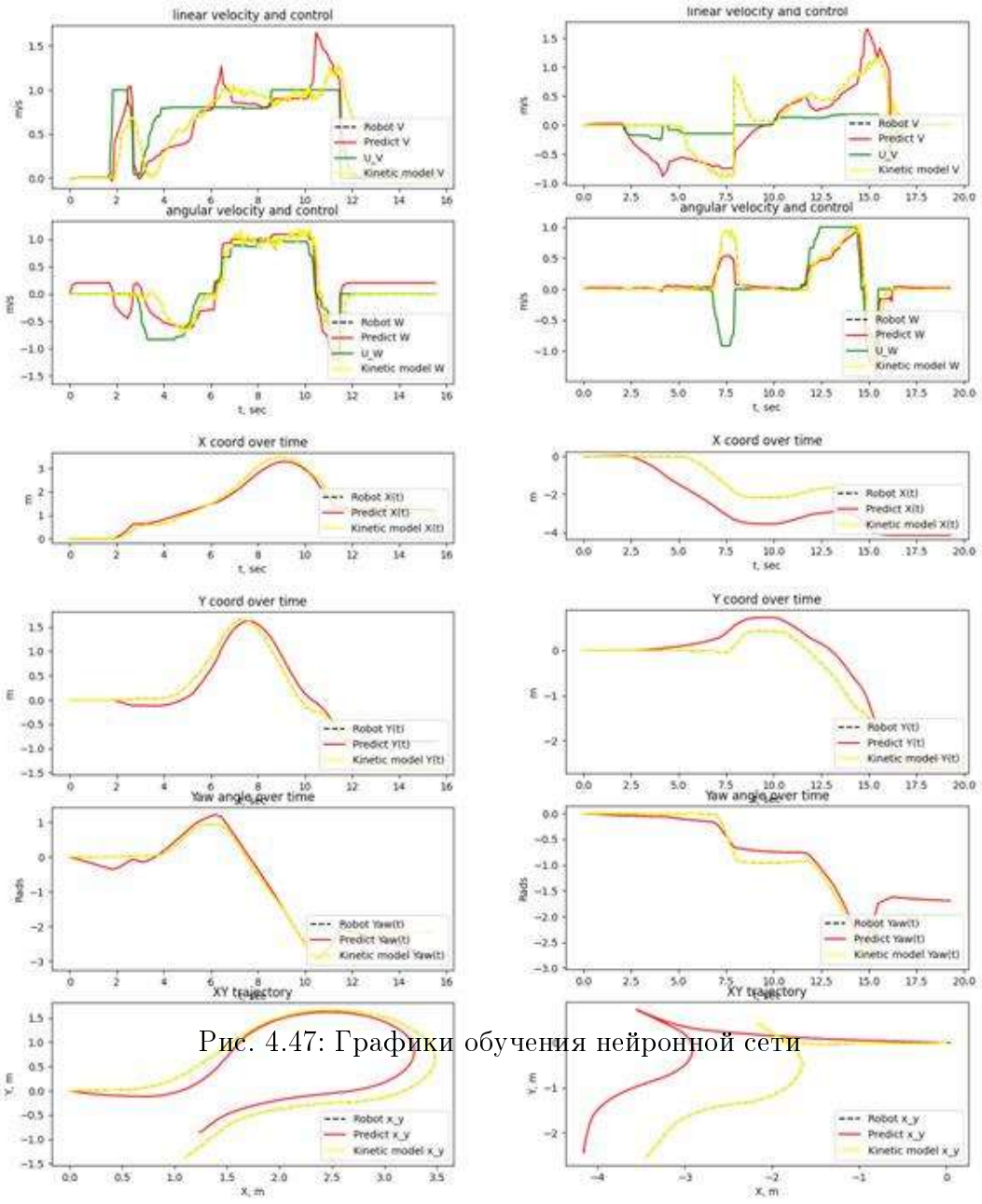

Рис. 4.47: Графики обучения нейронной сети

Дальнейшие эксперименты проводились с использованием ROS2 на компьютере Linux с процессором Intel Core i7, на котором был развернут алгоритм MPPI. Команды управления передавались на БТС.

Глобальная траектория определялась в виде сгенерированных точек в пространстве, расположенных на пунктирной центральной разметке трассы (рис. 4.46).

Измерение абсолютной ошибки траектории SLAM метода показало одинаковую ошибку на горке и на горизонтальной плоскости и составляет в среднем 0,99 процента от длины пройденного пути. Длина дорожки, проходящей по горке, составила 23,3 м. Погрешность отклонения более 0,03 м приводит к несчастным случаям на поворотах горки. Таким образом, критерием качества и контроля метода ASLAM-MPPI является количество дорожек, проходящих через горку. В сложных условиях, когда пространственные ограничения сильно сужают пространство допустимых перемещений, стратегия выбора пространства состояний для опти-мизации более эффективна, чем выборка в пространстве управления. Эксперимент показал, что прохождение горки с минимальной ошибкой в первую очередь зависит от системы локализации робота.

В экспериментах в качестве системы SLAM использовался Intel Realsense T265 – автономный датчик слежения с 6 степенями свободы, который запускает визуально-инерционный алгоритм SLAM, что позволяет точно оценить движение БТС с 6 степенями свободы и определить положение и ориентацию БТС с частотой 200 Гц., что в 10 раз быстрее, чем локализация с помощью самого известного алгоритма ORB-SLAM2 [269].

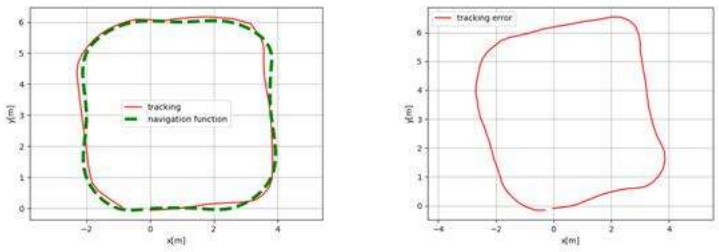

Рис. 4.48: (слева) Функция навигации; (справа) Ошибка идентификации начального положения, исправленная с помощью маркеров для метода ASLAM-MPPI

Метод ASLAM-MPPI показал в среднем более 7 кругов на трассе с горкой без сбоев. Усредненная траектория вдоль глобальной траектории (сплошная линия) и глобальная траектория (прерывистая функция) показаны на рис. 4.48 слева. Метод SLAM на базе камеры слежения Intel Realsense T265 позволяет сделать только 1 круг, а на втором происходит сбой из-за накопленной ошибки,

например, рис. 4.48 (справа), причем накопление ошибок приводит к невозможности преодоления поворотов на горке. Наклон горки составляет 45 градусов и представляет собой серьезное ограничение для мобильного робота.

В результате нашей работы мы провели полевые эксперименты с новым методом активной одновременной локализации и картографирования на основе модели прогнозирующего интегрального пути на реальном БТС.

Метод позволяет БТС следовать квазиоптимальной траектории в соответствии с заданным критерием качества, предотвращая сбои в системе локализации из-за выбросов или вырождения информации с датчиков, что очень важно для безопасного использования транспортного средства.

Активный SLAM позволяет нам обеспечить более надежную долгосрочную связь данных в интерфейсе, она включает в себя замыкание цикла, обнаружение и проверку. В результате метод ASLAM-MPPI демонстрирует значительную производительность (скорость алгоритма, отказоустойчивость) по сравнению, например, с навигационным стеком ROS [270].

Метод ASLAM-MPPI был протестирован в Робототехническом центре Российской академии наук, где развернут полигон, имитирующий промышленный объект со сложными конструкциями, пандусами в виде крутой горки, мостом и другими ограничениями. На рис. 4.49 показана часть трассы в форме моста и БТС, которое мы используем в экспериментах.

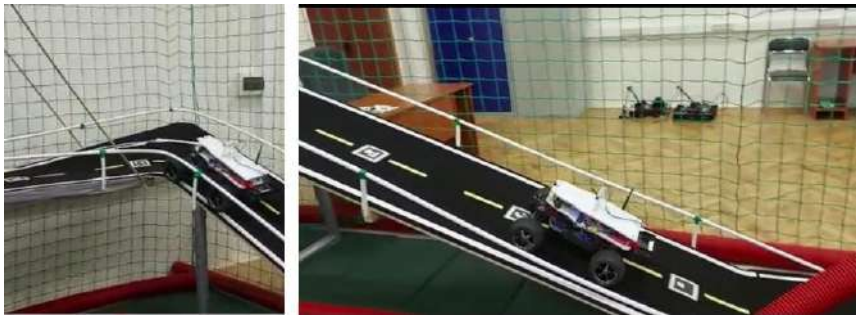

Рис. 4.49: Ориентиры в виде цветных линий, ярких конусов и маркеров

4.6.5 Выводы

В работе представлен новый метод активной одновременной локализации и картографирования на основе модели прогнозирующего интегрального пути и проведенные полевые эксперименты на реальном БТС, продемонстрировав новые

возможности, заключающиеся в способности соответствовать динамике системы с ограничениями привода и минимизировать прогнозируемые затраты, в том числе конфликтующие на сложном полигоне.

Метод позволяет БТС следовать квазиоптимальной траектории в соответствии с заданным критерием качества, предотвращая сбои в системе локализации из-за выбросов или вырождения информации с датчиков, что очень важно для безопасного использования транспортного средства.

Активный SLAM позволяет нам обеспечить более надежную долгосрочную связь данных в интерфейсе, она включает в себя замыкание цикла, обнаружение и проверку.

Метод позволяет БТС следовать оптимальной траектории в соответствии с заданным критерием качества, предотвращая сбои в системе локализации из-за выбросов или вырождения информации, что очень важно для безопасного использования БТС.

Метод обеспечивает более надежную долгосрочную связь данных в интерфейсе, включает замыкание цикла, обнаружение и проверку, а также интеграцию с распознаванием объектов. В то же время предложенный метод оптимизирует цели восприятия для надежного определения ключевых точек для замыкания цикла путем выборки траектории для максимального тестирования точек интереса. Классические схемы управления не способны предсказать затраты и соответствующую траекторию (например, ПИД-регуляторы). Кроме того, формулировка совокупности восприятия и действия как задачи оптимизации имеет преимущества в надежности системы управления.

Метод ASLAM-MPPI можно сравнить с методом [149], основанным на методе с моделью предсказания с глубокими сверточными нейронными сетями для понимания сцен в реальном времени. Полностью сверточная сеть способна изучать сложное внеплоскостное преобразование, необходимое для проецирования пикселей наблюдаемого изображения на плоскость земли и прогнозирования карты контролируемых объектов в области перед БТС. Решение состоит в том, чтобы обучить глубокую нейронную сеть преобразовывать визуальные данные с одной монокулярной камеры в стоимость функции в локальной системе координат, ориентированной на робота, превосходны. Затем эта карта затрат может быть непосредственно введена в алгоритм управления прогностической моделью. Однако недостатком метода могут быть не посещаемые районы, где обучающая выборка может не иметь достаточно надежных карт затрат. Метод с моделью предсказания использует только ту информацию, которую она может видеть на выходе нейронной сети, и заранее планирует в течение 1,5 секунд выдать управляющий сигнал. Этот 1,5-секундный временной горизонт приводит к чрезвы-

чайно робкому поведению, потому что доступный вид вперед имеет небольшое расстояние.

Метод ASLAM-MPPI использует картографическую информацию, основанную на всем горизонте событий, который может быть скорректирован с поступлением новой информации. Таким образом, он более устойчив к потере информации, чем MPPI с глубокими сверточными нейронными сетями. Кроме того, значительным вкладом работы на самом деле является разработка технологии метода активного SLAM для предотвращения сбоев в системе локализации. Значительным результатом является решение практической задачи реактивного планирования БТС на сложном полигоне, имитирующем производственную среду. Для этого глобальная траектория проходила через горку со сложными поворотами, по которой она не преодолевалась в ручном режиме БТС-управления без обучения оператора. MPPI с глубокими сверточными нейронными сетями не может распознавать подъемы и спуски с горки без дополнительной информации. Таким образом, карта затрат, созданная этим методом, не может быть затем непосредственно введена в алгоритм управления прогностической моделью. Классические схемы управления (например, PID, LQR, iLQR) могут надежно отслеживать глобальную траекторию, но не являются стабильными в случае сбоев в системе локализации.

В результате метод ASLAM-MPPI демонстрирует значительную производительность (скорость алгоритма, отказоустойчивость) по сравнению, например, с навигационным стеком ROS [271].

4.7 Навигация беспилотного транспортного средства

В данном разделе диссертационной работы приведены новые методы навигации мобиного робота. В разделе 4.7.1 рассмотрен вариант глобальной навигации. Он необходим для подтверждения и уточнения метода SLAM, т.к. одомнтрия со временем расходится и требуется уточнение местоположения мобильного робота. В разделе 4.7.2 предложен вариант навигации по внутренней модели. Речь идет о нахождении роботом своего местоположения при отсутствии связи.

4.7.1 Навигация беспилотного транспортного средства с помощью сетевого оператора и ARUCO-меток

Aruco – это библиотека для быстрого обнаружения квадратных маркеров на изображении, используемая в задачах компьютерного зрения, робототехники и дополненной реальности. Аруко-маркеры позволяют точно определять их положение и ориентацию в 3D-пространстве.

Aruco-маркер представляет собой черно-белый квадратный код, включающий внешнюю рамку для обнаружения и внутренний двоичный код, уникальный для каждого маркера. На рис. 4.50 представлен пример структуры маркера

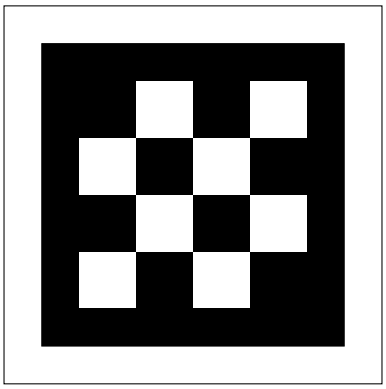

Рис. 4.50: Пример Aruco-маркера размером 4×4

Обозначим
– $M \in \{0,1\}^{n \times n}$ – матрица пикселей маркера;
– $c = (x_c, y_c)$ – координаты центра маркера на изображении;
– $R \in SO(3)$, $t \in \mathbb{R}^3$ – ориентация и положение маркера в 3D-пространстве относительно камеры;
– $K \in \mathbb{R}^{3\times3}$ – матрица внутренних параметров камеры.
Процесс обнаружения включает в себя следующие этапы:

1. обнаружение четырёх углов маркера на изображении;
2. идентификация ID по внутреннему коду;
3. решение задачи заключающейся в нахождении значений R и t таких, что

$$K[R|t]X_i = x_i, \quad \forall i,$$

где X_i – известные 3D-точки маркера, x_i – их проекции на изображение.

Aruco-маркеры поддерживаются в библиотеке OpenCV. Их реализация на языке Python может быть следующей:

```
import cv2
import numpy as np

aruco_dict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_6X6_250)
parameters = cv2.aruco.DetectorParameters_create()

frame = cv2.imread("marker_image.png")
corners, ids, rejected = cv2.aruco.detectMarkers(frame, aruco_dict,
parameters=parameters)

rvecs, tvecs, _ = cv2.aruco.estimatePoseSingleMarkers(corners, 0.05,
K, distCoeffs)
```

Aruco-маркеры предоставляют эффективное и точное решение для локализации объектов в 3D-пространстве. Благодаря своей простой структуре и надежному алгоритму обнаружения они широко используются в робототехнике, навигации и системах дополненной реальности.

Рассмотрим задачу (4.2)-(4.3).

Необходимо найти управление в форме функции от координат пространства состояний

$$u = h(x), \tag{4.46}$$

где $h(x)$ – искомая синтезирующая функция, $h(x) : \mathbb{R}^n \to \mathbb{R}^m$, удовлетворяющая следующему свойству:

$$\forall x \in \mathbb{R}, h(x) \in U.$$

Функция должна обеспечивать минимум функционалу, описывающему точность движения по траектории

$$J = \frac{1}{t_f r} \int_0^{t_f} \sqrt{\sum_{i=1}^{r} \varphi(x(t))\, dt} \to \min. \tag{4.47}$$

Для решения задачи (4.2)-(4.3) синтеза системы управления движением объекта по пространственной траектории используем управление, полученное методом сетевого оператора.

Для поиска решения, минимизируем функционал (4.4), описывающий точность движения по траектории, используем вариационный генетический алгоритм со следующими параметрами:

– размер начальной популяции – 50;
– число поколений – 10;
– число скрещиваемых пар в одном поколении – 30;
– число вариаций в одном решении – 10;
– число поколений между сменой базисного решения – 2;
– число элитарных решений – 16;
– вероятность мутации – 7.0;
– параметр для скрещивания – 4.0.

В результате синтеза было получено управление – поворот колес робота. Форма сетевого оператора, в виде списка [265], для полученного управления имеет следующий вид:

$$R = ((0, 19), (2, 0), (5, 4)), ((5, 10), (3, 5), (18, 1)). \tag{4.48}$$

Производим декодирование полученного списка в символьное выражение традиционной математической формы. В результате полученное после декодирования и упрощения решение (4.48) получается в виде:

$$u_1 = -3,5 \operatorname{arctg} \theta, \quad u_2 = const,$$

где θ – угол между направлением движения робота и направлением на цель.

В данной работе объектом управления является беспилотный автомобиль, построенный с использованием автомобильного шасси масштаба 1/10, рис. 1а, который весит 2 кг и способен развивать скорость до 60 м/с (рис. 4.51).

На автомобиль установлена вычислительная платформа NVIDIA Jetson TX2 с датчиком, видеокамерами и контроллером, позволяющим регулировать линейную скорость и скорость поворота колес.

Определение положения играет ключевую роль в задаче отслеживания траектории. Для оценки состояния робота используются ArUco-метки [138], расположенные на потолке (рис. 4.52), и система технического зрения, подсоединенная к машине через плату Raspberry Pi 3.

Используется маркер ArUco как двоичная матрица 6x6. Каждая строка внутренней матрицы 4x4 состоит из 2 бит данных и 2 биты обнаружения неисправностей. Эта матрица создала 16-битный код, что позволило нам обнаружить 65536 различных маркеров [168].

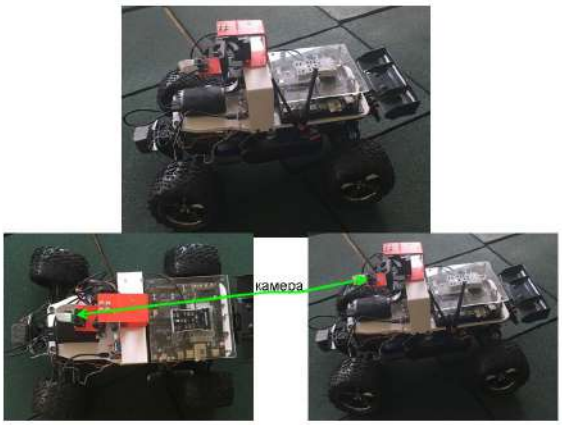

Рис. 4.51: Беспилотный автомобиль.

Рис. 4.52: ArUco-метки.

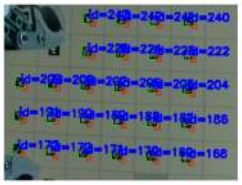

Рис. 4.53: Распознавание id ArUco-меток.

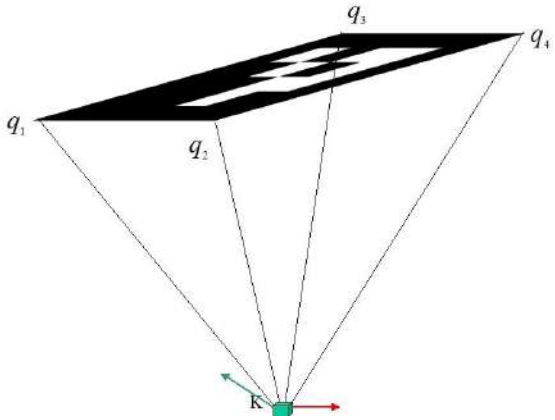

Рис. 4.54: Определение локальных координат по ArUco-меткам.

С помощью камеры, установленной на корпусе, робот распознает ArUco-метки, с помощью библиотеки OpenCV определяет id каждой увиденной метки (рис. 4.53), точки $q_1 - -q_4$, а также смещение относительно нее и рассчитывает угол θ. По установленной метрике он определяет локальные координаты.

Координаты углов маркера $q_1 - -q_4$ определяют положение робота (рис. 4.54).

Для оценки качества найденного управления робот помещается в произвольную точку поля так, чтобы в поле зрения камеры попала хотя бы одна ArUco-метка. Далее строится траектория движения робота из произвольной точки в заранее установленную конечную точку. Для аппроксимации траектории используем кубический сплайн (рис. 4.55).

Затем в начальную точку поля устанавливается робот. Запускается программа, с помощью которой согласно модели (1)-(5) находится траектория движения робота к цели – конечной точке. Движение робота и расчет угла θ производится по ArUco-меткам.

Как показали многочисленные испытания, робот просчитывает траекторию, наиболее близкую к аппроксимированной и проезжает по ней. На рис. 4.56

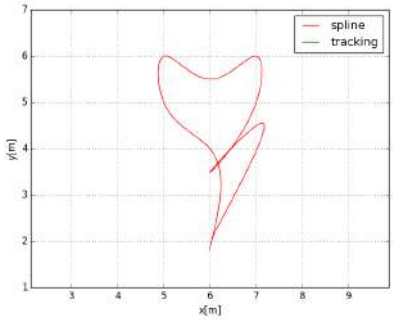

Рис. 4.55: Пример аппроксимации траектории движения робота.

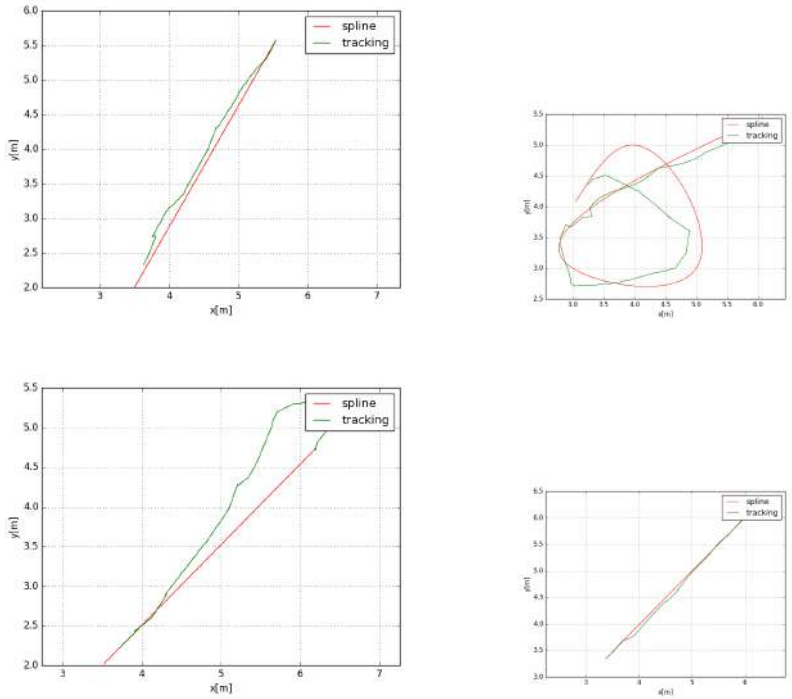

Рис. 4.56: Траектория движения робота.

приведено несколько траекторий движения робота. Черной линией обозначен сплайн – аппроксимация траектории, а серой – траектория, построенная самим роботом. Отметим, что развороты робота для движения к цели аппроксимация не учитывает.

Таким образом, получается, что из любой точки пространства, из которой

робот видит хотя бы одну ArUco-метку, он находит заданную конечную точку с помощью управления, синтезированного методом сетевого оператора.

4.7.2 Навигация по внутренней модели

В основе методов обеспечения отказоустойчивости лежат адаптивные методы перестройки структуры системы управления, а также параметров системы управления. Цель исследований в данной области – разработать метод идентификации динамики объекта управления (мобильного робота) для создания его эталонной модели. Для решения поставленной задачи авторы разработали метод, который с помощью эталонной модели позволяет идентифицировать динамику модели, а также корректировать систему управления при возникновении сбоев. Метод использует систему на основе динамической модели, такой как ВИМ - внутренняя интеллектуальная модель, одометрия которой не подвержена сбоям и недостаткам вышеуказанных систем. Система навигации с поддержкой ВИМ также используется при переходе с автоматического управления на ручное и обратно.

Структурная схема метода представлена на 4.57.

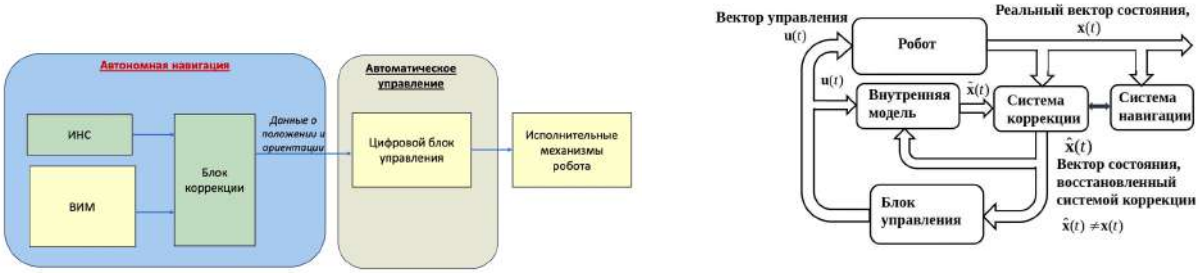

Рис. 4.57: Структурная схема метода.

Для автономного управления роботом в каждый момент времени необходимо получать информацию о его состоянии. Эту информацию роботу передают системы навигации, например (ГЛОНАСС и др.). В предлагаемом методе создания отказоустойчивой системы навигации вместе с роботом вычисляются «виртуальные» измерения от ВИМ совместно с ИНС и системой коррекции. Система коррекции представляет собой последовательно соединенные фильтры.

При обнаружении внезапных препятствий система управления переключается в режим объезда препятствий. Во время объезда препятствий необходимо получить позиционирование для возврата робота на заданную траекторию движения для выполнения операции. Информацию для позиционирования пред-

ставляет система ориентации (ВИМ+ИНС, система коррекции и система навигации).

Система коррекции представляет собой интеллектуальное устройство, которое отслеживает расхождение навигационной информации с информацией от внутренней модели и, при сбоях навигационной системы, принимает решение о коррекции, а в случае отказов – о её перезагрузке.

Оценка точности модели осуществлялась с помощью критерия среднеквадратического отклонения σ, т.е. отклонения прогнозируемых значений от реальных в каждом цикле прогноза:

$$\sigma = \frac{\sum_{t \in P} (y_t - \bar{y}_t)^2}{\sum_{t \in N} \bar{y}_t^2},$$

где N – число членов выборки; P – число членов прогноза; y_t, $\bar{y}_t$ – реальные и прогнозируемые величины.

Метод обеспечения отказоустойчивости системы автономной навигации на основе эталонных моделей был разработан и апробирован в Робототехническом центре ФИЦ ИУ РАН (рис. 4.58, а) на реальных беспилотных мобильных роботах, оснащенных современной визуально-инерциальной системой локализации и картографирования, которая подвержена сбоям в результате окклюзий и вибраций, как и все подобные системы (рис. 4.58, б). Для проведения натурных экспериментов был специально оборудован полигон, имитирующий сложную производственную среду, которую можно дополнить различными сложными ограничениями, а также внести помехи для датчиков роботов, чтобы вызывать сбои и отказы для тестирования метода.

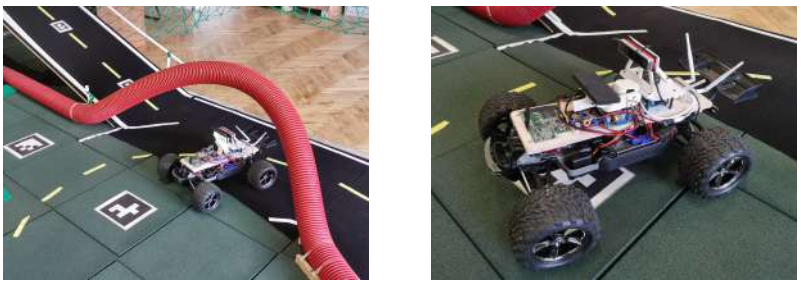

Рис. 4.58: а. Робототехнический центр ФИЦ ИУ РАН; б. Беспилотный мобильный робот.

Динамика беспилотного транспортного средства (внутренняя модель) была реализована на модели нейронной сети, включающей три скрытых слоя с двумя

нелинейностями. В качестве входных данных нейросеть принимает четыре переменные состояния (угол наклона, продольную скорость, поперечную скорость, курс), а также рулевое управление и управление скоростью. Далее вычисляется производная по времени от переменных состояния. На рис. 4.59 представлена обученная нейронная сеть, повторяющая линейную и угловую скорости робота и его курс при ручном управлении.

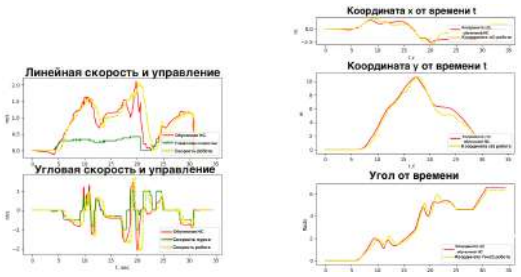

Рис. 4.59: Обученная нейронная сеть.

Предложенный в работе метод коррекции автономной навигации на основе эталонной динамической модели, позволил выполнять автоматическое управление мобильным роботом при сбоях, описанных выше.

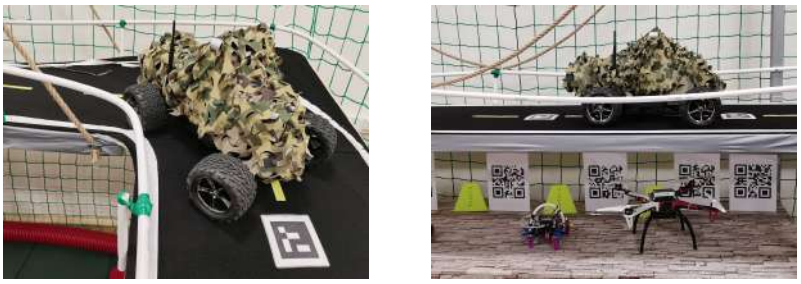

Рис. 4.60: а. Беспилотный мобильный робот; б. Беспилотный мобильный робот (сверху), дроны (снизу).

Разработанный метод идентификации динамики беспилотного мобильного робота и метод коррекции (живучести) на основе эталонной динамической модели, позволил выполнять автоматическое управление при сбоях, описанных выше.

На рис. 4.61 приведены фото траекторий, оборудованных для проведения экспериментов в Робототехническом центре ФИЦ ИУ РАН. Одна тректория

включает в себя полный круг с заездом на горку и спуском с нее. Вторая траектория нарисована на плоскости, но имеет изгиб.

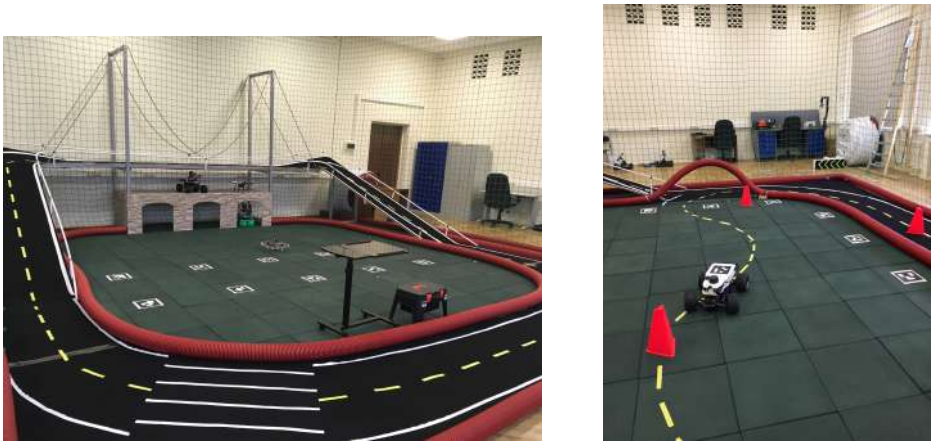

Рис. 4.61: Фото траекторий на трассе.

На рисунках 4.62 показана траектория движения реального робота (прерывистая линия) и его модели (сплошная линия). Траектория, представленная на рисунке 4.62 справа проходила по горке (рис. 4.61 слева) со скоростью 1,5 м/с. Скорость по второй траектории состявляла 2 м/с.

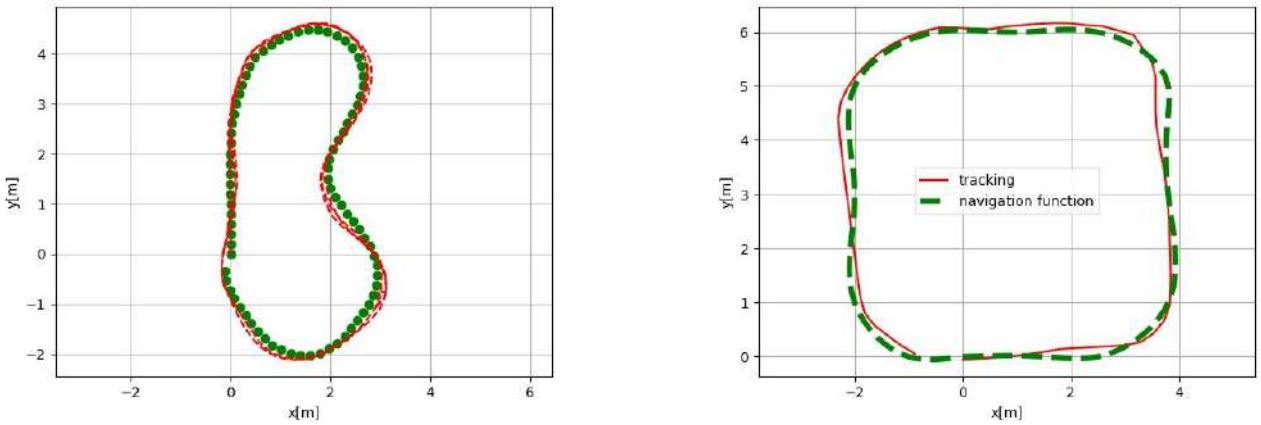

Рис. 4.62: Траектория по трассам рис. 4.61.

4.8 Заключение

В данном разделе был представлен разработанный новый метод идентификации мобильных роботов с использованием регрессионных модельных структур (нейронных сетей) и дифференциальных уравнений.

Также представлен новый метод получения функции выбора на основе эволюционного метода символьной регрессии. И приведены результаты применения предложенного метода к конкретному роботу, движущемуся по плоскости с переменной высотой, динамическими и статическими препятствиями и ограничениями скорости.

В разделе представлен новый метод активной одновременной локализации и картографирования на основе модели прогнозирующего интегрального пути и провели полевые эксперименты на реальном БТС, продемонстрировав новые возможности, заключающиеся в способности соответствовать динамике системы с ограничениями привода и минимизировать прогнозируемые затраты, в том числе конфликтующие на сложном полигоне. Метод позволяет БТС следовать квазиоптимальной траектории в соответствии с заданным критерием качества, предотвращая сбои в системе локализации из-за выбросов или вырождения информации, что очень важно для безопасного использования БТС. Классические схемы управления не способны предсказать затраты и соответствующую траекторию (например, ПИД-регуляторы). Кроме того, формулировка совокупности восприятия и действия как задачи оптимизации имеет преимущества в надежности системы управления. Метод ASLAM-MPPI использует картографическую информацию, основанную на всем горизонте событий, который может быть скорректирован с поступлением новой информации. Таким образом, он более устойчив к потере информации, чем MPPI с глубокими сверточными нейронными сетями. Кроме того, значительным вкладом нашей работы на самом деле является разработка технологии метода активного SLAM для предотвращения сбоев в системе локализации. Значительным результатом является решение практической задачи реактивного планирования БТС на сложном полигоне, имитирующем производственную среду. Для этого глобальная траектория проходила через горку со сложными поворотами, по которой она не преодолевалась в ручном режиме БТС-управления без обучения оператора. Активный SLAM позволяет нам обеспечить более надежную долгосрочную связь данных в интерфейсе, она включает в себя замыкание цикла, обнаружение и проверку.

Заключение

В диссертационной работе выполнены все поставленные задачи.

1. Проведен аналитический обзор методов решения задач оптимального управления мобильными роботами и существующих подходов к их практической реализации.

2. В диссертационной работе исследовано оптимальное движение трехколесного мобильного робота. Обсужден подход, основанный на принципе максимума. Отмечено, что прямое применение принципа максимума в модели трехколесного мобильного робота не имеет смысла из-за отсутствия регулярности, а также из-за особого режима управления относительно угловой скорости. В связи с этим предложен метод возмущений для решения нерегулярных задач оптимального управления с фазовым ограничением, целью которого является регуляризация исходной задачи. Такая тема представляется актуальной в связи с рядом прикладных задач, в частности, в робототехнике. Возмущенная задача, или ε-задача, уже является регулярной относительно фазовых ограничений. Этот факт позволил применить непрямой численный метод, основанный на классе алгоритмов стрельбы. Рассмотрен соответствующий пример, касающийся движения мобильного робота. Разработана схема регуляризации и предоставлены некоторые необходимые теоретические инструменты для дальнейшей численной реализации. Численная процедура, построенная с помощью непрямого подхода, опирается на непрерывность множителей, которая имеет место в регулярных задачах. Этот метод был применен к тестовой задаче о безинерционном движении трехколесного мобильного робота с передним приводом. Приведены результаты численного эксперимента.

Исследована задача оптимального по времени управления гусеничным мобильным роботом с инерционным рулевым управлением при наличии ограничения по состоянию, заданного единичной окружностью. Недостаток этой задачи заключается в отсутствии регулярности динамики относительно ограничения по состоянию. Поэтому был разработан определенный метод регуляризации, чтобы преодолеть такое явление, вызванное нерегулярностью. Было пред-

ложено построение аппроксимирующей последовательности регулярных *eps*-возмущенных задач. Отсюда следует, что возмущенная задача численно разрешима в рамках косвенного подхода, то есть с использованием принципа максимума, благодаря непрерывности меры-множителя, которая обеспечивается при условии регулярности. Этот факт обосновывает необходимость анализа возмущений, предпринятого в данной диссертационнй работе.

В рамках модели одноколесного велосипеда исследуется оптимальное по времени движение мобильного робота. Применяется косвенный численный метод, основанный на принципе максимума. Анализ показывает, что прямое применение принципа максимума не имеет смысла из-за отсутствия регулярности и ввиду сингулярного режима управления относительно угловой скорости. Поэтому для регуляризации исходной задачи предлагается метод возмущений. Возмущенная задача уже позволяет эффективно применять косвенный численный метод, основанный на алгоритмах съемки. Представлены результаты численных экспериментов.

3. В работе разработан, теоретически обоснован и численно проверен метод стрельбы для нахождения решения задачи о перемещении мобильного робота при фазовых ограничениях глубины 2. Метод применялся к задаче с динамикой управления ньютоновского типа. Предложенный метод позволил находить точные решения, которые впоследствии использовались для улучшения структуры пропорционально-интегрально-дифференцирующего регулятора, применяемого при нахождении траекторий мобильного робота в режиме реального времени. Представлены результаты численных экспериментов.

4. В работе описан созданный и реализованный в виде законченной программы алгоритм автоматического поиска оптимального пути между двумя заданными точками с набором фазовых ограничений в виде кругов различного радиуса, которые не пересекаются между собой. Помимо нахождения непосредственно кратчайшего пути, данный алгоритм может применяться для поиска траектории скорейшего движения колесного робота из одной заданной точки в другую с указанными ограничениями, поскольку в работе доказано, что для кинематической модели робота кратчайший путь соответствует и наиболее быстрому. Дальнейшее развитие алгоритма позволит учитывать препятствия произвольной формы, в том числе с негладкой границей и невыпуклые.

5. Разработан метод идентификации мобильных роботов с использованием регрессионных модельных структур (нейронных сетей) и дифференциальных уравнений. Разработана структурная схема системы управления мобильными роботами, обладающая запасом живучести. Методом идентификации получена эталонная модель мобильного робота, на основе которой разработан метод

коррекции системы навигации. Разработана программа идентификации беспилотного транспортного средства и программа коррекции системы навигации, которые апробированы на реальном мобильном роботе. Предложенный метод коррекции системы навигации может применяться для любых мобильных роботов. Кроме системы коррекции эталонная модель может быть применена для адаптации параметров робота при других внешних воздействиях. Таким образом, может быть увеличен запас живучести. Полученная эталонная модель на основе нейронной сети небольшого размера и дифференциальных уравнений, может быть реализована даже на маломощном устройстве. А полученный алгоритм коррекции навигационной системы достаточно точный и быстрый и может использоваться для агрессивного вождения мобильных роботов.

6. В данной работе была рассмотрена задача оптимизации управления мобильным роботом в реальном времени и представлен новый метод получения функции выбора на основе эволюционного метода символьной регрессии. Также был рассмотрен пример применения предложенного метода к конкретному роботу, движущемуся по плоскости с переменной высотой, динамическими и статическими препятствиями и ограничениями скорости. Полевые эксперименты и моделирование на тренажере демонстрируют новые возможности синтеза системы управления на основе сетевого оператора с использованием информации на выходе нейронной сети, а также плагинов gazebo. Метод позволяет достигать целей с заданным критерием качества. Параметры контроллера оптимизируются с помощью эволюционного алгоритма. Моделирование показывает улучшение работы контроллера в зависимости от оператора сети, но зависит от инициализации параметров и базового исходного решения. Также рассмотрен пример применения предложенного метода к конкретному роботу, перемещающемуся по полигону Центра робототехники со статическими препятствиями. Карта затрат подходит для планирования, когда точное местоположение на трассе не верно. Параметры контроллера оптимизируются с помощью эволюционного алгоритма метода оптимизации роя частиц. Моделирование показывает улучшение работы прогнозирующего контроллера MPPI, но зависят от инициализации параметров и выбора критерия оптимизации.

7. В диссертационной работе представлен новый метод активной одновременной локализации и картографирования на основе модели прогнозирующего интегрального пути и проведены полевые эксперименты на реальном БТС, продемонстрировав новые возможности, заключающиеся в способности соответствовать динамике системы с ограничениями привода и минимизировать прогнозируемые затраты, в том числе конфликтующие на сложном полигоне. Метод позволяет БТС следовать оптимальной траектории в соответствии с заданным

критерием качества, предотвращая сбои в системе локализации из-за выбросов или вырождения информации, что очень важно для безопасного использования БТС. Метод обеспечивает более надежную долгосрочную связь данных в интерфейсе, включает замыкание цикла, обнаружение и проверку, а также интеграцию с распознаванием объектов. В то же время метод оптимизирует цели восприятия для надежного определения ключевых точек для замыкания цикла путем выборки траектории для максимального тестирования точек интереса. Классические схемы управления не способны предсказать затраты и соответствующую траекторию (например, ПИД-регуляторы). Кроме того, формулировка совокупности восприятия и действия как задачи оптимизации имеет преимущества в надежности системы управления. Предложенный подход можно сравнить с методом, основанным на методе с моделью предсказания с глубокими сверточными нейронными сетями для понимания сцен в реальном времени. Полностью сверточная сеть способна изучать сложное внеплоскостное преобразование, необходимое для проецирования пикселей наблюдаемого изображения на плоскость земли и прогнозирования карты контролируемых объектов в области перед БТС. Решение состоит в том, чтобы обучить глубокую нейронную сеть преобразовывать визуальные данные с одной монокулярной камеры в стоимость функции в локальной системе координат, ориентированной на робота, превосходны. Затем эта карта затрат может быть непосредственно введена в алгоритм управления прогностической моделью. Однако недостатком метода могут быть не посещаемые районы, где обучающая выборка может не иметь достаточно надежных карт затрат. Метод с моделью предсказания использует только ту информацию, которую она может видеть на выходе нейронной сети, и заранее планирует в течение 1,5 секунд выдать управляющий сигнал. Этот 1,5-секундный временной горизонт приводит к чрезвычайно робкому поведению, потому что доступный вид вперед имеет небольшое расстояние. Предложенный метод ASLAM-MPPI использует картографическую информацию, основанную на всем горизонте событий, который может быть скорректирован с поступлением новой информации. Таким образом, он более устойчив к потере информации, чем MPPI с глубокими сверточными нейронными сетями. Кроме того, значительным вкладом нашей работы на самом деле является разработка технологии метода активного SLAM для предотвращения сбоев в системе локализации. Значительным результатом является решение практической задачи реактивного планирования БТС на сложном полигоне, имитирующем производственную среду. Для этого глобальная траектория проходила через горку со сложными поворотами, по которой она не преодолевалась в ручном режиме БТС-управления без обучения оператора. MPPI с глубокими сверточными нейронными сетями не

может распознавать подъемы и спуски с горки без дополнительной информации. Таким образом, карта затрат, созданная этим методом, не может быть затем непосредственно введена в алгоритм управления прогностической моделью. Классические схемы управления (например, PID, LQR, iLQR) могут надежно отслеживать глобальную траекторию, но не являются стабильными в случае сбоев в системе локализации.

Благодарности. Автор выражает благодарность своему научному консультанту Д.Ю. Карамзину за постановку задачу, помощь и поддержку в исследованиях, своим соавторам В.А. Березневу, И.В. Прокопьеву и В.Н. Белотелову за актуальные постановки задач, а также своим коллегам – коллективу сотрудников отдела 55 Отделения 5 ФИЦ ИУ РАН, которые принимали участие в представленных в диссертации исследованиях.

Литература

[1] *Эльсгольц Л.Э.* Вариационное исчисление. – М.: Гостехиздат, 1958. – 163 с.

[2] *Гельфанд И.М., Фомин С.В.* Вариационное исчисление. – М.-Л.: Физматгиз, 1961. – 228 с.

[3] *Петров Ю. П.* Вариационные методы теории оптимального управления. Изд. 2-е. – Л.: «Энергия», 1977. – 280 с.

[4] *Блисс Г.А.* Лекции по вариационному исчислению / Пер. с англ. Ю.К. Солнцева. Под ред. Л. Э. Эльсгольца. – Москва: Изд-во иностр. лит., 1950. – 348 с.

[5] *Янг Л.* Лекции по вариационному исчислению и теории оптимального управления. – М.: Мир, 1974. – 488 с.

[6] *Морозов С.Ф.* Разрывные задачи вариационного исчисления: Учеб. пособие / Нижегород. гос. ун-т им. Н. И. Лобачевского. – Н. Новгород: ННГУ, 1991. – 79 с.

[7] *Кузнецов Ю.А., Семенов А.В.* Избранные главы вариационного исчисления. Электронное учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2012. – 69 с.

[8] *Mordukhovich B.S.* Variational Analysis and Generalized Differentiation. V. I. Basic Theory. Springer-Verlag. 2006.

[9] *Тихомиров В.М.* Принцип Лагранжа и задачи оптимального управления. Изд-во МГУ, 1982.

[10] *Понтрягин Л.С., Болтянский В.Г., Гамкрелидзе Р.В., Мищенко Е.Ф.* Математическая теория оптимальных процессов. – М.: Наука, 1961.

[11] *Hestenes M.R.* Calculus of Variations and Optimal Control Theory. New York: Wiley, 1966.

[12] *Liberzon D.* Calculus of Variations and Optimal Control Theory: A Concise Introduction. Princeton University Press. 2012.

[13] *Гамкрелидзе Р.В.* Оптимальные процессы управления при ограниченных фазовых координатах // Известия АН СССР серия математическая. 1960. Т. 24. С. 315–356.

[14] *Дубовицкий А.Я., Милютин А.А.* Задачи на экстремум при наличии ограничений //Журнал вычислительной математики и математической физики. 1965. Т.5, No.3. С. 395–453.

[15] *Арутюнов А.В., Тынянский Н.Т.* О принципе максимума в задаче с фазовыми ограничениями. // Изв. АН СССР. Сер. техн. Кибернетика. – 1984. – № 4. – С. 60–68.

[16] *Понтрягин Л.С.* Принцип максимума в оптимальном управлении. – М.: Наука, 1989.

[17] *Алексеев В. М., Тихомиров В. М., Фомин С. В.* Оптимальное управление. М.: Наука. 1979. – 430 с.

[18] *Болтянский В.Г.* Математические методы оптимального управления. 2-е изд., перераб. и доп. – Москва: Наука, 1969. – 408 с.

[19] *Габасов Р., Кириллова Ф.М.* Особые оптимальные управления. М.: Пленум Пресс, 1978.

[20] *Горнов А. Ю.* Алгоритмы решения задач оптимального управления с терминальными ограничениями // ЖВТ. 2008. №4. С. 24–30.

[21] *Kushlyk O.I.* Optimal control problem with phase constraints for a linear system of equations with delay. Ukr Math J. 1991. V. 43. C. 1533–1538.

[22] *Longla M.* Pontryagin's principle of maximum for linear optimal control problems with phase constraints in infinite dimensional spaces // Discrete and Continuous Models and Applied Computational Science. 2008. №4.

[23] *Tyatyushkin A., Zarodnyuk T.* Numerical method for solving optimal control problems with phase constraints // Numerical Algebra, Control and Optimization. 2017. V. No. 4. P. 481–492.

[24] *Тятюшкин А.И.* Оптимизация управления и параметров в системах с фазовыми ограничениями // Журнал вычислительной математики и математической физики. - 2023. - Т. 63. - №6. - С. 950–961.

[25] *Theodore T.-M.* Optimal control of a two-phase flow model with state constraints // Mathematical Control and Related Fields, 2016, V. 6. No. 2. P. 335–362.

[26] *Makowski K., Neustadt L.W.* Optimal Control Problems with Mixed Control-Phase Variable Equality and Inequality Constraints // SIAM Journal on Control. 1974. V. 12. No. 2. P. 184–228.

[27] *Bonnans J. F., Hermant A.* Revisiting the analysis of optimal control problems with several state constraints // Control and Cybernetics. 2009. V. 38. No. 4. P. 1021–1052.

[28] *Bortakovskii A.S.* Elimination of Active Phase Constraints in Optimal Control Problems // J. Comput. Syst. Sci. Int. 2024. V. 63. P. 743–764.

[29] *Бортаковский А.С.* Исключение активных фазовых ограничений в задачах оптимального управления // Известия Российской академии наук. Теория и системы управления. - 2024. - №5. - С. 68-89.

[30] *Athans M., Falb P.L.* Optimal Control: An Introduction to the Theory and Its Applications. – McGraw Hill, New York, 1966. – 879 p.

[31] *А. Ю. Горнов, А. И. Тятюшкин, Е. А. Финкельштейн* Численные методы для решения терминальных задач оптимального управления // Ж. вычисл. матем. и матем. физ. 2016. Т. 56. № 2. С. 224–237.

[32] *Киселев Ю. Н., Орлов С. М., Орлов М. В.* Исследование одной нелинейной задачи оптимального управления с особыми режимами // Сборник научных трудов ф-та ВМК МГУ имени М.В. Ломоносова «Проблемы динамического управления» под редакцией Ю.С. Осипова, А.В. Кряжимского. Т. 5. МАКС Пресс Москва, 2010. С. 113–126.

[33] *Дарьина А.Н., Дивеев А.И., Карамзин Д.Ю., Перейра Ф.Л., Софронова Е.А., Чертовских Р.А.* Исследование метода возмущений для решения нерегулярных задач оптимального управления с фазовыми ограничениями // Вопросы теории безопасности и устойчивости систем. 2020. № 22. С. 25-51.

[34] *Дарьина А.Н., Дивеев А.И., Карамзин Д.Ю., Перейра Ф.Л., Софронова Е.А., Чертовских Р.А.* Регулярные возмущения оптимального движения трехколесного мобильного робота с передним приводом при ограниченных фазовых переменных // Вопросы теории безопасности и устойчивости систем. 2020. № 22. С. 52-77.

[35] *Матвийчук А.Р., Малев А.Г., Зимовец А.А.* Численные методы решения некоторых задач управления с фазовыми ограничениями // Вестник ННГУ. 2011. №4-2.

[36] *Карамзин Д.Ю.* Некоторые оценки для скачка производной функции-множителя Лагранжа в задачах оптимального управления с фазовыми ограничениями второго порядка // Известия Иркутского государственного университета. Серия Математика. 2024. Т. 49. С. 3–15.

[37] *Mordukhovich B.Sh.* Maximum principle in the problem of time optimal response with nonsmooth constraints // Journal of Applied Mathematics and Mechanics. 1976. Vol. 40, No 6. P. 960–969.

[38] *Hartl R. F., Sethi S. P., Vickson R. G.* A Survey of the Maximum Principles for Optimal Control Problems with State Constraints. SIAM Review. 1995. Vol. 37, No 2. P. 181–218.

[39] *Hermant A.* Stability Analysis of Optimal Control Problems with a Second-Order State Constraint // SIAM Journal on Optimization. 2009. Vol. 20. P. 104–129.

[40] *Chertovskih R., Daryina A., Diveev A., Karamzin D. Pereira F.L., Sofronova E.* Investigation of a perturbation method to solve essentially non-regular time-optimal control problems with state constraints // 19th European Control Conference (ECC 2020). 2020. Art. no. 9143703/ P. 849–854.

[41] *Chertovskih R., Daryina A., Diveev A., Karamzin D., Pereira F.L., Sofronova E.* Regular perturbations to the motion of a three-wheeled mobile robot with the front-wheel drive under restricted state variables // 19th European Control Conference (ECC 2020). 2020. Art. no. 9143809. P. 1210–1215.

[42] *Neustadt L.W.* An abstract variational theory with applications to a broad class of optimization problems. II: Applications // SIAM J. Control. 1967. Vol. 5. No. 3. P. 505–527.

[43] *Warga J.* Minimizing variational curves restricted to a preassigned set //Trans. Amer. Math. Soc.1964. No. 112.

[44] *Новиков Д.А.* О простейшей задаче быстродействия с фазовым ограничением при управлении пространственной ориентацией тела // Тр. Ин-та математики и механики УрО РАН. 2023. Т. 29, № 3. С. 62–72.

[45] *Ioffe A. and Tikhomirov V.* Theory of extremal problems, North-Holland Publishing Co., Amsterdam-New York, (1979)

[46] *Федоренко Р.П.* Приближенное решение задач оптимального управления – М.: Наука, 1978. – 488 с.

[47] *Красовский Н.Н.* Теория управления движением. М.: Наука, 1968.

[48] *Филиппов А.Ф.* О некоторых вопросах теории оптимального регулирования // Вестник МГУ, Матем. и мех. 1959. № 2. С. 25–38.

[49] *Арутюнов А.В.* К теории принципа максимума в задачах оптимального управления с фазовыми ограничениями. // Докл. АН СССР. – 1989. – Т. 304. – № 1. – С. 11-14.

[50] *Arutyunov A.V.* Optimality conditions: Abnormal and Degenerate Problems. Mathematics and Its Application. Kluwer Academic Publisher. 2000.

[51] *Арутюнов А.В., Карамзин Д.Ю.* Принцип максимума в задаче оптимального управления с фазовыми ограничениями типа равенств // Дифференциальные уравнения. 2015. Т. 51. №1. С. 34–46.

[52] *Arutyunov A., Karamzin D., Pereira F.L.* Optimal Impulsive Control. The Extension Approach. / Lecture Notes in Control and Information Sciences. – Springer, Cham, 2019. – 477 p.

[53] *Arutyunov A.V., Aseev S.M.* State constraints in optimal control. The degeneracy phenomenon, Systems & Control Letters. 1995. V. 26. No. 4. P. 267–273.

[54] *Дубовицкий А.Я., Дубовицкий В.А.* Необходимые условия сильного минимума в задачах оптимального управления с вырождением концевых и фазовых ограничений // Успехи мат. наук. 1985. V. 40, No. 2(242). С. 175—176.

[55] *Karamzin D., Pereira F.L.* On a Few Questions Regarding the Study of State-Constrained Problems in Optimal Control // Journal of Optimization Theory and Applications. 2019. Vol. 180. No 1. P. 235–255.

[56] *Karamzin D., Pereira F.L.* On higher-order state constraints // SIAM Journal on Control and Optimization. 2023. Vol. 61, No 4. P. 1913–1933.

[57] *Арутюнов А.В., Карамзин Д.Ю.* Свойства экстремалей в задачах оптимального управления с фазовыми ограничениями. Дифференциальные уравнения, 2016, Т. 52, № 11, с. 1464–1476.

[58] *Arutyunov A.V., Karamzin D.Yu.* On Some Continuity Properties of the Measure Lagrange Multiplier from the Maximum Principle for State Constrained Problems // SIAM Journal on Control and Optimization. 2015. Vol. 53. No 4. P. 2514–2540.

[59] *Захаров Е.В., Карамзин Д.Ю.*. К исследованию условий непрерывности меры-множителя Лагранжа в задачах с фазовыми ограничениями // Дифференциальные уравнения. 2015. Т. 51. N 3, с. 395–401.

[60] *Куржанский А.Б.* Управление и наблюдение в условиях неопределенности. М.: Наука, 1977.

[61] *Матвеев А. С.* Задачи оптимального управления с запаздываниями общего вида и фазовыми ограничениями // Изв. АН СССР. Сер. матем. 1988. Т. 52. № 6. С. 1200–1229.

[62] *Дубовицкий А. Я., Милютин А. А.* Необходимые условия слабого экстремума в задачах оптимального управления со смешанными ограничениями типа неравенства // Ж. вычисл. матем. и матем. физ. 1968. Т.8. №4. С. 725—779.

[63] *Vinter R.B., Ferreira M.M.A.* When is the maximum principle for state constrained problems nondegenerate? // J. Math. Anal. and Appl. 1994. V. 187. P. 438—467.

[64] *Ferreira M.M.A., Fontes F.A.C.C., Vinter R.B.* Non-degenerate necessary conditions for nonconvex optimal control problems with state constraints // J. Math. Anal. and Appl. 1999. Vol. 233.

[65] *Chertovskih R., Karamzin D., Khalil N.T., Pereira F.L.* An indirect numerical method for a time-optimal state-constrained control problem in a steady two-dimensional fluid flow // IEEE/OES Autonomous Underwater Vehicle Workshop, Proceedings. 2018. Art. no. 8729750.

[66] *Chertovskih R., Karamzin D., Khalil N.T., Pereira F.L.* Path-constrained trajectory time-optimization in a three-dimensional steady flow field // 18th European Control Conference (ECC 2019). 2019. Art. no. 8796083. P. 3746–3751.

[67] *Chertovskih R., Karamzin D., Khalil N.T., Pereira F.L.* Regular path-constrained time-optimal control problems in three-dimensional flow fields // European Journal of Control. 2020. 56 p. 98-106.

[68] *Chertovskih R., Karamzin D., Khalil N.T., Pereira F.L.* An Indirect Method for Regular State-Constrained Optimal Control Problems in Flow Fields. IEEE Transactions on Automatic Control, 2021, 66(2), pp. 787–793.

[69] *Arutyunov A.V., Karamzin D.Yu., Pereira F.L.*: Optimal Impulsive Control. The Extension Approach. Springer (2011)

[70] *Arutyunov A.V., Karamzin D.Yu., Pereira F.L.* The Maximum Principle for Optimal Control Problems with State Constraints by R.V. Gamkrelidze: Revisited // J. Optim. Theory Appl. 2011. Vol. 149. No 3. P. 474–493.

[71] *Arutyunov A.V., Karamzin D.Y., Pereira F.L.* Investigation of Controllability and Regularity Conditions for State Constrained Problems // IFAC-PapersOnLine. 2017. Vol. 50. No 1. P. 6295–6302.

[72] *Vinter R.* Optimal control, Springer Science & Business Media, (2000).

[73] *Clarke F.H., Vinter R.B.* Optimal multiprocesses // SIAM J. Control and Optim. 1989. Vol. 27. No 5. P. 1072–1091.

[74] *Fontes F.A.C.C., Frankowska H.* Normality and Nondegeneracy for Optimal Control Problems with State Constraints // J. Optim. Theory Appl. 2015. Vol. 166. P. 115–136.

[75] *Arutyunov A.V.* Perturbations of extremal problems with constraints and necessary optimality conditions // Journal of Soviet Mathematics. 1991. V. 54, No. 6. P. 1342—1400.

[76] *Arutyunov A.V., Aseev S.M.* Investigation of the degeneracy phenomenon of the maximum principle for optimal control problems with state constraints // SIAM J. Control Optim. 1997. V. 35, No. 3. P. 930–952.

[77] *Arutyunov A., Karamzin D.* A Survey on Regularity Conditions for State-Constrained Optimal Control Problems and the Non-degenerate Maximum Principle // Journal of Optimization Theory and Applications. 2020. 184(3). P. 697-723.

[78] *Матвеев А.С.* О необходимых условиях экстремума в задаче оптимального управления с фазовыми ограничениями // Дифференц. уравнения. 1987. Т. 23, No. 4. С. 629-–640.

[79] *Афанасьев А.П., Дикусар В.В., Милютин А.А., Чуканов С.В.* Необходимое условие в оптимальном управлении. – М.: Наука, 1990.

[80] *Galbraith G.N., Vinter R.B.* Lipschitz continuity of optimal controls for state constrained problems // SIAM J. Control and Optim. 2003. Vol. 42, No. 5.

[81] *Арутюнов А.В.* Свойства множителей Лагранжа в принципе максимума Понтрягина для задач оптимального управления с фазовыми ограничениями // Дифференциальные уравнения. 2012. Т. 48. №11.

[82] *Hager W.W.* Lipschitz continuity for constrained processes // SIAM J. Control and Optim. 1979. V. 17, No 3. P. 321–338.

[83] *Maurer H.* Differential stability in optimal control problems // Applied Mathematics and Optimization. 1979. Vol. 5, No 1. P. 283–295.

[84] *Fabien B. C.* Numerical solution of constrained optimal control problems with parameters // Applied Mathematics and Computation. 1996. Vol. 80. No 1. P. 43–62.

[85] *Bryson E.R., Yu-Chi Ho.* Applied Optimal Control. – Teylor&Francis, 1969.

[86] *Васильев Ф.П.* Методы оптимизации. – М.: Факториал пресс, 2002.

[87] *Betts J. T.* Practical Methods for Optimal Control Using Nonlinear Programming. SIAM. 2001

[88] *Betts J.T., Huffman W.P.* Path-constrained trajectory optimization using sparse sequential quadratic programming // Journal of Guidance, Control, Dynamics. 1993. Vol. 16. No. 1. P. 59–68.

[89] *Büskens, C. and Maurer, H.* SQP-methods for solving optimal control problems with control and state constraints: adjoint variables, sensitivity analysis and real-time control, Journal of computational and applied mathematics, 120(1-2), 85–108, (2000).

[90] *Haberkorn T., Trélat E.* Convergence results for smooth regularizations of hybrid nonlinear optimal control problems // SIAM Journal on Control and Optimization. 2011. Vol. 49. No 4. P.1498–1522.

[91] *Jacobson D., Lele M.* A transformation technique for optimal control problems with a state variable inequality constraint // IEEE Transactions on Automatic Control/ 1969. Vol. 14. No 5. P. 457–464.

[92] *van Keulen T., Gillot J., de Jager B., Steinbuch M.* Solution for state constrained optimal control problems applied to power split control for hybrid vehicles // Automatica. 2014. Vol. 50. No 1. P. 187–192.

[93] *Pytlak R.* Numerical Methods For Optimal Control Problems With State Constraints. – Springer, 2006.

[94] *Betts J.T., Campbell S.L., Engelsone A.* Direct transcription solution of optimal control problems with higher order state constraints: theory vs practice // Optim Eng. 2007. No 8. P. 1–19.

[95] *Malanowski K., Maurer H.* Sensitivity Analysis for Optimal Control Problems Subject to Higher Order State Constraints // Annals Oper. Res. 2001. Vol. 101. P. 43–73.

[96] *Dang T.P., Diveev A.I., Sofronova E.A.* A Problem of Identification Control Synthesis for Mobile Robot by the Network Operator Method // Proceedings of the 11th IEEE Conference on Industrial Electronics and Applications (ICIEA). 2016. P. 2413–2418.

[97] *Diehl M., Bock H. G., Diedam H., Wieber P. B.* Fast direct multiple shooting algorithms for optimal robot control. In Fast motions in biomechanics and robotics. – Berlin, Heidelberg: Springer, 2006. (P. 65-93).

[98] *Soltani-Zarrin R., Zeiaee A., Jayasuriya S.* Pointwise Angle Minimization: A Method for Guiding Wheeled Robots Based on Constrained Directions // In ASME Dynamic Systems and Control Conference. American Society of Mechanical Engineers. 2014. P. V003T48A004-V003T48A004.

[99] *Zeiaee A., Soltani-Zarrin R., Fontes F.A.C.C., Langari R.* Constrained directions method for stabilization of mobile robots with input and state constraints // Proceedings of the American Control Conference. 2017. P. 3706–3711.

[100] *Поляк Б.Т.* Градиентные методы минимизации функционалов // Журнал вычислительной математики и математической физики. – 1963. – Т. 3. – №4. – С. 643-653.

[101] *Nesterov Y.* Gradient methods for minimizing composite functions // Mathematical Programming. – 2013. – Volume 140. – pp. 125-161.

[102] *Гилл Ф., Мюррей У., Райт М.* Практическая оптимизация. / Перевод с англ. В.Ю. Лебедева. – Москва: Мир, 1985. – 509 с.

[103] *Стронгин Р.Г.* Численные методы в многоэкстремальных задачах:(Информационно-статистические алгоритмы). – М.: Наука. Гл. ред. физ.-мат. лит., 1978. – 240 с.

[104] *Horst R., Tuy H.* Global optimization: Deterministic approaches. 3rd revised and enlarged ed. – Springer, 1996. – 748 с.

[105] *Граничин О. Н., Поляк Б.Т.* Рандомизированные алгоритмы оптимизации и оценивания при почти произвольных помехах. – М.: Наука, 2003. – 291 с.

[106] *Evtushenko Yu.G., Posypkin M.A.* A deterministic algorithm for global multiobjective optimization. // Optimization Methods and Software. – 2014. – 29:5. – pp. 1005-1019.

[107] *Евтушенко Ю.Г.* Численный метод поиска глобального экстремума функций (перебор на неравномерной сетке) // Ж. вычисл. матем. и матем. физ. – 1971. – 11:6. – С. 1390-1403.

[108] *Карпенко А.П.* Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой. – М.: Издательство МГТУ им. Н.Э. Баумана, 2014. – 448 с. 257

[109] *Glover F., Kochenberger G. A. (eds.)* Handbook of metaheuristics. / International Series in Operations Research & Management Science. Vol 57. – Springer, Boston, MA, 2003. – 557 с.

[110] *Kennedy J., Eberhart R.* Particle swarm optimization // Proceedings of IEEE International conference on Neural Networks. 1995, pp. 1942–1948.

[111] *Shi C., Bu Y., Liu J.* Mobile robot path planning in three-dimensional environment based on ACO-PSO hybrid algorithm. IEEE/ASME intern. conf. on advanced intelligent Mathematics and Mathematical Modeling 57 mechatronics: AIM 2008 (Xian, China, July 2-5, 2008): Proc. N.Y.: IEEE, 2008.

[112] *Eberhart R., Kennedy J.* A new optimizer using particle swarm theory. 6th intern. symp. on micromachine and human science: MHS'95 (Nagoya, Japan, Oct. 4-6, 1995): Proc. N.Y.: IEEE, 1995. Pp. 39–43.

[113] *Daryina A.N., Prokopiev I.V.* An optimization method of the unmanned vehicle controller's parameters based on the optimization of particles's roy // Procedia Computer Science. 2021, 787-792.

[114] *Прокопьев И.В., Софронова Е.А.* Экспериментальное исследование методов синтеза управления в реальном времени движением беспилотного транспортного средства // Фундаментально-прикладные проблемы безопасности, живучести, надежности, устойчивости и эффективности систем. Материалы III международной научно-практической конференции, посвящённой 110-летию со дня рождения академика Н.А. Пилюгина, Елец, июнь 2019. С. 376-384.

[115] *Michalewicz Z., Krawczyk J., Kazemi M., Janikow C.Z.* Genetic algorithms and optimal control problems // Proceedings of the IEEE Conference on Decision and Control. 1991. 3. P. 1664–1666.

[116] *Дивеев А.И., Константинов С.В.* Исследование практической сходимости эволюционных алгоритмов оптимального программного управления колесным роботом // Известия РАН: Теория и системы управления. 2018. №4. С. 80-106.

[117] *Bertsekas D.P.* Nonlinear Programming. 2nd edition. – Athena Scientific, Belmont, MA, 1999. – 791 с.

[118] *Карманов В.Г.* Математическое программирование. м.: Физматлит, 2000.

[119] *Базара, Шетти* Нелинейное программирование. Теория и алгоритмы. / М. Базара, К. Шетти; пер. с англ. Т.Д. Березневой, В.А. Березнева. – Москва: Мир, 1982. – 583 с.

[120] F.H. Clarke. Optimization and Nonsmooth Analysis, John Wiley and Sons, New York. 1983.

[121] *Dijkstra E.W.* A note on two problems in connection with graphs // Numer. Math. Springer Science + Business media. 1959. V 1. N 1. P. 269-271.

[122] *Hart P.E., Nilsson N.J., Raphael, B.* A formal basis for the heuristic determination of minimum cost paths // IEEE Transactions on Systems Science and Cybernetics. 1968. Vol. 4(2) P. 100-107.

[123] *Stentz A.* (1995). The focussed D* algorithm for real-time replanning // Proceedings of the International Joint Conference on Artificial Intelligence. 1995. P. 1652-1659.

[124] *Дивеев А.И.* Метод сетевого оператора. М.: ВЦ РАН, 2010, 178 с.

[125] *Diveev A., Sofronova E.* The Network Operator Method for Search of the Most Suitable Mathematical Equation, in: Bio-Inspired Computational Algorithms and Their Applications, Edited by Dr. Shangce Gao, InTech, 2012. pp. 19-42.

[126] *Дивеев А.И., Мендес Флорес Н.Х.* Синтез системы пространственной стабилизации мобильного робота на основе обучения методом символьной регрессии // Вестник Российского университета дружбы народов. Серия: инженерные исследования. 2021. Т. 22. №2. С. 129-138.

[127] *Дарьина А.Н., Прокопьев И.В.* Метод оптимизации параметров контроллера беспилотного транспортногосредства на основе оптимизации роя частиц // Надежность и качество сложных систем. 2020. № 3 (31). С. 80-87.

[128] *Puterman M. L.* Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons. 1994.

[129] *Sutton R. S., Barto A. G.* Reinforcement Learning: An Introduction. MIT Press. 2018.

[130] *Karaman S., Frazzoli E.* Sampling-based algorithms for optimal motion planning //The International Journal of Robotics Research. 2011. Vol. 30(7). P. 846–894.

[131] *LaValle S.M.* Rapidly-exploring random trees: A new tool for path planning. Technical Report TR98-11. 1998. Department of Computer Science, Iowa State University.

[132] *Kavraki L.E., Svestka P., Latombe J. C., Overmars M. H.* Probabilistic roadmaps for path planning in high-dimensional configuration spaces // IEEE Transactions on Robotics and Automation. 1996. Vol. 12(4). P. 566–580.

[133] *Алексеев С.А., Калиниченко А.И., Клебан В.О., Шалыто А.А.* Автоматический синтез системы управления мобильным роботом для решения задачи «кегельринг» // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2011. Т. 72. № 2. С. 26–31.

[134] *Atanasov N., et al.* Kinematic Modeling and Calibration of Autonomous Ground Vehicles // IEEE Transactions on Robotics. 2014.

[135] *Хоанг Д.Т., Пыркин А.А.* Алгоритм траекторного управления движением мобильного робота без измерения координат положения // Научно-технический вестник информационных технологий, механики и оптики. 2021. Т. 21. № 6. С. 858–865.

[136] *Юхимец Д.А., Губанков А.С., Зуев А.В.* Метод формирования пространственных траекторий мобильного робота в неизвестной обстановке // Робототехника и техническая кибернетика. 2018. Т. 19. №2. С. 46–51.

[137] *Borenstein J., Koren Y.* Real-time obstacle avoidance for fast mobile robots. IEEE Trans. on Systems, Man, and Cybernetics, 1989, vol. 19, no. 5, pp. 1179–1187.

[138] *Garrido-Jurado S., Muñoz-Salinas R., Madrid-Cuevas F. J., Marín-Jiménez M. J.* Automatic generation and detection of highly reliable fiducial markers under occlusion // Pattern Recognition. 2014. V. 47. P. 2280-2292.

[139] *Mario D. Fiore, Felix Allmendinger, Ciro Natale* A general constraint-based programming framework for multi-robot applications // Robotics and Computer-Integrated Manufacturing. 2024. V. 86.

[140] *Дарьина А.Н., Прокопьев И.В.* Синтез живучих робототехнических систем управления с переменной структурой // Наукоемкие технологии. 2016. Т. 17, No. 6. С. 16–20.

[141] *Дергачёв С. А., Яковлев К. С.* Применение управления с прогнозирующими моделями и стохастической оптимизацией в задаче децентрализованного много-агентного избегания столкновений // XIV Всероссийское совещание

по проблемам управления (ВСПУ 2024) (Москва, ИПУ РАН, 17-20 июня 2024 г.). С. 1792–1796.

[142] *Дарьина А. Н., Прокопьев И.В.* Синтез функционала для оптимального управления мобильным роботом в реальном времени // Материалы IV международной научно-практической конференции «Фундаментально-прикладные проблемы безопасности, живучести, надежности, устойчивости и эффективности систем». Елец, Россия. 2020

[143] *Прокопьев И.В.* Модель интеллектного контроллера подстройки эталонной модели для системы координатно-параметрического управления // Труды ИСА РАН. Динамика неоднородных систем. 2010. 5 (53). С. 80-88.

[144] *Kschischang F. R., Frey B. J. and Loeliger H. A.* Factor Graphs and the Sum-Product Algorithm // IEEE Transactions on Information Theory, 2001, 47(2), 498–519.

[145] *Macenski S., et al.* ROS 2 Navigation Stack. 2020.

[146] *Magni L., Nicolao G., Magnani L., Scattolini R.* A Stabilizing Model-Based Predictive Control Algorithm for Nonlinear Systems // Automatica. 2001. V. 37. P. 1351–1362.

[147] *Rosenblatt J. K.* Optimal selection of uncertain actions by maximizing expected utility. Proc. 1999 IEEE Int. Symposium on Computational Intelligence in Robotics and Automation, November 1999, 95-100.

[148] *Weise T.* Global Optimization Algorithms - Theory and Application: Ph.D The-sis. -University of Kassel, 2008.

[149] *Williams G., Wagener N., Goldfain B., Drews P., Rehg J.M., Boots B., and Theodorou E.A.* Information Theoretic MPC for Model-Based Reinforcement Learning Conference: 2017 IEEE Interna-tional Conference on Robotics and Automation (ICRA) 29 May-3 June 2017

[150] *Сидоренко А. В.* Обучение с подкреплением при навигации мобильных роботов // Вестник Полоцкого государственного университета. Серия С. Фундаментальные науки. 2021. №12.

[151] *Кузнецов А.А.* Обзор возможности обучения роботов при помощи алгоритмов глубокого обучения // Вестник науки. 2022. Т. 51. №6 С. 239–243.

[152] *Goodfellow I., Bengio Y., Courville A.* Deep Learning. MIT Press. 2016

[153] *Лю В.* Методы планирования пути в среде с препятствиями (обзор) // Математика и математическое моделирование. 2018. № 01. С. 15–58.

[154] *Нимай Чандра Дас, Скакун А.Д.* Алгоритм планирования траектории мобильного робота. Электронный ресурс. `https://scm.etu.ru/assets/files/2021/scm21/papers/228-231.pdf` (дата последнего обращения 30.05.2025 г.)

[155] *Яковлев К. С., Андрейчук А. А., Скрынник А. А., Панов А. И.* Методы планирования и обучения в задачах многоагентной навигации // Доклады Российской академии наук. Математика, информатика, процессы управления. Т. 508, № 1, 2022. с. 88–93.

[156] *Ge S.S., Cui Y.J.* New potential functions for mobile robot path planning // IEEE Trans. on Robotics and Automation. 2000. Vol. 16. No. 5. P. 615–620.

[157] *Lumelsky V., Stepanov A.* Dynamic path planning for a mobile automaton with limited information on the environment. IEEE Trans. on Automatic Control, 1986, vol. 31, no. 11, pp. 1058–1063.

[158] *Glasius R., Komoda A., Stan C.A.M. Gielen* Neural network dynamics for path planning and obstacle avoidance. Neural Networks, 1995, vol. 8, no. 1, pp. 125–133.

[159] *Kuffner J.J., LaValle S.M.* RRT-connect: An efficient approach to to single-query path planning. IEEE intern. conf. on robotics and automation: ICRA'2000 (San Francisco, CA, USA, April 24-28, 2000): Proc. N.Y.: IEEE, 2000. Vol. 2. P. 995–1001.

[160] *Бобцов А.А., Добриборщ Д., Капитонов А.А.* Система навигации и управления движением мобильным роботом // Научно-технический вестник информационных технологий, механики и оптики. 2017. Т. 17. № 2. С. 365–367.

[161] *Голован А.А., Гришин А.А., Жихарев С.Д., Ленский А.В.* Алгоритмы решения задачи навигации мобильных роботов. М.: Институт механики МГУ имени М.В. Ломоносова. Препринт №57-99. 1999. 54 с.

[162] *Дарьина А.Н., Прокопьев И.В.* Метод активной одновременной локализации и картографирования на основе модели прогнозирующего интегрального пути для мобильных роботов // Нейрокомпьютеры: разработка, применение. 2021. Т. 23. № 6. С. 12-23.

[163] *Лесникова А. А., Першина Ж. С.* Система локальной навигации мобильного робота на основе визуальной одометрии // Инженерные и научные приложения на базе технологий NI (NIDays-2014): сб. тр. 13 междунар. науч.-практ. конф., Москва, 19-20 нояб. 2014 г. - Москва : ДМК, 2014. С. 224–225.

[164] *Dellaert F., Kaess V.* Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. The International Journal of Robotics Research, 2006, 25(12), 1181-1203.

[165] *H. Durrant-Whyte and T. Bailey* "Simultaneous localization and mapping: part I,"in IEEE Robotics and Automation Magazine, vol. 13, no. 2, pp. 99-110, June 2006, doi: 10.1109/MRA.2006.1638022.

[166] *Guizilini V., Ramos F.* Towards real-time 3d continuous occupancy mapping using hilbert maps. The International Journal of Robotics Research, 2018, 37, 566-584.

[167] *Kaess M., Johannsson H., Roberts R., Ila V., Leonard J.J. and Dellaert F.* iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. The International Journal of Robotics Research, 2012, 31, 217-236.

[168] *Sani M. F., Karimian Gh.* Automatic Navigation and Landing of an Indoor AR.Drone Quadrotor Using ArUco Marker and Inertial Sensors // International Conference on Computer and Drone Applications (IConDA). 2017.

[169] *Sun K., Mohta K., Pfrommer B., Watterson M., Liu S., Mulgaonkar Y., Taylor C.J., and Kumar V.* Robust stereo visual inertial odometry for fast autonomous flight, IEEE Robotics and Automation Letters, 2018, vol. 3, no. 2, pp. 965–972.

[170] *Zhao S., Wang P., Zhang H., Fang Z., and Scherer S.* TP-TIO: A robust thermal-inertial odometry with deep thermalpoint, 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

[171] *Thrun S., Montemerlo M.* The GraphSLAM Algorithm With Applications to Large-Scale Mapping of Urban Structures. The International Journal of Robotics Research, 2005, 25(5-6), 403-430.

[172] *Marut A., Wojtowicz K., Falkowski K.* ArUco markers pose estimation in UAV landing aid system // In Proceedings of the 2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace). – Turin, Italy, 19-21 June 2019. P. 261–266.

[173] ROS Wik. Электронный ресурс.: `http://wiki.ros.org/aruco`. (дата последнего обращения 30.05.2023 г.)

[174] *Wong J.* Theory of Ground Vehicles, Wiley, (2008).

[175] *Field A., Cherchas D., and Calisal S.* Optimal control of an autonomous underwater vehicle, World Automatic Congress, (2000).

[176] *Rawlings J. B., Mayne D. Q., Diehl M.* (2017). Model Predictive Control: Theory, Computation, and Design.

[177] *LaValle S. M.* Planning Algorithms. Cambridge University Press. 2006.

[178] *Измаилов А.Ф., Солодов М.В.* Численные методы оптимизации. М.: Физматлит, 2003.

[179] *Mur-Artal R., Montiel J. M. M. and Tardos J. D.* Orb-slam: a ver-satile and accurate monocular slam system // IEEE Transactions on Robotics. 2015. 31(5):1147–1163.

[180] *Bloesch M., Omari S., Hutter M., and Siegwart R.* Robust visual inertial odometry using a direct EKF-based approach, 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 298–304.

[181] *Qin T., Li P., and Shen S.* VINS-Mono: A robust and versatile monocular visual-inertial state estimator, IEEE Transactions on Robotics, 2018, vol. 34, no. 4, pp. 1004–1020.

[182] *Khattak S., Papachristos C., and Alexis K.* Keyframe-based thermal-inertial odometry // Journal of Field Robotics, 2020, vol. 37, no. 4, pp. 552–579.

[183] *Saputra M.R.U., de Gusmao P.P.B., Lu C.X., Almalioglu Y., Rosa S., Chen C., Wahlstroem J., Wang W., Markham A., and Trigoni N.* DeepTIO: A deep thermal-inertial odometry with visual hallucination, IEEE Robotics and Automation Letters, 2020, vol. 5, no. 2.

[184] *Dickmann J., Klappstein J., Hahn M., Appenrodt N., Bloecher H., Werber K., and Sailer A.* Automotive radar – The key technology for autonomous driving: From detection and ranging to environmental understanding, 2016 IEEE Radar Conference (RadarConf).

[185] *Bereznev V.A.* The principle of dividing feasible trajectories in a robot control problem // Procedia Computer Science. 2021. P. 456-459.

[186] *Aswani A.* Provably safe and robust learning-based model predictive control // Automatica. 2013. Vol. 49. No. 5. P. 12161226.

[187] *Chakrabarty A.* Support Vector Machine Informed Explicit Nonlinear Model Predictive Control Using Low-Discrepancy Sequences / A. Chakrabarty et al. // IEEE Transactions on Automatic Control. 2017. Vol. 62, No. 1. P. 135-148.

[188] *Chen H.* A Quasi-Innite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability // Automatica. 1998. Vol. 34. No. 10. P. 1205-1217.

[189] *Domahidi A.* Learning a feasible and stabilizing explicit model predictive control law by robust optimization // Proceedings of the IEEE Conference on Decision & Control. 2011. No. EPFL-CONF-169723.

[190] *Fontes F.A.C.C.* A General Framework to Design Stabilizing Nonlinear Model Predictive Controllers // Systems & Control letters. 2000. Vol. 42. No. 2. P. 127-143.

[191] *Grune L.* Nonlinear model predictive control. Springer London. 2011

[192] *Hertneck M.* Learning an approximate model predictive controller with guarantees // IEEE Control Systems Letters. 2018. Vol. 2. No. 3. P. 543–548.

[193] *Pin G.* Approximate model predictive control laws for constrained nonlinear discrete-time systems: analysis and o ine design // International Journal of Control. 2013. Vol.86. No.5. P. 804-820.

[194] *Rawlings J.B., Mayne D.Q.* Model Predictive Control: Theory and Design. Madison: Nob Hill Publishing. 2009.

[195] *Rosolia U., Borrelli F.* Learning model predictive control for iterative tasks. a datadriven control framework // IEEE Transactions on Automatic Control. 2018. Vol. 63. No. 7.

[196] Электронный ресурс. `https://sites.google.com/view/mpc-robotics` (дата последнего обращения 25.01.2025).

[197] *Williams G., Aldrich A., and Theodorou E. A.* Model predictive path integral control: From theory to parallel computation // Journal of Guidance, Control, and Dynamics, pp. 1–14, 2017.

[198] *Williams G., Drews P., Goldfain B., Rehg J. M. and Theodorou E. A.* Aggressive driving with model predictive path integral control, in 2016 IEEE International Conference on Robotics and Automation (ICRA), May 2016, pp. 1433–1440.

[199] Justin Fu's MPPI implementation. Электронный ресурс. `https://github.com/justinjfu/mpPI` (дата последнего обращения 19.04.2025).

[200] *Ахметзянов И.З., Ионов М.А., Карабцев В.С.* Модификация алгоритма RRT для определения оптимальной траектории движения автомобиля при объезде препятствий // Вестник СибАДИ. 2017. Т. 58. № 6. С. 148-154.

[201] *Hsu D., Kindel R., Latombe J.-C., Rock S.* Randomized kinodynamic motion planning with moving obstacles // The International Journal of Robotics Research. 2002. V. 21. P. 233–255.

[202] *Kunz T., Stilman M.* Kinodynamic RRTs with fixed time step and best-input extension are not probabilistically complete // Algorithmic Foundations of Robotics XI. Springer. 2015. P. 233–244.

[203] *Алак Сабах Алхалили, Лукьянов Е.А.* Управление движением колесного мобильного робота на основе имитационного моделирования // Вестник БГТУ им. В.Г. Шухова. 2022. № 8. С. 112–121.

[204] *Bhatia S., Kaur G.* Multi-agent systems: A survey // Artificial Intelligence Review/ 2020. Vol. 53. No/ 2. P. 1–40.

[205] *Ogata K.* Modern Control Engineering. Prentice Hall. 2010.

[206] *Карпасюк И.В.* Модификация метода потенциалов для поиска путей на взвешенном графе // Технические средства систем управления и связи. Материалы конференции «Информационные технологии и технические средства управления» (ICCT-2021), 14-й Международной конференции «Акустооптические и радиолокационные методы измерений и обработки информации» (ARMIMP-2021). Астрахань. 2021. P. 248–250.

[207] *Liu V.* Methods of path planning in an environment with obstacles (review) // Mathematics and mathematical modeling: a network scientific publication. Moscow: MGTU. 2018. No. 01. P. 15–58.

[208] *Lim C. W., Yong L. S., Ang M.H.* Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment // Proceedings of the IEEE International Symposium on Intelligent Control (ISIC). Vancouver, Canada. 2002. P. 821–826.

[209] *Zhenyu W., Lin F.* Obstacle prediction–based dynamic path planning for a mobile robot // International Journal of Advancements in Computing Technology. 2012. Vol. 4. No. 3. P. 118–124.

[210] *А.Н. Дарьина.* Метод ньютоновского типа для задач оптимального управления // В сборнике: Фундаментально-прикладные проблемы безопасности, живучести, надежности, устойчивости и эффективности систем Материалы II международной научно-практической конференции, посвящённой 105-летию со дня рождения адмирала флота СССР дважды героя Советского Союза Сергея Георгиевича Горшкова. 2018. С. 237–242.

[211] *А.Н. Дарьина.* О задаче управления группой роботов // В сборнике: Фундаментально-прикладные проблемы безопасности, живучести, надежности, устойчивости и эффективности систем Материалы II международной научно-практической конференции, посвящённой 105-летию со дня рождения адмирала флота СССР дважды героя Советского Союза Сергея Георгиевича Горшкова. 2018. С. 242–246.

[212] *Jing R., McIsaac K.A., Patel R.V.* Modified Newton's method applied to potential field-based navigation for mobile robots // IEEE Transactions on Robotics. 2006. Vol. 22. No. 2. P. 384–391.

[213] *Shiltagh N., Jalal L.* Optimal path planning for intelligent mobile robot navigation using modified particle swarm optimization // International Journal of Engineering and Advanced Technology. 2013. Vol. 2. No. 4. P. 260–267.

[214] *Yee Z.C., Ponnambalam S.G.* Mobile robot path planning using ant colony optimization // IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Singapore, 2009. P. 851– 856.

[215] *Антонов В.О., Гурчинский М.М., Петренко В.И., Тебуева Ф.Б.* Численный метод планирования траектории обхода препятствий на основе итеративной кусочно-линейной аппроксимации // Системы управления, связи и безопасности. 2018. № 1. С. 168–182.

[216] *Medvedev M., Pshikhopov V.* Path planning of mobile robot group based on neural networks // Lecture Notes in Computer Science 2020. T. 12144LNAI. P. 51–62.

[217] *Moreno J.A., Castro M.* Heuristic algorithm for robot path planning based on a growing elastic net // Progress in artificial intelligence: 12th Portuguese conf. on artificial intelligence: EPIA 2005 (Covilhã, Portugal, December 5-8, 2005): Proc. B.: Springer, 2005. Vol. 3808. P. 447–454.

[218] *Pshikhopov V.Kh., Gurenko B.V., Medvedev M.Yu.* Algorithms of adaptive positiontrajectory control systems of moving objects // Control problems. 2015. No. 4. P. 66–75.

[219] *Drews P., Williams G., Goldfain B., Theodorou E.A., Rehg J.M.* Aggressive Deep Driving: Combining Convolutional Neural Networks and Model Predictive Control, 2017, arXiv:1707.05303.

[220] *D. Q. Mayne* Model predictive control: Recent developments and future promise // Automatica. 2014. V. 50, No. 12. P. 2967–2986.

[221] *Howard T. M., Kelly A.* Optimal rough terrain trajectory generation for wheeled mobile robots // International Journal of Robotics Research. 2007. Vol. 26. No. 2. P. 141–166.

[222] *Howard T. M., Kelly A.* Trajectory and spline generation for all-wheel steering mobile robots // Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems, Beijing, China. 2006.

[223] *Howard T.M., Green C., Ferguson D., Kelly A.* State space sampling of feasible motions for high performance mobile robot navigation in complex environments // Journal of Field Robotics. 2008. Vol. 25. No. 6-7. P. 325–345.

[224] *Howard T.M., Green C., Kelly A.* Receding horizon model-predictive control for mobile robot navigation of intricate paths // In Proceedings of the 7th International Conference on Field and Service Robotics. 2009.

[225] *Kelly A., Nagy B..* Reactive nonholonomic trajectory generation via parametric optimal control // International Journal of Robotics Research. 2003. Vol. 22. No 7-8. P. 583–601.

[226] *Kuwata Y., Teo J., Karaman S., Fiore G., Frazzoli E., How J. P.* Motion planning in complex environments using closed-loop prediction // in AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, HI. Art. num. AIAA–2008–7166. 2008.

[227] *Diveev A.I.* A Numerical Method for Network Operator for Synthesis of a Control System with Uncertain Initial Values // Journal of Computer and Systems Sciences International. 2012. Vol. 51, No. 2. P. 228-–243.

[228] *Diveev A.I., Sofronova E.A.* Application of network operator method for synthesis of optimal structure and parameters of automatic control system // Proceed-ings of 17-th IFAC World Congress, Seoul. 2008. P. 6106 — 6113.

[229] *Diveev A.I., Sofronova E.A.* Numerical method of network operator for multi-objective synthesis of optimal control system // Proceedings of Seventh International Conference on Control and Automation (ICCA'09) Christchurch, New Zea-land. 2009. P. 701–708.

[230] *Zhang J. and Singh S.* Loam: Lidar odometry and mapping in real-time. In Robotics: Science and Systems, volume 2, 2014.

[231] *J. Engel T. Schops and D. Cremers* Lsd-slam: Large-scale direct monocular slam. In ¨ European Conference on Computer Vision, pages 834–849. Springer, 2014.

[232] *Mur-Artal R., Tardos J. D.* ORB-SLAM2: an open-source SLAM system for monocular, stereo,and RGB-D cameras, IEEE Transactions on Robotics, 2017, 33 (5), 1255–1262.

[233] *Newcombe R. A., Izadi S., Hilliges O., Molyneaux D., Kim D., Davison A.J., Kohi P., Shotton J., Hodges S., and Fitzgibbon A.* Kinectfusion: Real-time dense surface mapping and tracking. In Mixed and augmented reality (IS-MAR), 2011 10th IEEE international symposium on. P. 127–136.

[234] *Kendall A., Cipolla R.* Modelling uncertainty in deep learning for camera relocalization. Proceedings of the International Conference on Robotics and Automation (ICRA), 2016.

[235] *Vasilev D. V., Latyshev A., Kuderov P., Shiman N., Panov A. I.* Dynamical Distance Adaptation in Goal-Conditioned Model-Based Reinforcement Learning // Advances in Neural Computation, Machine Learning, and Cognitive

Research VIII. Selected Papers from the XXVI International Conference on Neuroinformatics, October 21–25, 2024, Moscow, Russia. Pages 185–198.

[236] *Морозова В.И., Логунова Д.И.* Прогнозирование методом машинного обучения // Молодой ученый. 2022. Т. 416. № 21. С. 202–204.

[237] *Нестерова М., Скрынник А., Панов А.* Adaptive Curriculum Learning: Optimizing Reinforcement Learning Through Dynamic Task Sequencing // XXVI Международная научно-техническая конференция «Нейроинформатика-2024» (Россия, Москва, 21–25 октября 2024 г.).

[238] *Панов А. И.* Обучение с подкреплением: теоретические основы и алгоритмические реализации // Совместный семинар РАИИ и ФИЦ ИУ РАН «Проблемы искусственного интеллекта» (Москва, 30 октября 2024 г.).

[239] *Каллан Р.* Основные концепции нейронных сетей. Москва. Санкт-Петербург, Киев: Издательский дом «Вильямс». 2003. – 288 с.

[240] *Эйкхофф П.* Основы идентификации систем управления. Оценивание параметров и состоя-ния. М.: Мир. 1975. 687 с.

[241] *Чернухин Ю.В., Доленко Ю.С., Бутов П.А.* Бионические подходы к обработке сенсорной информации в нейросетевых системах управления интеллектуальных мобильных роботов // Известия ЮФУ. Технические науки. 2012. №5. С. 194–199.

[242] *Пякилля Б. И.* Идентификация математической модели робототехнической системы // МСиМ. 2014. №4 (32). С. 100–104.

[243] *Нильсон Н.* Искусственный интеллект. Методы поиска решений. М.: Издательство «Мир». 1973. 270 с.

[244] *Hart P. E., Nilsson N. J., Raphael B.* Correction to «A Formal Basis for the Heuristic Determination of Minimum Cost Paths». SIGART Newsletter, 1972, 37, 28–29.

[245] *Иоффе М. Л.* Принцип Аккермана и его реализация в современных автомобилях // Известия вузов. Машиностроение. 2021. №9 (738).

[246] *Лапин А.В., Зубов Н.Е., Пролетарский А.В.* Обобщение формулы Аккермана для некоторого класса многомерных динамических систем с векторным входом // Вестник МГТУ им. Н. Э. Баумана. Сер. Естественные науки. 2023. Т. 109. № 4.

[247] *Ljung L.* System Identification: Theory for the User. Prentice Hall. 1999.

[248] *Siciliano B., et al.* Robotics: Modelling, Planning and Control. Textbook. 2009.

[249] *Домбровский В.В.* Эконометрика: учебник. М.: Новый учебник. 2004. 342 с.

[250] *Турчак Л. И., Плотников П. В.* Основы численных методов: учебное пособие. М.: Физматлитю 2002.

[251] *Nelles O.* Nonlinear System Identification. Springer. 2001.

[252] *Slotine J.-J., Li W.* Applied Nonlinear Control. Prentice Hall. 1991. 459 p.

[253] *Thrun S., Burgard W., Fox D.* Probabilistic Robotics. MIT Press. 2005. 668 p.

[254] *Маккалох Дж., Питтс У.* Логические исчисления идей, относящихся к нервной деятельности.// Автоматы. М.: ИЛ, 1956.

[255] *Raissi M., Perdikaris P., Karniadakis G. E.* (2019). Physics-informed neural networks. Journal of Computational Physics.

[256] Обзор рекуррентных сетей и их модификаций для работы с последовательностями. Электронный ресурс. `education.yandex.ru/handbook/ml/article/nejroseti-dlya-raboty-s-posledovatelnostyami` (дата последнего обращения 30.04.2025 г.)

[257] Synthetic Trajectory Generation Through Convolutional Neural Networks: arxiv.org/abs/2407.16938

[258] Efficient Multi-Agent Trajectory Prediction with Adaptation (ADAPT): arxiv.org/abs/2307.14187

[259] Machine Learning for Autonomous Vehicle's Trajectory Prediction: arxiv.org/abs/2307.07527

[260] Пример интеграции нейросети с маршрутизатором для генерации траекторий. Электронный ресурс. `habr.com/ru/companies/otus/articles/904352/` (дата последнего обращения 09.02.2024 г.)

[261] *Bellman R.* Dynamic programming, 1957.

[262] *Beckov A.V., Prokopiev I.V.* Problem of Cost Function Synthesis for Mobile Robot's Trajectory and the Network Operator Method for its Solution // 13th International Symposium Intelligent Systems, St. Pe-tersburg, Russia. 2018. P. 695–701.

[263] *Дарьина А. Н., Прокопьев И.В.* Метод нейросетевого управления в реальном времени на основе синтеза функции выбора // Надежность и качество сложных систем. 2019. Т. 28. № 4. С. 41—50.

[264] *Дарьина А.Н., Дивеев А.И., Прокопьев И.В.* Робототехнический центр ФИЦ ИУ РАН // Вопросы теории безопасности и устойчивости систем, выпуск 21. М.: ФИЦ ИУ РАН. 2019. С. 66–77.

[265] *Дивеев А.И., Доценко А. В.* Библиотека Python для синтеза интеллектуальных систем управления // Вестник РУДН. Серия: Инженерные исследования. 2018. V. 19, No. 2. С. 177–189.

[266] *Todoran H.G., Bader M.* Extended Kalman Filter (EKF)-based Local SLAM in Dynamic Environments // Advances in Intelligent Systems and Computing. 2015

[267] *Montemerlo M., Thrun S., Koller D., Wegbreit B.* FastSLAM: A factored solution to the simultaneous localization and mapping problem // Proceedings of the AAAI National Conference on Artificial Intelligence, 2002, 593–598.

[268] *Bajcsy R.* Active Perception. Proceedings of the IEEE, 1988, 76(8), 966-1005.

[269] *Leung C., Huang S., Dissanayake G.* Active SLAM using Model Predictive Control and Attractor Based Exploration. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, 5026–5031.

[270] *Scaramuzza D., Fraundorfer F.* Visual Odometry [Tutorial]. Part I: The First 30 Years and Fun-damentals. IEEE Robotics and Automation Magazine, 2011, 18(4), 80-92.

[271] Электронный ресурс. `https://wiki.ros.org/navigation`. (дата последнего обращения 30.05.2025 г.)

Приложение А

Листинги программ к главам 1 и 2

А.1 Листинг программы для точного нахождения оптимального пути между двумя точками при наличии фазовых ограничений

```
1
2  import numpy as np
3  from scipy.integrate import solve_ivp
4  from scipy.optimize import root
5  import matplotlib.pyplot as plt
6
7  # Parameters
8  epsilon = 0.1
9  A = np.array([-1.5, 0.3])
10 B = np.array([1.5, -0.5])
11 alpha0 = -np.pi / 4
12 tol_boundary = 1e-3
13 max_refinements = 7  # Maximum of 7 step reductions
14
15 # Function a_eps(t)
16 def compute_a_eps(psi1, psi2, mu, x1, x2, alpha):
17     cos_a = np.cos(alpha)
18     sin_a = np.sin(alpha)
19     return (psi1 + mu * x1) * cos_a + (psi2 + mu * x2) * sin_a
20
21 # Calculation of lambda according (24)-(26)
22 def compute_lambda(a_eps_T):
23     threshold = epsilon**2 / np.sqrt(1 - epsilon) if epsilon < 1
    else np.inf
```

```
    if a_eps_T <= -threshold:
        return (-a_eps_T + np.sqrt(a_eps_T**2 + epsilon**2 * (epsilon + 1))) / (2 * (epsilon + 1))
    elif a_eps_T >= threshold:
        return (a_eps_T + np.sqrt(a_eps_T**2 + epsilon**2 * (epsilon + 1))) / (2 * (epsilon + 1))
    else:
        return np.sqrt((a_eps_T**2 + epsilon**3) / (4 * epsilon))

# Calculation mu on border according (27)-(29)
def compute_mu_on_boundary(psi1, psi2, psi3, x1, x2, alpha, a_eps, lambda_reg):
    cos_a = np.cos(alpha)
    sin_a = np.sin(alpha)
    xcos = x1 * cos_a + x2 * sin_a
    psi_dot_x = psi1 * x1 + psi2 * x2
    if a_eps <= -2 * epsilon * lambda_reg:
        return (2 * lambda_reg / epsilon**2) * xcos - psi_dot_x
    elif a_eps >= 2 * epsilon * lambda_reg:
        return -(2 * lambda_reg / epsilon**2) * xcos - psi_dot_x
    else:
        psi_cos = psi1 * cos_a + psi2 * sin_a
        numerator = -psi_cos * xcos + (epsilon**3) * psi_dot_x
        denominator = xcos**2 + epsilon**3
        return numerator / (denominator + 1e-10)

# Border check
def on_boundary(x1, x2):
    return abs(x1**2 + x2**2 - 1) < tol_boundary

# The system of differential equations
def system(t, Y, lambda_reg, mu_val):
    x1, x2, alpha = Y[0:3]
    psi1, psi2, psi3 = Y[3:6]
    a_eps = compute_a_eps(psi1, psi2, mu_val, x1, x2, alpha)
    p_eps = np.sign(a_eps) * min(abs(a_eps) / (2 * epsilon * lambda_reg), 1.0)
    u_eps = np.sign(psi3) * min(abs(psi3) / (2 * epsilon * lambda_reg), 1.0)
    v1_eps = (epsilon / (2 * lambda_reg)) * (psi1 + mu_val * x1)
    v2_eps = (epsilon / (2 * lambda_reg)) * (psi2 + mu_val * x2)
    dx1 = p_eps * np.cos(alpha) + epsilon * v1_eps
    dx2 = p_eps * np.sin(alpha) + epsilon * v2_eps
    dalpha = u_eps
    dpsi1 = -mu_val * (p_eps * np.cos(alpha) + epsilon * v1_eps)
    dpsi2 = -mu_val * (p_eps * np.sin(alpha) + epsilon * v2_eps)
```

```
    dpsi3 = p_eps * (
        np.sin(alpha) * (psi1 + mu_val * x1) -
        np.cos(alpha) * (psi2 + mu_val * x2)
    )
    dmu = 0.0
    return [dx1, dx2, dalpha, dpsi1, dpsi2, dpsi3, dmu]

# Adaptive integration step
def integrate_with_adaptive_step(t0, tf, y0, lambda_reg, mu_init, max_steps=10000):
    t_vals = [t0]
    y_vals = [y0]
    mu_vals = [mu_init]
    h = (tf - t0) / 100  # The initial step
    t = t0
    y = np.array(y0)
    mu_current = mu_init
    while t < tf and len(t_vals) < max_steps:
        h = min(h, tf - t)
        refined = 0
        success = False
        while refined <= max_refinements:
            h_try = h / (2 ** refined)
            try:
                k1 = np.array(system(t, y, lambda_reg, mu_current))
                k2 = np.array(system(t + h_try / 2, y + h_try * k1 / 2, lambda_reg, mu_current))
                k3 = np.array(system(t + h_try / 2, y + h_try * k2 / 2, lambda_reg, mu_current))
                k4 = np.array(system(t + h_try, y + h_try * k3, lambda_reg, mu_current))
                y_new = y + (h_try / 6) * (k1 + 2*k2 + 2*k3 + k4)
                x1_new, x2_new = y_new[0], y_new[1]
                crossed_boundary = on_boundary(y[0], y[1]) != on_boundary(x1_new, x2_new)
                violated = x1_new**2 + x2_new**2 < 1 - 1e-3
                if violated or (crossed_boundary and abs(mu_current) > 1e-2):
                    if refined < max_refinements:
                        refined += 1
                        continue
                    else:
                        h_try = h / (2 ** max_refinements)
                        y_new = y + (h_try / 6) * (k1 + 2*k2 + 2*k3 + k4)
                else:
```

```
                    t += h_try
                    y = y_new
                    t_vals.append(t)
                    y_vals.append(y)
                    if crossed_boundary and not violated:
                        a_eps = compute_a_eps(y[3], y[4], mu_current, y[0], y[1], y[2])
                        mu_new = compute_mu_on_boundary(y[3], y[4], y[5], y[0], y[1], y[2], a_eps, lambda_reg)
                        if abs(mu_new) < 1e-2:
                            mu_current = mu_new
                        else:
                            return None, None, None
                    mu_vals.append(mu_current)
                    success = True
                    break
            except Exception as e:
                if refined < max_refinements:
                    refined += 1
                else:
                    return None, None, None
        if not success:
            break
    return np.array(t_vals), np.array(y_vals), np.array(mu_vals)

# A function for the shooting method
def shooting(guess):
    theta, beta, T = guess
    psi1_T = np.cos(theta)
    psi2_T = np.sin(theta)
    psi3_T = 0.0
    alpha_T = beta
    a_eps_T = compute_a_eps(psi1_T, psi2_T, 0.0, B[0], B[1], alpha_T)
    global lambda_reg
    lambda_reg = compute_lambda(a_eps_T)
    y0 = [B[0], B[1], alpha_T, psi1_T, psi2_T, psi3_T, 0.0]
    t_vals, y_vals, mu_vals = integrate_with_adaptive_step(0, T, y0, lambda_reg, 0.0)
    if y_vals is None:
        return [1e6] * 3
    x1_0, x2_0, alpha_0 = y_vals[-1, 0], y_vals[-1, 1], y_vals[-1, 2]
    residual_x1 = x1_0 - A[0]
    residual_x2 = x2_0 - A[1]
    residual_alpha = alpha_0 - alpha0
```

```
145     return [residual_x1, residual_x2, residual_alpha]
146 
147 # Initial approximation
148 initial_guess = [np.pi + 0.5, -np.pi / 6, 3.8]
149 
150 # Solution
151 result = root(shooting, initial_guess, method='lm', tol=1e-4)
152 
153 if result.success:
154     theta_opt, beta_opt, T_opt = result.x
155     psi1_T = np.cos(theta_opt)
156     psi2_T = np.sin(theta_opt)
157     a_eps_T = compute_a_eps(psi1_T, psi2_T, 0.0, B[0], B[1],
    beta_opt)
158     lambda_reg = compute_lambda(a_eps_T)
159     y0 = [B[0], B[1], beta_opt, psi1_T, psi2_T, 0.0, 0.0]
160     t_vals, y_vals, mu_vals = integrate_with_adaptive_step(0, T_opt,
        y0, lambda_reg, 0.0)
161     if y_vals is not None:
162         t = t_vals
163         x1 = y_vals[:, 0][::-1]  # From A to B
164         x2 = y_vals[:, 1][::-1]
165         alpha = y_vals[:, 2][::-1]
166         mu_vals = mu_vals[::-1]
167 
168         # Trajectory graph
169         plt.figure(figsize=(10, 8))
170         theta_circle = np.linspace(0, 2*np.pi, 100)
171         plt.plot(np.cos(theta_circle), np.sin(theta_circle), 'k-',
    linewidth=2, label='Obstacle $x_1^2 + x_2^2 = 1$')
172         plt.plot(x1, x2, 'b-', linewidth=2, label='The trajectory')
173         plt.plot(A[0], A[1], 'go', label='A')
174         plt.plot(B[0], B[1], 'ro', label='B')
175         plt.xlabel('$x_1$')
176         plt.ylabel('$x_2$')
177         plt.axis('equal')
178         plt.grid(True)
179         plt.legend()
180         plt.title('Trajectory with adaptive step')
181         plt.show()
182 
183         # Graph of mu(t)
184         plt.figure(figsize=(8, 4))
185         plt.plot(t, mu_vals, 'm-', linewidth=2, label=r'$\mu(t)$')
186         plt.axhline(0, color='k', linestyle='--', linewidth=0.8)
187         plt.xlabel('Time')
```

```
        plt.ylabel(r'$\mu(t)$')
        plt.grid(True)
        plt.legend()
        plt.title('The Lagrange multiplier $\mu(t)$')
        plt.show()

        # Control p and u
        plt.subplot(3, 1, 3)
        a_eps_vals = [compute_a_eps(psi1_i, psi2_i, mu_i, x1_i, x2_i
    , alpha_i)
                      for x1_i, x2_i, alpha_i, psi1_i, psi2_i, mu_i in
    zip(x1, x2, alpha, psi1, psi2, mu)]
        p_vals = [compute_p(a) for a in a_eps_vals]
        u_vals = [compute_u(psi3_i) for psi3_i in psi3]
        plt.plot(t, p_vals, label=r'$p(t)$', drawstyle='steps-mid',
    linewidth=2)
        plt.plot(t, u_vals, label=r'$u(t)$', drawstyle='steps-mid',
    linewidth=2)
        for t_sw in switch_times_p:
            plt.axvline(t_sw, color='orange', linestyle='--',
    linewidth=1.5, label='Switching p' if t_sw == switch_times_p[0]
    else "")
        for t_sw in switch_times_u:
            plt.axvline(t_sw, color='magenta', linestyle='--',
    linewidth=1.5)
        plt.xlabel("Time $t$")
        plt.ylabel("Control")
        plt.legend()
        plt.grid(True)

    plt.tight_layout()
    plt.show()

else:
    print("The shooting method didn't match")
    print("Message:", result.message)
```

A.2 Листинг программы для точного нахождения оптимального пути между двумя точками при наличии фазовых ограничений глубины 2

```
import numpy as np
```

```
from scipy.optimize import fsolve
import matplotlib.pyplot as plt

# Problem parameters
T = 3
R = 1
A = np.array([-1.3, 0])
B = np.array([1.3, 0])
V0 = np.array([0, 0])

# Method parameters
P = 2
h = 0.01 * P
eps = 0.02 * P
eps_R = 0.05
reg = 0.1
delta = 0.1
kappa = 0.5
const = 5
L = 3
N = int(T // (h * kappa))

# Scanning sector (spherical angles)
a1, a2 = 0, np.pi
b1, b2 = 0, np.pi
c1, c2 = 0, 2 * np.pi

# Functions
def norm2(x):
    return np.dot(x, x)

def prod(x, y):
    return np.dot(x, y)

def con(T_adm, delta):
    def F(z):
        return z*z * (delta + (1 + 2*T_adm) * np.exp(z*z)) - delta
    z = fsolve(F, 1.0)[0]
    C = (2 * np.exp(z*z) * (1/delta)) / (2*z*z - (1/delta)*(1 + 2*
   T_adm)*z*z*z*z*np.exp(z*z))
    return C

# Preallocate arrays
x = np.zeros((N, 2))
y = np.zeros((N, 2))
u = np.zeros((N, 2))
```

```
psi_x = np.zeros((N, 2))
psi_y = np.zeros((N, 2))
mu = np.zeros(N)
x_min = np.zeros((N, 2))
u_min = np.zeros((N, 2))
mu_min = np.zeros(N)
Time = np.zeros(N)

# Initialize minimum time
T_min = T
eps2 = eps * eps
h0 = h

s1 = a1 - h
while s1 < a2:
    s1 += h
    rho = np.sqrt(1 - s1*s1).real
    print(f"s1={s1:.10f}")

    theta = b1 - h
    while theta < b2:
        theta += h
        phi = c1 - h
        while phi < c2:
            phi += h

            # Initializing the initial conditions
            x[0] = A.copy()
            y[0] = V0.copy()
            psi_x[0] = [np.cos(s1), np.sin(s1) * np.cos(theta)]
            psi_y[0] = [
                np.sin(s1) * np.sin(theta) * np.sin(phi),
                np.sin(s1) * np.sin(theta) * np.cos(phi)
            ]
            mu[0] = 0
            u[0] = [0, 0]

            i = 1
            flag = 0
            h_step = h0  # current step

            while Time[i-1] < T_min:
                Time[i] = Time[i-1] + h_step

                # Adaptive step
                if norm2(x[i-1]) < (R + eps_R)**2 and flag == 0:
```

```
                    h_step *= kappa
                    flag = 1
                elif norm2(x[i-1]) > (R + eps_R)**2 and flag == 1:
                    h_step = h0
                    flag = 0

                # The "landing" condition
                if norm2(x[i-1]) < (R + eps)**2 and abs(prod(x[i-1],
 y[i-1])) < eps2:
                    s = i - 1
                    while Time[s] < T_min:
                        s += 1
                        Time[s] = Time[s-1] + h_step
                        if s >= N:
                            break

                        R0 = np.sqrt(norm2(x[s-1]))
                        r = norm2(y[s-1])**2 - R0**2

                        if abs(r) < delta:
                            print(f"Regularity fails at t = {s *
 h_step:.10f} :-(")
                            i = N
                            break

                        if s - i < L:
                            mu[s] = 0
                        else:
                            beta = prod(psi_y[s-1], x[s-1])
                            D = r * (beta**2 - norm2(psi_y[s-1]) *
 R0**2)
                            if D < 0:
                                mu[s] = 0
                            else:
                                mu[s] = (beta * r - norm2(y[s-1]) *
 np.sqrt(D)) / (R0**2 * r)
                            mu[s] = np.real(mu[s])

                        if not ((mu[s] < mu[s-1] + eps2) and (mu[s]
 > -const * (s - i + 1) * h_step)):
                            i = N
                            break

                        alpha = psi_y[s-1] - mu[s] * x[s-1]
                        w = np.sqrt(norm2(alpha))
                        if w > reg:
```

```
                            u[s] = alpha / w
                        else:
                            u[s] = u[s-1].copy()

                        x[s] = x[s-1] + h_step * y[s-1]
                        y[s] = y[s-1] + h_step * u[s]
                        psi_x[s] = psi_x[s-1] + h_step * mu[s] * u[s]
                        psi_y[s] = psi_y[s-1] - h_step * psi_x[s-1] + h_step * 2 * mu[s] * y[s-1]

                        pa = prod(x[s], y[s])
                        pb = pa / (R0**2)
                        y[s] = y[s] - x[s] * pb

                        if norm2(x[s] - B) < eps2:
                            if s * h_step < T_min:
                                T_min = s * h_step
                                x_min[:s+1] = x[:s+1].copy()
                                u_min[:s+1] = u[:s+1].copy()
                                mu_min[:s+1] = mu[:s+1].copy()
                                print(f"Post-boundary hitting the target detected at t = {T_min:.10f} !!!")
                                print(f"theta = {theta:.10f}, phi = {phi:.10f}")
                                N = s + 1

                        if norm2(x[s]) < (R - eps)**2 or norm2(x[i-1] - B) > 2 * norm2(A - B):
                            break

                        if norm2(x[s]) > (R + eps_R)**2 and flag == 1:
                            h_step = h0
                            flag = 0

                    break

                else:
                    mu[i] = 0
                    alpha = psi_y[i-1]
                    w = np.sqrt(norm2(alpha))
                    if w > reg:
                        u[i] = alpha / w
                    else:
                        u[i] = u[i-1].copy()
```

```
174
175                     x[i] = x[i-1] + h_step * y[i-1]
176                     y[i] = y[i-1] + h_step * u[i]
177                     psi_x[i] = psi_x[i-1].copy()
178                     psi_y[i] = psi_y[i-1] - h_step * psi_x[i-1]
179
180                     if norm2(x[i] - B) < eps2:
181                         if i * h_step < T_min:
182                             T_min = i * h_step
183                             x_min[:i+1] = x[:i+1].copy()
184                             u_min[:i+1] = u[:i+1].copy()
185                             mu_min[:i+1] = mu[:i+1].copy()
186                             print(f"Hitting the target detected at t
    = {T_min:.10f}")
187                             N = i + 1
188
189                     if norm2(x[i]) < R**2 or norm2(x[i-1] - B) > 2 *
    norm2(A - B):
190                         break
191
192                     i += 1
193                     if i >= N:
194                         break
195
196             flag = 0
197             h_step = h0
198
199 print(f"T_min = {T_min:.10f}")
200
201 # Plots
202 t_steps = min(int(T_min // h0) + 1, len(x_min))
203
204 plt.figure(1)
205 plt.axis('equal')
206 plt.plot(A[0], A[1], 'h', markersize=7, markerfacecolor='green',
    label='Start A')
207 plt.plot(B[0], B[1], 'h', markersize=7, markerfacecolor='red', label
    ='Target B')
208 circle = plt.Circle((0, 0), R, color='blue', fill=False, linestyle='
    --')
209 plt.gca().add_patch(circle)
210 plt.plot(x_min[:t_steps, 0], x_min[:t_steps, 1], 'b-', linewidth
    =1.5)
211 plt.title("Trajectory")
212 plt.xlabel("x")
213 plt.ylabel("y")
```

```
plt.legend()
plt.grid(True)

plt.figure(2)
plt.plot(Time[:t_steps], mu_min[:t_steps], 'r-', linewidth=1.2)
plt.title(r'$\mu(t)$')
plt.xlabel('Time')
plt.ylabel(r'$\mu$')
plt.grid(True)

plt.figure(3)
plt.plot(Time[:t_steps], u_min[:t_steps, 0], label=r'$u_1(t)$',
    linewidth=1.2)
plt.plot(Time[:t_steps], u_min[:t_steps, 1], label=r'$u_2(t)$',
    linewidth=1.2)
plt.title('Control $u(t)$')
plt.xlabel('Time')
plt.ylabel('u')
plt.legend()
plt.grid(True)

plt.show()
```

Приложение B

Листинг программы к главе 3

B.1 Листинг программы для точного нахождения оптимального пути между двумя точками при наличии круговых препятствий

****

Модуль Shortpath.py (главный исполняемый модуль)

****

```
1
2  import dijkstra as dj
3  import aux_classes as aux
4  import draw_routines as draw
5  import file_routines as file
6  import matplotlib.pyplot as plt
7  import numpy as np
8  import geometry as geom
9  import designators
10 import argparse
11 from matplotlib import patches
12
13 class Formatter:   # class to generate circle names with leading
       zeros
14     def __init__(self, nmax: int):
15         self.ndigits: int = 0
16         while nmax // (10 ** self.ndigits):
17             self.ndigits += 1
18         self.formatstring = '{:0' + f'{self.ndigits}' + 'd}'
19
```

```
20      def format(self, n):
21          return self.formatstring.format(n)
22
23
24  def generate_graph(start, finish, circles, umax: float = 1, base:
        float = 1):
25      # names taken from designators.py
26      # st = 'S'    # name for start point
27      # fn = 'F'    # name for finish point
28      # cp = 'A'    # name for point on a circle
29      # # ex = 'B'    # name for exit point on a circle
30      # rt = 'R'    # designator for right-side connection
31      # lt = 'L'    # designator for left-side connection
32      # na = 'N'    # designator for ignored name
33      # dl = '_'    # delimiter in point names
34      st = designators.d_start
35      fn = designators.d_finish
36      cp = designators.d_circlepoint
37      rt = designators.d_right
38      lt = designators.d_left
39      dl = designators.d_delimiter
40      if umax is None or umax <= 0:
41          umax = np.float64(1)
42      if base is None or base <= 0:
43          base = np.float64(1)
44      ncircles = len(circles)
45      fmt = Formatter(ncircles)
46      fmt_st = st * fmt.ndigits
47      fmt_fn = fn * fmt.ndigits
48      cn = [fmt.format(i) for i in range(ncircles)]   # circle names
        formatted to have uniform length
49      g = dj.Graph(0)   # initializing empty graph
50      # adding start point
51      g.add_vertex(st, start)
52      # adding finish point
53      g.add_vertex(fn, finish)
54      # trying to add direct connection from start to finish
55      if not geom.is_obstructed([start, finish], circles):
56          # needs testing
57          conn = aux.Line(start, finish, umax)
58          name1 = dl.join([st, fn])
59          name2 = dl.join([fn, st])
60          g.add_vertex(name1, conn.begin)
61          g.add_vertex(name2, conn.end)
62          g.add_connection(name1, name2, conn)
63          phi = conn.begin[2] - start[2]
```

```
64          g.add_connection(st, name1, aux.Rotation(start, phi, umax,
    base))
65          rev = conn.end
66          rev[2] += np.pi
67          phi = finish[2] - rev[2]
68          g.add_connection(name2, fn, aux.Rotation(rev, phi, umax,
    base))
69      # adding circles
70      for i in range(ncircles):
71          # connecting to start point
72          # finding tangent lines between start and circle
73          n, lines = geom.tangent_p2c(start, circles[i])   # points:
    left, right looking from start
74          if n >= 2:
75              p_names = [dl.join([st, cn[i], rt]), dl.join([st, cn[i],
     lt])]
76              for k in [0, 1]:   # 0 for left, 1 for right
77                  if not geom.is_obstructed(lines[k], circles):
78                      # since we do not plan to go back to start we
    only create directed connection from start to circle
79                      # and no reverse connection. Same for rotations,
     and we do not add intermediate rotations between
80                      # all directions that exit from starting point
    towards circles
81                      g.add_vertex(p_names[k], lines[k][0])   # adding
     exit point from start
82                      #  connecting start point to exit point with
    rotation
83                      phi = lines[k][0][2] - start[2]
84                      g.add_connection(st, p_names[k], aux.Rotation(
    start, phi, umax))
85                      # adding entry point on circle
86                      name = dl.join([cp, cn[i], fmt_st, [rt, lt][k]])
87                      g.add_vertex(name, lines[k][1])
88                      # connecting exit point to circle (using data
    from tangent_p2c output)
89                      g.add_connection(p_names[k], name, aux.Line(
    lines[k][0], lines[k][1], umax))
90          # finding tangent lines between circle and finish
91          n, lines = geom.tangent_c2p(circles[i], finish)  # points:
    left, right looking from circle
92          if n >= 2:
93              p_names = [dl.join([fn, cn[i], lt]), dl.join([fn, cn[i],
     rt])]
94              for k in [0, 1]:  # 0 for left, 1 for right
95                  if not geom.is_obstructed(lines[k], circles):
```

```
96                  # since we do not plan to go back from finish we
     only create directed connection from finish
97                  # to circle and no reverse connection. Same for
    rotations as with start point.
98                  g.add_vertex(p_names[k], lines[k][1])  # adding
    entry point to finish
99                  #  connecting entry point to finish point with
    rotation
100                 phi = finish[2] - lines[k][1][2]   # taking
    outgoing direction of line
101                 g.add_connection(p_names[k], fn, aux.Rotation(
    lines[k][1], phi, umax, base))
102                 # adding exit point on circle
103                 name = dl.join([cp, cn[i], fmt_fn, [lt, rt][k]])
104                 g.add_vertex(name, lines[k][0])
105                 # connecting circle point to exit point (using
    data from tangent_c2p output)
106                 g.add_connection(name, p_names[k], aux.Line(
    lines[k][0], lines[k][1], umax))
107         # connecting to other circles
108         for j in range(i+1, ncircles):
109             n, lines = geom.tangent_c2c(circles[i], circles[j])
110             if n >= 4:    # assuming all lines are present = circles
     do not overlap
111                 # names for first circle   # [type, from, to,
    touch_from, touch_to]
112                 p_names1 = [dl.join([cp, cn[i], cn[j], lt, lt]), dl.
    join([cp, cn[i], cn[j], rt, rt]),
113                             dl.join([cp, cn[i], cn[j], lt, rt]), dl.
    join([cp, cn[i], cn[j], rt, lt])]
114                 # names for second circle
115                 p_names2 = [dl.join([cp, cn[j], cn[i], rt, rt]), dl.
    join([cp, cn[j], cn[i], lt, lt]),
116                             dl.join([cp, cn[j], cn[i], lt, rt]), dl.
    join([cp, cn[j], cn[i], rt, lt])]
117                 for k in range(n):
118                     if not geom.is_obstructed(lines[k], circles):
119                         # adding vertices for points on both circles
120                         g.add_vertex(p_names1[k], lines[k][0])
121                         g.add_vertex(p_names2[k], lines[k][1])
122                         # adding 2 connections in both directions
123                         g.add_connection(p_names1[k], p_names2[k],
    aux.Line(lines[k][0], lines[k][1], umax))
124                         g.add_connection(p_names2[k], p_names1[k],
    aux.Line(lines[k][1], lines[k][0], umax))
125     # adding arc connections between all points on a single circle
```

```
126     for i in range(ncircles):
127         des_len = len(cp)
128         del_len = len(dl)
129         ind_len = fmt.ndigits
130         # looking for all points that have circle number <i> at the
    second position in name
131         names = [n for n in g.vertex_data if n[0:des_len] == 'A' and
132                  n[des_len + del_len: des_len + del_len + ind_len]
    == fmt.format(i)]
133         prefix_len = len(cp) + 2 * fmt.ndigits + 3 * len(dl)
134         names_l = [n for n in names if n[prefix_len: prefix_len + 1]
     == 'L']
135         names_r = [n for n in names if n[prefix_len: prefix_len + 1]
     == 'R']
136         for n1 in names_l:
137             idx1 = g.vertex_data.index(n1)
138             for n2 in names_r:
139                 idx2 = g.vertex_data.index(n2)
140                 conn = aux.Arc(circles[i], g.vertex_locations[idx1],
     g.vertex_locations[idx2], 1, umax, base)
141                 dphi = conn.get_dphi()   # should be nonnegative
142                 if dphi < np.pi:   # only adding connection if arc
    angle is less than pi
143                     g.add_connection(n1, n2, conn)
144                     g.add_connection(n2, n1, aux.Arc(circles[i], g.
    vertex_locations[idx2], g.vertex_locations[idx1],
145                                                      -1, umax, base)
    )
146     return g
147
148
149 def main():
150     ### Usage in command line: shortpath.py [-d / -t] [-f setupfile.
    txt]
151     ### -d for distance, -t for time (default)
152     ### setupfile.txt:  file with scenario setup
153     #
    ######################################################################

154     parser = argparse.ArgumentParser('shortpath')
155     parser.add_argument('-t', '--time', help='Search mode, use -d
    for distance or -t for time (default).',
156                         action='store_true')
157     parser.add_argument('-d', '--distance', help='Search mode, use -
    d for distance or -t for time (default).',
158                         action='store_true')
```

```
159     parser.add_argument('-f', '--filename',
160                         help='-f <name>: Name of setup file to
    process. Will open a dialog if none provided.', type=str,
161                         default='')
162     args = parser.parse_args()
163     mode = 't'  # 't' for time, 'd' for distance
164     if args.time:
165         mode = 't'
166     elif args.distance:
167         mode = 'd'
168 
169     if args.filename == '':
170         in_name = file.get_setup_filename()
171     else:
172         in_name = args.filename
173 
174     print(in_name)
175 
176     start, end, circles, umax, base = file.parse_setup(in_name)
177     out_name = file.generate_out_name(in_name)
178 
179     if umax is None:
180         umax = 1
181     g = generate_graph(start, end, circles, umax, base)
182     # g.set_umax(0.5)
183     p_start = designators.d_start
184     p_finish = designators.d_finish
185     distance, path = g.dijkstra(p_start, p_finish, mode)
186 
187     print(path)
188     #   temporary for checking conections
189     # arr = [aux.Connection] * 0
190     # arr = []
191     # for i in range(len(g.vertex_data)):
192     #     for j in range(len(g.vertex_data)):
193     #         if g.conn_matrix[i][j] is not None:
194     #             arr.append(id(g.conn_matrix[i][j]))
195     # for i in range(len(arr)):
196     #     print(arr[i], '\n')
197     # for i in range(len(arr)):
198     #     if arr[i] in arr[0:i]:
199     #         print('!!!')
200     if mode == 'd':
201         modestr = 'Distance: '
202     elif mode == 't':
203         modestr = 'Time: '
```

```
204     print(modestr, distance)
205     lines = []
206     arcs = []
207     rotations = []
208     drawlines = None
209     cpoints = None
210     drawrots = None
211     drawarcs = None
212     gnames = g.vertex_data
213     for i in range(len(path) - 1):
214         conn = g.conn_matrix[gnames.index(path[i]), gnames.index(
    path[i + 1])]
215         if type(conn) is aux.Line:
216             lines.append(conn)
217             # lines = np.append(lines, conn)
218         elif type(conn) is aux.Arc:
219             arcs.append(conn)
220         elif type(conn) is aux.Rotation:
221             rotations.append(conn)
222     if len(lines) > 0:
223         drawlines = draw.draw_lines(lines)
224     if len(arcs) > 0:
225         drawarcs = draw.draw_arcs(arcs)
226     if len(rotations) > 0:
227         drawrots = draw.draw_rotations(rotations)
228     if len(circles) > 0:
229         cpoints = draw.draw_circles(circles)
230     file.save_report(out_name, g, path, mode, distance)
231 
232     fig1 = plt.figure(1)
233     ax = fig1.add_subplot(1,1,1)
234     if not cpoints is None:
235         plt.plot(cpoints[0], cpoints[1], 'r')
236         for c in circles:
237             p = patches.Circle((c[0],c[1]), c[2], fill=False, hatch=
    'x', color='r')
238             ax.add_patch(p)
239     if not drawlines is None:
240         plt.plot(drawlines[0], drawlines[1], 'c')
241     # plt.plot(drawlines[0], drawlines[1], 'c.')
242     if not drawarcs is None:
243         plt.plot(drawarcs[0], drawarcs[1], 'k')
244     if not drawrots is None:
245         plt.plot(drawrots[0], drawrots[1], 'k')
246     # plt.plot(dl2[0], dl2[1])
247     # plt.plot(da2[0], da2[1])
```

```
248     plt.grid()
249     plt.gca().set_aspect('equal')
250     plt.show()
251 
252 
253 if __name__ == '__main__':
254     main()
```

****

Модуль aux classes.py

****

```
1 
2 from abc import abstractmethod
3 import numpy as np
4 
5 class Connection:
6     def __init__(self):
7         pass
8 
9     @abstractmethod
10     def get_length(self):
11         pass
12 
13     @abstractmethod
14     def get_time(self):
15         pass
16 
17     @abstractmethod
18     def set_umax(self, umax):
19         pass
20 
21     @abstractmethod
22     def calc_len_time(self):
23         pass
24 
25 
26 def normalize_angle(phi: float):   # normalizing angle phi to be in
    range (-pi, pi]
27     p = phi
28     while p > np.pi:
29         p -= 2 * np.pi
30     while p <= -np.pi:
31         p += 2 * np.pi
32     return p
33 
34 
```

```
35 def point_check(p):
36     errorlevel = np.uint8(0)
37     l: int = 0
38     try:
39         l = len(p)
40     except TypeError:
41         errorlevel += 1   # p is not iterable
42     if not l == 3:
43         errorlevel += 2   # length of p is not 3 (x, y, and
    direction angle)
44     else:
45         if False in np.isreal(p):
46             errorlevel += 4   # elements of p are not numbers
47     return errorlevel
48
49
50 def circle_check(c):
51     errorlevel = np.uint8(0)
52     l: int = 0
53     try:
54         l = len(c)
55     except TypeError:
56         errorlevel += 1   # c is not iterable
57     if not l == 3:
58         errorlevel += 2   # length of c is not 3
59     else:
60         if False in np.isreal(c):
61             errorlevel += 4   # elements of p are not numbers
62         if errorlevel <= 4:
63             if c[2] <= 0:
64                 errorlevel += 8   # circle radius must be >0
65     return errorlevel
66
67
68 def report_point(el):
69     if el & 0x1 > 0:
70         raise ValueError('Error: point initializer is not a sequence
    of two numbers')
71     if el & 0x2 > 0:
72         raise ValueError('Error: length of the point initializer
    must be 3')
73     if el & 0x4 > 0:
74         raise ValueError('Error: elements of the point initializer
    must be real values')
75
76
```

```
77 def report_circle(el):
78     if el & 0x1 > 0:
79         raise ValueError('Error: point initializer is not a sequence
       of two numbers')
80     if el & 0x2 > 0:
81         raise ValueError('Error: length of the point initializer
      must be 3')
82     if el & 0x4 > 0:
83         raise ValueError('Error: elements of the point initializer
      must be real values')
84     if el & 0x8 > 0:
85         raise ValueError('Error: circle radius must be positive')
86
87
88 class Line(Connection):
89     def __init__(self, p1, p2, umax=1):
90         super().__init__()
91         if umax > 0:
92             self.umax = np.float64(umax)
93         else:
94             self.umax = 0
95         el1 = point_check(p1)
96         el2 = point_check(p2)
97         if not el1 == 0 and el2 == 0:
98             report_point(el1)
99             report_point(el2)
100        phi = np.atan2(p2[1] - p1[1], p2[0] - p1[0])
101        self.begin = np.array(p1, dtype=np.float64)
102        self.begin[2] = normalize_angle(phi)
103        self.end = np.array(p2, dtype=np.float64)
104        self.end[2] = normalize_angle(phi)   # update 0.2
105        self.length = float('inf')
106        self.time = float('inf')
107
108    def __getitem__(self, item):
109        if item == 0:
110            return self.begin
111        elif item == 1:
112            return self.end
113        else:
114            raise IndexError(f'Error: index {item} out of range for
      Line class')
115
116    def get_length(self):
117        if self.length == float('inf'):
118            self.calc_len_time()
```

```
119         return self.length
120
121     def get_time(self):
122         if self.length == float('inf'):
123             self.calc_len_time()
124         return self.time
125
126     def set_umax(self, umax):
127         if umax > 0:
128             self.umax = np.float64(umax)
129             self.length = float('inf')
130
131     def calc_len_time(self):
132         self.length = np.linalg.norm(self.end[0:2] - self.begin
    [0:2])   # only x y coordinates in length
133         if self.umax > 0:
134             # self.time = 0.5 * self.length / self.umax
135             self.time = self.length / self.umax  # model with v = (
    u1 + u2) / 2
136
137
138 class Arc(Connection):
139     def __init__(self, c, p1, p2, dir=0, umax=1, base=1):
140         super().__init__()
141         if umax > 0:
142             self.umax = np.float64(umax)
143         else:
144             self.umax = np.float64(0)
145         if base > 0:
146             self.base = np.float64(base)
147         else:
148             self.base = np.float64(1)
149         el1 = point_check(p1)
150         el2 = point_check(p2)
151         el3 = circle_check(c)
152         if not el1 == 0 and el2 == 0 and el3 == 0:
153             report_point(el1)
154             report_point(el2)
155             report_circle(el3)
156
157         #  normalizing points to be on circle
158         cen = np.array(c, dtype=np.float64)
159         vec = np.array([p1[0] - c[0], p1[1] - c[1], 0], dtype=np.
    float64)
160         self.begin = cen + vec * c[2] / np.linalg.norm(vec)
161         phi1 = np.atan2(self.begin[1] - cen[1], self.begin[0] - cen
```

```
    [0])
162
163         # self.begin[2] = p1[2]   # restoring angle to the point p1
164         vec = np.array([p2[0] - c[0], p2[1] - c[1], 0], dtype=np.
    float64)
165         self.end = cen + vec * c[2] / np.linalg.norm(vec)
166         phi2 = np.atan2(self.end[1] - cen[1], self.end[0] - cen[0])
167
168         # self.end[2] = p2[2]   # restoring angle to the point p2
169         self.circle = cen
170
171         if not np.isreal(dir):
172             raise ValueError('Error: direction must be a real number
    . Use 0 for automatic selection.')
173         if dir == 0:
174             # determining dir to have shortest arc
175             if np.sin(phi2 - phi1) >= 0:
176                 self.dir = np.float64(1)
177             else:
178                 self.dir = np.float64(-1)
179         elif dir > 0:
180             self.dir = np.float64(1)
181         elif dir < 0:
182             self.dir = np.float64(-1)
183         # assigning self angles for begin and end
184         self.phi_s = phi1
185         self.phi_f = phi2
186         # amending direction angles for begin and end according to
    dir
187         self.begin[2] = normalize_angle(phi1 + 0.5 * np.pi * self.
    dir)
188         self.end[2] = normalize_angle(phi2 + 0.5 * np.pi * self.dir)
       # update 0.2 "-" -> "+"
189         self.length = float('inf')
190         self.time = float('inf')
191
192     def __getitem__(self, item):
193         if item == 0:
194             return self.begin
195         elif item == 1:
196             return self.end
197         else:
198             raise IndexError(f'Error: index {item} out of range for
    Arc class')
199
200     def get_length(self):
```

```
201         if self.length == float('inf'):
202             self.calc_len_time()
203         return self.length
204 
205     def get_time(self):
206         if self.length == float('inf'):
207             self.calc_len_time()
208         return self.time
209 
210     def set_umax(self, umax):
211         if umax > 0:
212             self.umax = umax
213             self.length = float('inf')
214 
215     def set_base(self, base):
216         if base > 0:
217             self.base = base
218 
219     def get_dphi(self):
220         phi = np.array([self.phi_s, self.phi_f])  # update 0.2
221         if self.dir > 0 and phi[1] < phi[0]:
222             phi[1] += 2 * np.pi
223         if self.dir < 0 and phi[1] > phi[0]:
224             phi[1] -= 2 * np.pi
225         dphi = np.max(phi) - np.min(phi)
226         return dphi
227 
228     def calc_len_time(self):
229         x = self.circle[0]
230         y = self.circle[1]
231         r = self.circle[2]
232         b = self.base
233 
234 
235         dphi = self.get_dphi()
236         self.length = r * dphi
237         # vmax = self.umax * 2 * r / (r + 1)
238         vmax = self.umax * 2 * r / (2 * r + b)
239         if vmax > 0:
240             self.time = self.length / vmax
241 
242 
243 class Rotation(Connection):
244     def __init__(self, p, phi, umax=1, base=1):
245         super().__init__()
246         if umax > 0:
```

```
247             self.umax = np.float64(umax)
248         else:
249             self.umax = np.float64(0)
250         if base > 0:
251             self.base = np.float64(base)
252         else:
253             self.base = np.float64(1)
254         el1 = point_check(p)
255         if not el1 == 0:
256             report_point(el1)
257         if not np.isreal(phi):
258             raise ValueError('Error: rotation angle must be a real
    number')
259         self.phi = normalize_angle(phi)
260         if self.phi >= 0:
261             self.dir = np.float64(1)
262         else:
263             self.dir = np.float64(-1)
264         self.phi = self.phi * self.dir
265 
266         self.begin = np.array(p, dtype=np.float64)
267         self.end = np.array(p, dtype=np.float64)
268         self.end[2] = self.begin[2] + self.phi * self.dir
269 
270         self.length = float('inf')
271         self.time = float('inf')
272 
273     def get_length(self):
274         if self.length == float('inf'):
275             self.calc_len_time()
276         return self.length
277 
278     def get_time(self):
279         if self.length == float('inf'):
280             self.calc_len_time()
281         return self.time
282 
283     def calc_len_time(self):
284         self.length = 0
285         if self.umax > 0:
286             # self.time = 0.5 * np.fabs(self.phi) / self.umax
287             self.time = 0.5 * self.base * np.fabs(self.phi) / self.
    umax
288 
289     def set_umax(self, umax):
290         if umax > 0:
```

```
            self.umax = np.float64(umax)
            self.length = float(inf)


def main():
    p1 = np.array([-1, 0, 0])
    p2 = np.array([0, 1, 0])
    c = np.array([0, 0, 1])
    a = Arc(c, p2, p1, -1, 1, 1)
    print(a.get_length())
    print(a.begin, ' ', a.end)


if __name__ == '__main__':
    main()
```

****

Модуль designators.py

****

```

#  defines designators used in names of graph vertices

d_start = 'S'   # name for start point
d_finish = 'F'   # name for finish point
d_circlepoint = 'A'   # name for point on a circle
d_entrypoint = 'A'   # name for entry point of a circle (not
   implemented)
d_exitpoint = 'B'   # name for exit point on a circle (not
   implemented)
d_right = 'R'   # designator for right-side connection
d_left = 'L'   # designator for left-side connection
d_na = 'N'   # designator for ignored name
d_delimiter = '_'   # delimiter in point names
d_typeminus = '-'   # designator for type 1 connection (incoming
   line -> clockwise rotation)
d_typeplus = '+'   # designator for type 2 connection (incoming line
    -> counter-clockwise rotation)
```

********** Модуль dijkstra.py **********

```

from aux_classes import *
import sys


```

```
class Graph:
    def __init__(self, size=0):
        # self.adj_matrix = [[0] for _ in range(size)]
        # self.adj_matrix = -np.ones((size, size))
        self.conn_matrix = np.empty((size, size), dtype=Connection)
        self.size = size
        self.vertex_data = [''] * size
        self.vertex_locations = [np.array(2)] * size

    def set_edge(self, u, v, weight):
        if 0 <= u < self.size and 0 <= v < self.size:
            self.adj_matrix[u][v] = weight
            # self.adj_matrix[v][u] = weight  # For undirected graph

    def set_vertex_data(self, vertex, data):
        if 0 <= vertex < self.size:
            self.vertex_data[vertex] = data

    def add_vertex(self, data, location):
        self.vertex_data.append(data)
        self.vertex_locations.append(location)
        self.size += 1
        # self.adj_matrix = np.pad(self.adj_matrix, ((0, 1), (0, 1))
    , mode='constant', constant_values=-1)
        self.conn_matrix = np.pad(self.conn_matrix, ((0, 1), (0, 1))
    , mode='empty')

    def add_connection(self, v1, v2, conn):
        i1 = self.vertex_data.index(v1)
        i2 = self.vertex_data.index(v2)
        # self.adj_matrix[i1][i2] = conn.get_time()
        self.conn_matrix[i1][i2] = conn

    def dijkstra(self, start_vertex_data, end_vertex_data, mode='t')
    :
        start_vertex = self.vertex_data.index(start_vertex_data)
        end_vertex = self.vertex_data.index(end_vertex_data)
        distances = [float('inf')] * self.size
        hops = [sys.maxsize] * self.size   # number of hops
        predecessors = [None] * self.size
        distances[start_vertex] = 0
        hops[start_vertex] = 0
        visited = [False] * self.size

        for _ in range(self.size):
            min_distance = float('inf')
```

```
49              u = None
50              for i in range(self.size):
51                  if not visited[i] and distances[i] < min_distance:
52                      min_distance = distances[i]
53                      u = i
54
55              if u is None or u == end_vertex:
56                  # debug info
57                  # print(f"Breaking out of loop. Current vertex: {
    self.vertex_data[u]}")
58                  # print(f"Distances: {distances}")
59                  break
60
61              visited[u] = True
62              # debug info
63              # print(f"Visited vertex: {self.vertex_data[u]}")
64
65              for v in range(self.size):
66                  # if self.adj_matrix[u][v] != -1 and not visited[v]:
67                  #     alt = distances[u] + self.adj_matrix[u][v]   #
     + self.conn_matrix.get_time()
68                  #     if alt < distances[v]:
69                  #         distances[v] = alt
70                  #         predecessors[v] = u
71                  if not self.conn_matrix[u][v] is None and not
    visited[v]:
72                      if mode == 'd':
73                          alt = distances[u] + self.conn_matrix[u][v].
    get_length()
74                      elif mode == 't':
75                          alt = distances[u] + self.conn_matrix[u][v].
    get_time()
76                      alt_hops = hops[u] + 1
77                      if alt < distances[v] or (alt == distances[v]
    and alt_hops < hops[v]):
78                          distances[v] = alt
79                          hops[v] = alt_hops   # if equal distance,
    choose path with fewest hops
80                          predecessors[v] = u
81
82          return distances[end_vertex], self.get_path(predecessors,
    start_vertex_data, end_vertex_data)
83
84      def get_path(self, predecessors, start_vertex, end_vertex):
85          path = []
86          current = self.vertex_data.index(end_vertex)
```

```
87          while current is not None:
88              path.insert(0, self.vertex_data[current])
89              current = predecessors[current]
90              if current == self.vertex_data.index(start_vertex):
91                  path.insert(0, start_vertex)
92                  break
93          # return '->'.join(path)  # Join the vertices with '->'
94          return path
95
96      def set_umax(self, umax):
97          if umax > 0:
98              for i in range(len(self.vertex_data)):
99                  for j in range(len(self.vertex_data)):
100                     if not self.conn_matrix[i][j] is None:
101                         self.conn_matrix[i][j].set_umax(umax)
102
103
104
105 def main():
106     # example usage for points connected with lines
107     g = Graph(0)
108     point1 = [0, 0, 0]    # A
109     point2 = [1, 1, 0]    # B
110     point3 = [1, 0.1, 0]    # C
111     point4 = [1, 2.5, 0]    # D
112     gpoints = [point1, point2, point3, point4]
113     gnames = ['A', 'B', 'C', 'D']
114     for p in gpoints:
115         g.add_vertex(gnames[gpoints.index(p)], p)
116     connections = [['A', 'B'], ['A', 'C'], ['B', 'C'], ['B', 'D'], [
    'C', 'D']]
117     for c in connections:
118         g.add_connection(c[0], c[1], Line(gpoints[gnames.index(c[0])
    ], gpoints[gnames.index(c[1])]))
119     print(g.vertex_locations)
120
121
122     distance, path = g.dijkstra('A', 'D')
123     print(f"Path: {path}, Distance: {distance}")
124     print(g.conn_matrix)
125
126
127 if __name__ == '__main__':
128     main()
```

****

Модуль draw routines.py

****

```
1
2  #  for plotting:
3  #  circle = circle_points(c, n)
4  #  plt.plot(circle[0], circle[1])
5  #
6  #  or
7  #  arc = arc_points(c, a, b, dir, n)
8  #  plt.plot(arc[0], arc[1])
9
10 import numpy as np
11 import matplotlib.pyplot as plt
12 import aux_classes as aux
13
14 def circle_points(c, n=50):
15     #  creates an array of points for drawing a circle.
16     #  c in format [x_c, y_c, r] - x,y coordinates and radius of the
       circle
17     #  n - number of points to draw
18     #  returns (n+1) points where the first and the last points are
       the same, for closed loop.
19     x = c[0]
20     y = c[1]
21     r = c[2]
22     points =np.empty((2, n + 1))
23     for i in range(n+1):
24         points[0, i] = x + r * np.cos(2 * i * np.pi / n)
25         points[1, i] = y + r * np.sin(2 * i * np.pi / n)
26     return points
27
28
29 def arc_points(c, a, b, dir, n=50):
30     #  creates an array of points for drawing an arc of circle c
       from ray ([x_c, y_c] -> a) to ray ([x_c, y_c] -> b)
31     #  c - circle in format [x_c, y_c, r]
32     #  a - starting point [x_a, y_a]
33     #  b - finishing point [x_b, y_b]
34     #  dir - direction of drawing (>0 for counter-clockwise, <0 for
       clockwise
35     #  n - number of points
36     x = c[0]
37     y = c[1]
38     r = c[2]
39     phi = np.atan2([a[1] - y, b[1] - y], [a[0] - x, b[0] - x])
```

```
40      if dir > 0 and phi[1] < phi[0]:
41          phi[1] += 2 * np.pi
42      if dir < 0 and phi[1] > phi[0]:
43          phi[1] -= 2 * np.pi
44      angles = np.linspace(phi[0], phi[1], n)
45      points = np.transpose([[x + r * np.cos(ang), y + r * np.sin(ang)
    ] for ang in angles])
46      return points
47
48
49  def draw_circles(circles, n=50):
50      #  prepares an array of points for plotting the array of circles
        <circles>
51      #  n - number of points for each circle to be plotted, default n
    =50
52      #  circles in format [circle1, circle2, ...]
53      #  where circle1 = [x1, y1, r1] etc.
54      #  to plot use plt.plot(points[0], points[1])
55      return np.transpose([circle_points(c, n) for c in circles], (1,
    2, 0))
56
57
58  def draw_lines(lines):
59      #  prepares an array of points for drawing the array of lines
60      #  lines in format [line1, line2, ...]
61      #  where line1 = [[x1, y1], [x2, y2]] etc.
62      #  to plot use plt.plot(points[0], points[1])
63      try:
64          if type(lines[0]) is aux.Line:   # for compatibility with
    aux.Line and other aux classes drawing
65              return np.transpose([[l.begin[0:2], l.end[0:2]] for l in
        lines], (2, 1, 0))
66          else:
67              return np.transpose(lines, (2, 1, 0))
68      except IndexError:
69          # return np.transpose(np.empty((0, 2, 3)), (2,1,0))
70          return np.empty((3, 2, 0))
71
72
73  def draw_arcs(arcs, n=50):
74      return np.transpose([arc_points(a.circle, a.begin, a.end, a.dir,
        n) for a in arcs], (1, 2, 0))
75
76
77  def draw_points(points):
78      return np.transpose(points, (1, 0))
```

```
79
80
81  def draw_rotations(rotations, l=1):
82      if len(rotations) == 0:
83          return np.empty((3, 2, 0))
84      else:
85          lines = []
86          for r in rotations:
87              p1 = r.begin + l * np.array([np.cos(r.begin[2]), np.sin(
        r.begin[2]), 0])
88              p2 = p1 + 0.15 * l * r.dir * np.array([-np.sin(r.begin
        [2]), np.cos(r.begin[2]), 0])
89              p3 = r.end + l * np.array([np.cos(r.end[2]), np.sin(r.
        end[2]), 0])
90              lines.append([r.begin, p1])
91              lines.append([p1, p2])
92              lines.append([r.end, p3])
93          lines = np.array(lines)
94          return np.transpose(lines, (2, 1, 0))
95
96
97  def main():
98      x1 = 0
99      y1 = 0
100     r1 = 1
101     x2 = 2
102     y2 = 1
103     r2 = 1.5
104     x3 = 1
105     y3 = 2
106     r3 = 2
107     p1 = [0, 1, 2]
108     p2 = [3, 4, 2]
109     p3 = [1, 2, 2]
110     p4 = [2, 0, 1]
111     lines = np.array([[p1, p2], [p3, p4]])
112     circles = np.array([[x1, y1, r1], [x2, r2, r2], [x3, y3, r3]])
113     arc1 = aux.Arc([1, 0, 2], [4, -4, 0], [-5, -5, 0], 0, 1)
114     arc2 = aux.Arc([3, 1, 1], [5, 0, 0], [-1, -10, 0], 0, 1)
115     rot1 = aux.Rotation([0, 1, 0.5], -np.pi/4)
116     rot2 = aux.Rotation([1, 1, -0.5], -np.pi/4)
117     phi1 = arc1.begin[2]
118     al1 = [arc1.begin, arc1.begin + np.array((np.cos(phi1), np.sin(
        phi1), 0))]
119     arc_lines = [[arc1.begin, arc1.begin + np.array([np.cos(arc1.
        begin[2]), np.sin(arc1.begin[2]), 0])],
```

```
                  [arc1.end, arc1.end + np.array([np.cos(arc1.end[2])
    , np.sin(arc1.end[2]), 0])],
                  [arc2.begin, arc2.begin + np.array([np.cos(arc2.
    begin[2]), np.sin(arc2.begin[2]), 0])],
                  [arc2.end, arc2.end + np.array([np.cos(arc2.end[2])
    , np.sin(arc2.end[2]), 0])]]
    points = draw_circles(circles, 50)
    points2 = draw_lines(lines)
    points3 = draw_arcs([arc1, arc2])
    points4 = draw_lines(arc_lines)
    points5 = draw_rotations([rot1, rot2])
    plt.figure(1)
    plt.gca().set_aspect('equal')
    plt.grid()
    plt.plot(points[0], points[1])
    plt.plot(points2[0],points2[1])
    plt.plot(points3[0], points3[1])
    plt.plot(points4[0], points4[1])
    plt.plot(points5[0], points5[1], 'k')
    plt.show()
    return


if __name__ == '__main__':
    main()
```

****

Модуль file routines.py

****

```

import tkinter as tk
from tkinter import filedialog
import numpy as np
from matplotlib import pyplot as plt
import draw_routines as draw
import geometry as geom
import aux_classes as aux
import designators


def get_setup_filename():
    root = tk.Tk()
    root.withdraw()
    filename = filedialog.askopenfilename(title='Open setup file',
    filetypes=[('Text files', '*.txt')])
    return filename
```

```
17
18
19 def parse_setup(filename):
20     start = np.empty(shape=(0,3))
21     have_start = False
22     end = np.empty(shape=(0,3))
23     have_end = False
24     circles = np.empty(shape=(0,3))
25     umax = None
26     have_umax = False
27     base = None
28     have_base = False
29     n_circles = 0
30     with open(filename, 'r') as fhandle:
31         lines = fhandle.readlines()
32     for line in lines:
33         datatype, data = parse_line(line)
34         if datatype == 'start':
35             if have_start:
36                 print('Warning: found more than one start point,
   ignoring all extra start points.')
37             else:
38                 if len(data) == 3:
39                     start = data
40                     have_start = True
41         elif datatype == 'end':
42             if have_end:
43                 print('Warning: found more than one end point,
   ignoring all extra end points.')
44             else:
45                 if len(data) == 3:
46                     end = data
47                     have_end = True
48         elif datatype == 'circle':
49             if len(data) == 3:
50                 circles = np.append(circles, [data], axis=0)
51         elif datatype == 'umax':
52             if have_umax:
53                 print('Warning: found more than one umax, ignoring
   all extra values.')
54             else:
55                 if len(data) == 1:
56                     umax = data
57                     have_umax = True
58         elif datatype == 'base':
59             if have_base:
```

```
60                  print('Warning: found more than one base, ignoring
    all extra values.')
61              else:
62                  if len(data) == 1:
63                      base = data
64                      have_base = True
65          else:
66              print(f'Warning: unidentified label {data} found in
    setup file.')
67      ol = geom.overlap(circles)
68      if len(ol) > 0:
69          raise ValueError(f'Error: overlapping circles detected: {ol
    }, check setup.')
70      if geom.is_inside(start, circles):
71          raise ValueError('Start point inside one of the circles,
    check setup')
72      if geom.is_inside(end, circles):
73          raise ValueError('End point inside one of the circles, check
     setup')
74      if umax is not None and umax <= 0:
75          raise ValueError('Error: umax in setup is nonpositive.')
76      if base is not None and base <= 0:
77          raise ValueError('Error: base in setup is nonpositive.')
78      return start, end, circles, umax, base
79
80
81  def find_occurrences(s, ch):
82      return [i for i, letter in enumerate(s) if letter == ch]
83
84
85  def parse_line(line):
86      lc = '('
87      rc = ')'
88      delimiter = ','
89      out_data = np.empty(shape=0)
90      pos_l = find_occurrences(line, lc)
91      pos_r = find_occurrences(line, rc)
92      if not len(pos_l) == 1:
93          raise ValueError(f'Error: expected 1 opening bracket, found
    {len(pos_l)}.')
94      if not len(pos_r) == 1:
95          raise ValueError(f'Error: expected 1 closing bracket, found
    {len(pos_r)}.')
96      pos_del = find_occurrences(line, delimiter)
97      datatype = "".join(line[0:pos_l[0]].split()).lower()   # remove
    all spaces in line and convert to lowercase
```

```
98      n_del = len(pos_del)
99      if (datatype == 'circle') or (datatype == 'start') or (datatype
        == 'end'):
100         if n_del == 2:
101             tmpstr = line[pos_1[0] + 1: pos_del[0]]
102             out_data = np.append(out_data, float(tmpstr))
103             tmpstr = line[pos_del[0] + 1: pos_del[1]]
104             out_data = np.append(out_data, float(tmpstr))
105             tmpstr = line[pos_del[1] + 1: pos_r[0]]
106             out_data = np.append(out_data, float(tmpstr))
107         else:
108             print(f'Error reading {datatype}: expected 2 delimiters,
        found {n_del}')
109     elif (datatype == 'umax'):
110         if n_del == 0:
111             tmpstr = line[pos_1[0] + 1: pos_r[0]]
112             val = None
113             try:
114                 val = float(tmpstr)
115             except ValueError:
116                 print(f'Warning: cannot convert {tmpstr} to float,
        ignoring input.')
117             if np.isreal(val):
118                 out_data = np.append(out_data, val)
119     elif (datatype == 'base'):
120         if n_del == 0:
121             tmpstr = line[pos_1[0] + 1: pos_r[0]]
122             val = None
123             try:
124                 val = float(tmpstr)
125             except ValueError:
126                 print(f'Warning: cannot convert {tmpstr} to float,
        ignoring input.')
127             if np.isreal(val):
128                 out_data = np.append(out_data, val)
129
130     return datatype, out_data
131
132
133 def report_setup():
134     pass
135
136
137 def generate_out_name(name):
138     dir_sep = find_occurrences(name, '/')
139     if len(dir_sep) == 0:
```

```
        pos_dir = 0
    else:
        pos_dir = dir_sep[-1]
    dot_sep = find_occurrences(name, '.')
    print(dot_sep)
    if len(dot_sep) == 0:
        pos_dot = -1
    else:
        pos_dot = dot_sep[-1]
    thename = name[pos_dir+1:pos_dot]   # file name without slash and extension
    path = name[0: pos_dir]   # no slash here, extracting path
    # print(path)
    out_name = path + '/' + 'report_' + thename + '.txt'
    print(out_name)
    return out_name


def save_report(filename, g, path, mode, distance):
    short_strings = []
    full_strings = []
    p_start = designators.d_start
    p_finish = designators.d_finish
    gnames = g.vertex_data
    modes = {'t': 'Time', 'd': 'Length'}
    for i in range(len(path) - 1):
        conn = g.conn_matrix[gnames.index(path[i]), gnames.index(path[i + 1])]
        results = {'t': conn.get_time(), 'd': conn.get_length()}
        if type(conn) is aux.Line:
            short_strings.append(f'Line({conn.begin}, {conn.end}): {modes[mode]}={results[mode]}\n')
        elif type(conn) is aux.Arc:
            short_strings.append(f'Arc({conn.circle}, {conn.begin}, {conn.end}, {conn.dir}): '
                                 f'{modes[mode]}={results[mode]}\n')
        elif type(conn) is aux.Rotation:
            short_strings.append(f'Rotation({conn.begin}, {conn.phi}, {conn.dir}): {modes[mode]}={results[mode]}\n')
    short_strings.append(f'Path: {path}\n')
    short_strings.append(f'Overall {modes[mode].lower()}: {distance}\n')

    with open(filename, 'wt') as fhandle:
        for s in short_strings:
            fhandle.write(s)
```

```
180
181
182 def main():
183     start, end, circles, umax, base = parse_setup(get_setup_filename
    ())
184
185     print('Start = ', start)
186     print('End = ', end)
187     print('Circles: ', circles)
188     print('Umax = ', umax)
189     print('Base = ', base)
190
191
192     # drawing
193     plt.figure(1)
194     dcircles = draw.draw_circles(circles)
195     plt.plot(dcircles[0], dcircles[1])
196     dpoints = draw.draw_points([start, end, [2.5, 2.5, 1]])
197     plt.plot(dpoints[0], dpoints[1], 'b.')
198     plt.grid()
199     plt.gca().set_aspect('equal')
200     plt.show()
201
202
203 if __name__ == '__main__':
204     main()
```

****

Модуль geometry.py

****

```
1
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import draw_routines as draw
5 from aux_classes import normalize_angle
6
7
8 def add_angle(a, b):
9     #   adding direction angle to both points
10    phi = np.atan2(b[1] - a[1], b[0] - a[0])
11    a[2] = normalize_angle(phi)
12    # b[2] = normalize_angle(phi + np.pi)
13    b[2] = normalize_angle(phi)
14
15
16 def tangent_c2c(circle1, circle2):
```

```
    x1 = np.float64(circle1[0])
    y1 = np.float64(circle1[1])
    r1 = np.float64(circle1[2])
    x2 = np.float64(circle2[0])
    y2 = np.float64(circle2[1])
    r2 = np.float64(circle2[2])
    c1 = np.array([x1, y1, 0])
    c2 = np.array([x2, y2, 0])
    d = np.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2))
    #  if negative or zero radius in arguments
    if r1 <= 0 or r2 <= 0:
        return 0, np.array([])

    #  variant 1 - one circle inside the other - no tangents
    if d < np.fabs(r1 - r2):
        return 0, np.array([])

    #  variant 2 - tangent circles - inner touch and outer touch, 1 tangent, point b taken as unit segment
    #  from tangent point leftwards along the common tangent, looking from circle 1
    if d == np.fabs(r1 - r2):
        e1 = (c2 - c1) / d
        # phi = np.atan2(e1[0], -e1[1])
        a1 = r1 * e1 + c1
        b1 = a1 + np.array([-e1[1], e1[0], 0])   # the stub is made to produce a line even in there is no line
        add_angle(a1, b1)
        return 1, np.array([[a1, b1]])

    # variant 3 - outer touch - 3 tangents, 2 external and 1 common
    # !!! check again all variants, variant 3 not tested
    if d == r1 + r2:
        cphi = (r1 - r2) / d
        sphi = np.sqrt(1 - cphi * cphi)
        matr = np.array([[cphi, -sphi, 0], [sphi, cphi, 0], [0, 0, 1]])
        e1 = (c2 - c1) / d
        a1 = matr.dot(e1) * r1 + c1
        b1 = matr.dot(e1) * r2 + c2
        add_angle(a1, b1)
        a2 = matr.transpose().dot(e1) * r1 + c1
        b2 = matr.transpose().dot(e1) * r2 + c2
        add_angle(a2, b2)
        a3 = r1 * e1 + c1
        b3 = a3 + np.array([-e1[1], e1[0], 0])  # the stub is made
```

```
    to produce a line even in there is no line
59          add_angle(a3, b3)
60          # order: 1) external left to left, 2) external right to
    right, 3) common (leftwards looking from circle 1)
61          return 3, np.array([[a1, b1], [a2, b2], [a3, b3]])
62
63      #  variant 3 - intersecting circles, 2 common tangents
64      if d < (r1 + r2):
65          cphi = (r1 - r2) / d
66          sphi = np.sqrt(1 - cphi * cphi)
67          matr = np.array([[cphi, -sphi, 0], [sphi, cphi, 0], [0, 0,
    1]])
68          e1 = (c2 - c1) / d
69          a1 = matr.dot(e1) * r1 + c1
70          b1 = matr.dot(e1) * r2 + c2
71          add_angle(a1, b1)
72          a2 = matr.transpose().dot(e1) * r1 + c1
73          b2 = matr.transpose().dot(e1) * r2 + c2
74          add_angle(a2, b2)
75          #  order : 1) outer left,  2) outer right
76          return 2, np.array([[a1, b1], [a2, b2]])
77
78      #  variant 4 (default) - distant circles, 4 common tangents (2
    outer, 2 inner)
79      if d > (r1 + r2):
80          cphi1 = (r1 - r2) / d  # for outer tangents
81          sphi1 = np.sqrt(1 - cphi1 * cphi1)
82          cphi2 = (r1 + r2) / d  # for inner tangents
83          sphi2 = np.sqrt(1 - cphi2 * cphi2)
84          matr1 = np.array([[cphi1, -sphi1, 0], [sphi1, cphi1, 0], [0,
     0, 1]])
85          matr2 = np.array([[cphi2, -sphi2, 0], [sphi2, cphi2, 0], [0,
     0, 1]])
86          e1 = (c2 - c1) / d
87          #  outer tangent - left to left
88          a1 = matr1.dot(e1) * r1 + c1
89          b1 = matr1.dot(e1) * r2 + c2
90          add_angle(a1, b1)
91          #  outer tangent - right to right
92          a2 = matr1.transpose().dot(e1) * r1 + c1
93          b2 = matr1.transpose().dot(e1) * r2 + c2
94          add_angle(a2, b2)
95          #  inner tangent - left to right
96          a3 = matr2.dot(e1) * r1 + c1
97          b3 = matr2.dot(-e1) * r2 + c2
98          add_angle(a3, b3)
```

```
99          #  inner tangent - right to left
100         a4 = matr2.transpose().dot(e1) * r1 + c1
101         b4 = matr2.transpose().dot(-e1) * r2 + c2
102         add_angle(a4, b4)
103         #  order : 1) outer left,  2) outer right,  3) inner left to
        right,  4) inner right to left
104         return 4, np.array([[a1, b1], [a2, b2], [a3, b3], [a4, b4]])
105     else:
106         return 0, np.array([])
107
108
109 def tangent_p2c(p, c):
110     xp = np.float64(p[0])
111     yp = np.float64(p[1])
112     xc = np.float64(c[0])
113     yc = np.float64(c[1])
114     rc = np.float64(c[2])
115     p1 = np.array([xp, yp, 0])
116     c1 = np.array([xc, yc, 0])
117     d = np.sqrt((xp - xc) * (xp - xc) + (yp - yc) * (yp - yc))
118     if rc <= 0:
119         return 0, np.array([])
120     elif d < rc:
121         return 0, np.array([])
122     elif d == rc:
123         e1 = (p1 - c1) / d
124         # phi = np.atan2(e1[0], -e1[1])
125         a1 = p1
126         # a1[2] = phi
127         b1 = p1 + np.array([e1[1], -e1[0], 0])   # the stub is made
        to produce a line even in there is no line
128         # b1[2] = phi
129         add_angle(a1, b1)
130         return 1, np.array([[a1, b1]])
131     else:
132         e1 = (p1 - c1) / d
133         cphi = rc / d
134         sphi = np.sqrt(1 - cphi * cphi)
135         matr = np.array([[cphi, sphi, 0], [-sphi, cphi, 0], [0, 0,
        1]])
136         a1 = p1.copy()
137         a2 = p1.copy()
138         b1 = c1 + matr.dot(e1) * rc
139         b2 = c1 + matr.transpose().dot(e1) * rc
140         add_angle(a1, b1)
141         add_angle(a2, b2)
```

```
142         #  order:  1) left, 2) right
143         return 2, np.array([[a1, b1], [a2, b2]])
144 
145 
146 def tangent_c2p(c, p):
147     n, res = tangent_p2c(p, c)
148     if n == 0:
149         return 0, np.array([])
150     elif n == 1:
151         q = res[0]
152         return 1, np.array([[q[0], q[0] + (q[0] - q[1])]])
153     elif n == 2:
154         q1 = res[0]
155         q2 = res[1]
156         for p in q1:
157             p[2] = normalize_angle(p[2] - np.pi)
158         for p in q2:
159             p[2] = normalize_angle(p[2] - np.pi)
160         return 2, np.array([[q2[1], q2[0]], [q1[1], q1[0]]])
161     else:
162         return 0, np.array([])
163 
164 
165 def is_obstructed_by(l, c):
166     a = np.float64(l[0][0:2])
167     b = np.float64(l[1][0:2])
168     center = np.array([c[0], c[1]])
169     rc = c[2]
170     el = (b - a) / np.linalg.norm(b - a)
171     eort = np.array((-el[1], el[0]))
172     d = np.fabs(eort.dot(center - a))   # distance from circle
    center to the line
173     between = el.dot(center - a) * el.dot(center - b)   # negative
    means cen is projected between a and b
174     if d < rc and between < 0:
175         return True
176     else:
177         return False
178 
179 
180 def is_obstructed(l, circles):
181     res = False
182     for c in circles:
183         if is_obstructed_by(l, c):
184             res = True
185             break
```

```
186     return res
187
188
189 def is_overlapped_by(circle1, circle2):
190     x1 = circle1[0]
191     y1 = circle1[1]
192     r1 = circle1[2]
193     x2 = circle2[0]
194     y2 = circle2[1]
195     r2 = circle2[2]
196
197     d2 = (x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1)
198     if d2 <= (r1 + r2) * (r1 + r2):   # change to < if allowing
        touch
199         return True
200     else:
201         return False
202
203
204 def overlap(circles):
205     res = []
206     for i in range(len(circles)):
207         for j in range(i+1, len(circles)):
208             if is_overlapped_by(circles[i], circles[j]):
209                 res = (i,j)
210                 break
211     return res
212
213
214 def is_inside_of(point, circle):
215     xp = point[0]
216     yp = point[1]
217     xc = circle[0]
218     yc = circle[1]
219     rc = circle[2]
220     d2 = (xc - xp) * (xc - xp) + (yc - yp) * (yc - yp)
221     if d2 < rc * rc:
222         return True
223     else:
224         return False
225
226
227 def is_inside(point, circles):
228     res = False
229     for c in circles:
230         if is_inside_of(point, c):
```

```
231             res = True
232             break
233     return res
234 
235 
236 def main():
237     x1 = 0
238     x2 = 4
239     y1 = 0
240     y2 = 3
241     r1 = 3
242     r2 = 1
243     xa = 0
244     ya = 0
245     ra = 3
246     xp = 5
247     yp = 2
248     a_start = np.array((1, 0))
249     b_finish = np.array((1, -1))
250     c1 = np.array([x1, y1, r1])
251     c2 = np.array([x2, y2, r2])
252     p1 = np.array([xp, yp, 0])
253 
254     n, res = tangent_c2c(c1, c2)
255     lines = np.empty((n, 2, 3))
256     for i in range(n):
257         lines[i] = res[i]
258     # lines = [res[2]]
259 
260     # print(res)
261 
262     n2, res2 = tangent_p2c(p1, c1)
263     lines2 = np.empty((n2, 2, 3))
264     for i in range(n2):
265         lines2[i] = res2[i]
266     # print(res2)
267 
268     n3, res3 = tangent_c2p(c2, p1)
269     # n3, res3 = tangent_p2c(p1, c2)
270     lines3 = np.empty((n3, 2, 3))
271     for i in range(n3):
272         lines3[i] = res3[i]
273 
274     print(res3)
275 
276     # drawing
```

```
277     plt.figure(1)
278     circles = draw.draw_circles([c1, c2])
279     plt.plot(circles[0], circles[1])
280 
281     lines_prepared = draw.draw_lines(lines)
282     lines2_prepared = draw.draw_lines(lines2)
283     lines3_prepared = draw.draw_lines(lines3)
284     plt.plot(lines_prepared[0], lines_prepared[1], 'b.')   #  adding
        end points
285     plt.plot(lines_prepared[0], lines_prepared[1], 'b')    #
       drawing the lines from array
286     plt.plot(lines2_prepared[0], lines2_prepared[1], 'r')
287     plt.plot(lines2_prepared[0], lines2_prepared[1], 'r.')
288     plt.plot(lines3_prepared[0], lines3_prepared[1], 'g')
289     plt.plot(lines3_prepared[0], lines3_prepared[1], 'g.')
290     plt.grid()
291     plt.gca().set_aspect('equal')
292     plt.show()
293 
294 
295 if __name__ == '__main__':
296     main()
```

****

Пример входного файла:

****

```
1 
2 start(0,0,0)
3 circle(1.5,1.5,0.8)
4 circle(3.5,3.5,0.8)
5 circle(1,4,1.5)
6 circle(4,1,1.5)
7 end(5,5,0)
8 umax(1)
9 base(1)
```

****

Приложение С

Листинг программы к главе 4

С.1 Листинг программы для решения задачи оптимального управления методом ASLAM-MPPI

Основной модуль: поиск управления

```
1
2 import numpy as np
3
4
5 def calc_softmax_seq(batch_costs: np.array, control_batch: np.
   ndarray, temperature: float):
6     """Calculate control using softmax function.
7
8     Args:
9         control_batch: np.ndarray of shape [batch_size, time_steps,
   2]
10         costs: np.array of shape [batch_size]
11     """
12
13     batch_costs = batch_costs - np.min(batch_costs)
14     exponents = np.exp(-1 / temperature * (batch_costs))
15     softmaxes = exponents / np.sum(exponents)  # -> shape = [
   batch_size]
16
17     control = (control_batch * softmaxes[:, np.newaxis, np.newaxis])
   .sum(axis=0)
18
19     return control
20
21
```

```
def find_min_seq(batch_costs: np.array, control_batch: np.ndarray):
    best_idx = np.argmin(batch_costs, axis=0)
    return control_batch[best_idx]
```

Модуль нахождения функции стоимости

```

import numpy as np
from numba import njit


def triangle_cost(state, reference_trajectory, reference_intervals, obstacles, weights, powers):
    """Cost according to nearest segment.

    Args:
        state: np.ndarray of shape [batch_size, time_steps, 8] where 3 for x, y, yaw, v, w, v_control, w_control, dts
        ref_traj: np.array of shape [ref_traj_size, 3] where 3 for x, y, yaw
        desired_v: float
        obstacles: list of points [x, y]
        weights: weights for cost components
        powers: powers for cost components

    Return:
        costs: np.array of shape [batch_size]
    """


    costs = np.zeros(shape=state.shape[0])
    obstacles_c = obstacles_cost(state, obstacles, eps=0.15,
                                 weight=weights['obstacle'],
                                 power=powers['obstacle'])

    triangle_c = triangle_cost_segments(state, reference_trajectory, reference_intervals,
                                        weight=weights['reference'],
                                        power=powers['reference'])

    goal_c = goal_cost(state, reference_trajectory[-1],
                       weight=weights['goal'],
                       power=powers['goal'])

    costs += triangle_c + goal_c + obstacles_c
    return costs

```

```
38
39 @njit
40 def triangle_cost_segments(state, reference_trajectory,
   reference_intervals, weight=1, power=1):
41
42     x_dists = state[:, :, :1] - reference_trajectory[:, 0]
43     y_dists = state[:, :, 1:2] - reference_trajectory[:, 1]
44     dists = np.sqrt(x_dists ** 2 + y_dists ** 2)
45
46     if len(reference_trajectory) == 1:
47         return dists.sum(2).sum(1)
48
49     first_sides = dists[:, :, :-1]
50     second_sides = dists[:, :, 1:]
51     opposite_sides = reference_intervals
52
53     first_obtuse_mask = is_angle_obtuse(first_sides, second_sides,
   opposite_sides)
54     second_obtuse_mask = is_angle_obtuse(second_sides, first_sides,
   opposite_sides)
55
56     cost = np.zeros(shape=(dists.shape[0]))
57     dists_to_segments = np.empty(len(reference_intervals))
58     for i in range(dists.shape[0]):
59         for j in range(dists.shape[1]):
60             for k in range(len(reference_intervals)):
61
62                 first_side = first_sides[i, j, k]
63                 second_side = second_sides[i, j, k]
64                 opposite_side = opposite_sides[k]
65
66                 if is_angle_obtuse(first_side, second_side,
   opposite_side):
67                     dists_to_segments[k] = first_side
68
69                 elif is_angle_obtuse(second_side, first_side,
   opposite_side):
70                     dists_to_segments[k] = second_side
71                 else:
72                     dists_to_segments[k] = heron(
73                         opposite_side,
74                         first_side,
75                         second_side
76                     )
77             cost[i] += dists_to_segments.min()
78         cost[i] /= dists.shape[1]
```

```
79
80      return (cost*weight)**power
81
82
83  @njit
84  def is_angle_obtuse(opposite_side, b, c):
85      return opposite_side ** 2 > (b ** 2 + c ** 2)
86
87  @njit
88  def heron(opposite_side, b, c):
89      eps = 0.000001
90      if abs(opposite_side) < eps:
91          return min(b, c)
92
93      p = (opposite_side + b + c) / 2.0
94      h = 2.0 / opposite_side * np.sqrt(p * (p - opposite_side) * (p -
        b) * (p - c))
95      return h
96
97
98  def obstacles_cost(state, obstacles, eps, weight, power):
99      costs = np.zeros(shape=(state.shape[0]))
100     if not len(obstacles):
101         return costs
102
103     x, y, r = obstacles[:, 0], obstacles[:, 1], obstacles[:, 2]
104
105     x_dists = state[:, :, :1] - x
106     y_dists = state[:, :, 1:2] - y
107     dists = np.sqrt(x_dists ** 2 + y_dists ** 2) - r
108     dists = dists.min(2).min(1)
109
110     dists[dists < 0] = 0
111
112     # Points close to the obstacle
113     approx_mask = (dists < eps)
114     costs[approx_mask] = ((eps - dists[approx_mask]) * weight) **
       power
115
116     return costs
117
118
119 @njit
120 def goal_cost(state, goal, weight=1, power=1):
121     x_dists = state[:, :, 0] - goal[0]
122     y_dists = state[:, :, 1] - goal[1]
```

```
123     dists = np.sqrt(x_dists ** 2 + y_dists ** 2)
124 
125     return weight * (dists[:, -1] ** power)
```

Модуль установки метрик

```

import numpy as np


def mean_dist_metric(ref_traj, path_points):
    error = 0
    for ref in ref_traj[:, :3]:
        min_dist = __find_min_dist(ref, path_points[:, :3])
        error += min_dist

    return error / len(ref_traj)


def __find_min_dist(ref_pt, path_points):
    min_dist = np.inf
    for pt in path_points:
        curr_dist = np.linalg.norm(pt - ref_pt)
        if min_dist >= curr_dist:
            min_dist = curr_dist

    return min_dist
```