Санкт-Петербургский государственный университет

На правах рукописи

Попков Александр Сергеевич

Оптимальное позиционное управление в нелинейных управляемых системах

Научная специальность 2.3.1.
Системный анализ, управление и обработка информации, статистика

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
доктор физико-математических наук, профессор
Смирнов Николай Васильевич

Санкт-Петербург — 2022

Оглавление

ВВЕДЕНИЕ

Актуальность темы. В диссертационной работе рассматривается задача построения оптимального управления, переводящего объект из начального состояния на некоторую плоскость. Поиск управлений осуществляется в классе кусочно-постоянных функций. Для описания динамического процесса используется система обыкновенных дифференциальных уравнений. Качество управления определяется значением линейного или квадратичного целевого функционала.

До начала динамического процесса определяется функция управления (программное управление), вычисляется траектория движения объекта. В следствие различных внутренних и внешних факторов, в процессе движения объект может отклониться от исходной траектории. Тогда появляется смысл перестроения управления в режиме реального времени, исходя из текущей позиции. Такое управление называется позиционным.

Подобные задачи имеют широкое распространение в различных сферах. Например, автором работы представленные алгоритмы были испытаны на следующих моделях:

- Управление вращательным движением вала электродвигателя.
- Перемещение беспилотной тележки.
- Распределение инвестиций в отрасли экономики.
- Стабилизация движений квадрокоптера.
- Демпфирование колебаний искусственного спутника.
- Управление математическим маятником.
- Демпфирования колебаний «спящего волчка» Лагранжа.

Многие процессы управления техническими объектами имеют объективно нелинейный характер, следовательно для их точного моделирования и, в последствии, управления требуется аппарат построения управления для нелинейных систем.

В теории обыкновенных дифференциальных уравнений известно аналитическое решение задачи Коши для линейных систем любой размерности: выведена формула Коши. Это привело к развитию методов решения линейных задач управления. Однако, аналитические решения для нелинейных систем дифференциальных уравнений существуют лишь для очень узких видов нелинейностей, что делает невозможным изобретение точных методов нахождения управления для систем с нелинейной правой частью. В данной работе разрабатывается метод численного построения управления для нелинейных по фазовым переменным систем при помощи итеративной линеаризации правой части системы и нахождения оптимального управления для линейной системы.

Также об актуальности задачи говорит публикационная активность по данной тематике как в отечественных, так и зарубежных изданиях в последние десятилетия. Кроме того, активно разрабатываются и улучшаются программные комплексы для подобных задач.

Цели и задачи. Основной целью работы является разработка численного метода поиска приближенного решения в нелинейной задаче оптимального управления. Под нелинейной задачей понимается управление в системе обыкновенных дифференциальных уравнений, содержащей нелинейности по фазовым переменным. Необходимо построить алгоритмы построения программных и позиционных управлений. Программное управление вычисляется перед началом движения, затем оно пересчитывается исходя из текущего положения с учетом внутренних и внешних возмущений. Предполагается, что управление корректируется спустя заданный промежуток времени с последней корректировки, при этом движение управляемого объекта не прекращается, что позволяет говорить об управлении в режиме реального времени.

Для достижения цели необходимо решить ряд задач:

1. Изучить современные научные наработки в данном направлении, определить перспективные направления.

2. Исследовать множество достижимости и множество управляемости в линейной задаче управления при выборе управления в классе кусочно-постоянных функций. Информация об этих множествах имеет важное значение при решении нелинейной задачи, поскольку она так или иначе сводится к линейной.
3. Разработать метод решения задачи с линейной системой и выбором управления в классе кусочно-линейных и кусочно-квадратичных функций. Рассмотрение таких видов управления позволяет находить решения с лучшим значением целевой функции и накладывать ограничения по гладкости на искомое управление.
4. Проработать подход к решению линейной задачи с дополнительными выпуклыми и невыпуклыми ограничениями на управления и фазовые переменные. Задание ограничений такого типа позволяет рассматривать более специфичные классы задач.
5. Разработать численный алгоритм построения программных и позиционных управления в нелинейной задаче при выборе управления в классе кусочно-постоянных функций.
6. Исследовать теоретические свойства алгоритмов.
7. Реализовать алгоритм построения управлений в нелинейной задаче в виде набора функций на языке Python. Апробировать алгоритм на различных примерах и сценариях.

Основные положения, выносимые на защиту

1. Метод построения множества достижимости и управляемости для линейной задачи в классе кусочно-постоянных управлений при наличии двусторонних ограничений, строгое теоретическое обоснование метода.
2. Алгоритмы сведения задачи оптимального управления к задаче линейного программирования для кусочно-линейных и кусочно-квадратичных классов управлений.

3. Алгоритмы сведения задачи оптимального управления с кусочно-постоянным управлением и линейным (квадратичным) функционалом при наличии дополнительных выпуклых и невыпуклых ограничений к задаче частично целочисленного линейного (квадратичного) программирования.
4. Алгоритмы построения программных и позиционных управлений для нелинейной задачи оптимального управления с кусочно-постоянным управлением.
5. Программная реализация представленных алгоритмов в виде комплекса программ.

Методология и методы исследования. Основная идея решения задач оптимального управления в различных постановках состоит в сведении к задачам математического программирования. После формирования методов построения управлений проводится исследование теоретических свойств решений и испытание алгоритмов на тестовых примерах. Используются методы линейной алгебры, математического анализа, дифференциальных уравнений, теории управления, линейного и квадратичного программирования.

Научная и практическая ценность работы состоит в предложенных конструктивных алгоритмах решения поставленных задач. Представлены методы построения управления для нелинейной задачи в программном и позиционном режимах, которые апробированы на множестве примеров и могут быть использованы для управления конкретными устройствами.

Исходная задача может быть расширена рассмотрением дополнительных классов управляющих функций и дополнительных ограничений на управления и фазовые переменные. Рассмотрение задачи в таких постановках имеет практический смысл, поскольку зачастую прикладные модели содержат специфичные ограничения или специальные требования к решению.

Также практическую значимость представляет алгоритм построения множеств достижимости и управляемости. В различных прикладных задачах важной информацией считаются сведения о всевозможных будущих состояниях объекта. Предложенный алгоритм является конструктивным и может быть легко реализован для прикладной задачи. Кроме того, доказательство, что такой алгоритм позволяет сформировать точные множества, имеет собственную существенную научную ценность.

Наконец, реализованные алгоритмы сопровождаются программным кодом, который может быть использован как основа для создания импортонезависимых программных продуктов, позволяющих решать задачи оптимального управления. Такая задача представляется важной и актуальной с практической точки зрения.

Научная новизна

1. Разработан алгоритм построения управления в программном и позиционном режимах для задачи оптимального управления с нелинейной системой, терминальными условиями и ограничениями на управление. Суть алгоритма состоит в последовательной линеаризации нелинейной функции вдоль траектории движения и последующим вычислением новой траектории с управлением, являющимся оптимальным решением линеаризованной задачи.
2. Для линейной задачи, в которой управление выбирается в классе кусочно-постоянных функций и имеет двусторонние ограничения, предложен метод построения множеств достижимости и управляемости в виде системы линейных алгебраических неравенств. Показано, что данные множества являются многогранниками в пространстве фазовых переменных, приведено теоретическое обоснование алгоритма.
3. Разработан метод нахождения оптимального управления для линейной задачи с кусочно-квадратичным управлением, терминальными условиями и прямыми ограничениями на управления.

Ключевая особенность метода состоит в сведении исходной задачи к выпуклой задаче программирования с квадратичными ограничениями. Алгоритмы решения задач такого класса имеют куда меньшую сложность, чем алгоритмы для невыпуклых задач.

4. Предложен алгоритм построения оптимального управления для линейной задачи с кусочно-постоянным управлением при наличии выпуклых и невыпуклых ограничений на фазовые переменные (в заданные моменты времени) и на управления. Невыпуклые ограничения задаются как совокупность групп неравенств, в каждой момент времени должны выполняться ограничения хотя бы одной группы. Показана возможность сведения задачи в такой постановке к задаче частично целочисленного линейного или квадратичного программирования.

Обоснованность и достоверность обеспечивается корректностью постановок задач, полученных из литературы и в ходе семинаров на факультете прикладной математики — процессов управления Санкт-Петербургского государственного университета. Изложенные результаты были апробированы на множестве конференций, а публикации, послужившие основой диссертационной работы, прошли рецензирование и были опубликованы российских и международных изданиях. Все предложенные алгоритмы были запрограммированы и протестированы на различных задачах.

Личный вклад автора. Диссертация является самостоятельным трудом автора. Основные положения, выносимые на защиту, представляют личный вклада автора. Все результаты, изложенные в диссертации, были достигнуты автором работы, за исключением тех мест, где это оговорено явно и указана ссылка на первоисточник. Большая часть результатов основана на публикациях автора в научных изданиях. Программный код, представленный в приложении, был реализован автором.

Апробация работы. Результаты, изложенные в данной диссертации, докладывались и обсуждались на конференциях:

1. XLV международная научная конференция аспирантов и студентов «Процессы управления и устойчивость» (CPS'14), 1–4 апреля 2014, Санкт-Петербург, Россия.
2. Международная конференция по компьютерным технологиям в физических и инженерных приложениях (ICCTPEA–2014), 30 июня – 4 июля 2014, Санкт-Петербург, Россия.
3. XLVI международная научная конференция аспирантов и студентов «Процессы управления и устойчивость» (CPS'15), 6–9 апреля 2015, Санкт-Петербург, Россия.
4. III международная конференция «Устойчивость и процессы управления» (SCP 2015), посвященной 85-летию со дня рождения профессора, чл.-корр. РАН В. И. Зубова, 5–9 октября 2015, Санкт-Петербург, Россия.
5. International Workshop on Applications in Information Technology (IWAIT–2018), 8–10 октября 2015, Аидзувакамацу, Япония.
6. XLVII международная научная конференция аспирантов и студентов «Процессы управления и устойчивость» (CPS'16), 4–7 апреля 2016, Санкт-Петербург, Россия.
7. XIII Международная конференция «Устойчивость и колебания нелинейных систем управления» (Конференция Пятницкого), 1–3 июня 2016, Москва, Россия.
8. 3rd International Conference on Applications in Information Technology (ICAIT–2018), 1–3 ноября 2018, Аидзувакамацу, Япония.
9. IV международная конференция «Устойчивость и процессы управления» (SCP 2020), посвященная 90-летию со дня рождения профессора, чл.-корр. РАН В. И. Зубова, 5–9 октября 2020, Санкт-Петербург, Россия.

Публикации. Результаты по теме диссертации изложены в одиннадцати научных публикациях [1–11], из которых две статьи опубликованы в журналах, включенных в перечень изданий ВАК [10;11], а семь работ — в изданиях, индексируемых в базах Scopus и Web of Science [2;4;7;8;10;11]. Также была зарегистрирована программа для ЭВМ «AdaptCopter» [12].

Поддержка. Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект № 19-31-90033).

Объем и структура работы. Диссертация состоит из введения, четырех глав, заключения и двух приложенияий. Полный объём диссертации составляет 148 страниц, включая 11 рисунков и 4 таблицы. Список литературы содержит 92 наименования.

Краткое содержание работы. Во **введении** приведена краткая постановка задачи, обоснована актуальность, сформированы цели и задачи диссертации, представлены основные положения, выносимые на защиту, описана научная и практическая ценность и научная новизна работы. Далее приведена информация об апробации работы, основных публикациях и финансовой поддержке, представлено краткое содержание работы.

В первой главе рассматривается задача оптимального управления с линейной системой. Задача терминального управления состоит в переводе управляемого объекта из начального положения на некоторую плоскость за заданное время. Управление выбирается из класса кусочно-постоянных функций с фиксированным периодом дискретизации. Заданы три целевых критерия, определяющих качество решения задачи. Первый функционал является линейным, второй может быть сведен к линейному, третий — квадратичный. Для каждой конкретной задачи выбирается один из этих функционалов или их линейная комбинация.

Описан метод решения линейной задачи, состоящий в переходе к задаче линейного или квадратичного программирования с последующим решением полученной оптимизационной задачи. Также в этой главе при-

веден обзор методов решения задач математического программирования и сравнительный анализ ПО для решения таких задач.

Несмотря на то, что рассматриваемая модель хорошо изучена литературе, глава 1 имеет важное значение в общей структуре работы, поскольку полученные результаты являются основой, на которую делается опора в последующих главах.

Во второй главе исследуется вопрос построения множеств достижимости и управляемости для линейной модели из главы 1. Показано, что задача построения этих множеств сводится к задаче линейного отображения многомерного куба из одного пространства в другое. В результате исследования свойств множеств доказано, что они являются выпуклыми многогранниками и могут быть описаны системой линейных неравенств. Основным результатом главы является конструктивный алгоритм построения множеств и его строгое доказательство. Проведен анализ сложности алгоритма.

В третьей главе рассматривается линейная задача с дополнительными условиями и ограничениями. В первой части главы изучается вопрос поиска оптимального решения при выборе управления в альтернативных классах функций. Показано, что линейная задача с кусочно-линейным управлением и линейным (квадратичным) функционалом сводится задаче линейного (квадратичного) программирования. В случае кусочно-квадратичного управления продемонстрирован переход к выпуклой задаче программирования с квадратичными ограничениями.

Во второй части главы базовая линейная модель дополняется выпуклыми и невыпуклыми линейными ограничениями на фазовые переменные в узловых точках и на управления. Для выпуклых ограничений приведен алгоритм сведения к задаче линейного (квадаратичного) программирования, а для модели с невыпуклыми ограничениями — к задаче частично целочисленного линейного (квадратичного) программирования.

В четвертой главе представлен алгоритм построения управления в программном и позиционном режимах для нелинейной задачи оптимального управления. Алгоритм вычисления программного управления состоит в итеративном пересчете управления на основе траектории объекта, найденной на предыдущем шаге. Находится линейное приближение нелинейной функции вдоль этой траектории, затем определяется обновленное управление как оптимальное решение линеаризованной задачи. После этого вычисляется решение задачи Коши исходной системы, замкнутой найденным управлением. Полученное решение является новой траекторией движения объекта.

Далее описывается алгоритм построения позиционного управления. В узловых точках управление пересчитывается, исходя из текущей позиции или прогноза позиции объекта. Для пересчета используется алгоритм программного режима, но дополнительное внимание уделяется времени вычисления.

Изучены теоретические свойства алгоритмов, выведены условия сходимости к допустимому решению исходной задачи. Работа алгоритмов продемонстрирована на двух тестовых задачах.

В заключении перечислены достигнутые результаты и приведена оценка дальнейших перспектив развития.

В приложении А описана структура программного комплекса для решения линейной задачи оптимального управления при выборе управления в классе кусочно-постоянных, кусочно-линейных или кусочно-квадратичных функций.

В приложении Б описана структура программного комплекса для решения нелинейной задачи с кусочно-постоянным управлением в программном и позиционном режимах.

ГЛАВА 1. ЛИНЕЙНАЯ ЗАДАЧА ОПТИМАЛЬНОГО УПРАВЛЕНИЯ

Первая глава является вводной. В ней рассматривается линейная задача оптимального управления и дается необходимая терминология. В параграфе 1.1 приводится постановка задачи в обобщенной форме. В параграфе 1.2 производится обзор литературы по данной тематике, обозначается место текущей работы. В параграфе 1.3 описывается алгоритм сведения исходной задачи к задаче линейного программирования. В параграфе 1.4 приведены рассуждения о методах решения задач линейного программирования и применении программных пакетов для этих целей.

1.1 Постановка задачи

1.1.1 Динамика объекта

Рассмотрим линейную задачу управления. Предположим, что движение управляемого объекта описывается системой обыкновенных дифференциальных уравнений:

$$\dot{x} = A(t)x + B(t)u + d(t). \tag{1.1}$$

Здесь

- t — независимая переменная (время), принимающая значения на отрезке от 0 до T, где T — гиперпараметр модели.
- $x = x(t)$ — n-мерная вектор-функция фазовых переменных. Компоненты $x_j(t)$ непрерывны при $t \in [0, T]$.

- $\dot{x}$ — вектор производных от фазовых переменных по времени. Под производной здесь и далее понимается производная слева.
- $u = u(t)$ — r-мерная вектор-функция управлений. Компоненты $u_i(t)$ непрерывны слева.
- $A(t)$ — матричная функция от t размерности $n \times n$.
- $B(t)$ — матричная функция от t размерности $n \times r$, состоящая из столбцов $b_i(t)$.
- $d(t)$ — n-мерная вектор-функция.

Кроме того, предполагается, что функции $A(t)$, $B(t)$ и $d(t)$ непрерывны на отрезке $[0,\ T]$.

С помощью замены переменных можно добиться исключения вектора $d(t)$ из системы. Однако, его наличие важно для будущих выводов. В частности, $d(t)$ будет использоваться при линеаризации нелинейной функции.

1.1.2 Терминальные условия

Целью управления является перевод объекта из заданного начального положения на некоторое линейное многообразие за заранее определенное время T:

$$x(0) = x_*, \quad Hx(T) = g^0, \tag{1.2}$$

где H является матрицей полного ранга и имеет размерность $m \times n$ $(m \leqslant n)$, g^0 — m-мерный вектор.

Важным частным случаем является вариант $m = n$. Тогда решается задача перевода объекта в заданное финальное положение $x(T) = H^{-1}g^0$. Другой частный случай: $m = 0$. При таком условии траектория управляемого объекта имеет свободный правый конец.

1.1.3 Ограничения на управления

На компоненты вектора управлений наложены прямые ограничения:

$$l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1,r}. \tag{1.3}$$

Следовательно, в базовой постановке задачи управления выбираются из r-мерного прямоугольника.

Будем выбирать управления $u_i(t)$ в классе кусочно-постоянных функций с заданным периодом дискретизации $h = T/N$:

$$u_i(t) = \begin{cases} u_{ik}, \quad t \in [t_{k-1},\ t_k), \\ k = \overline{1,N}, \end{cases} \qquad i = \overline{1,r}. \tag{1.4}$$

где $t_k = kh$.

1.1.4 Целевая функция

В данной работе рассматриваются три целевые функции: две линейные и одна квадратичная.

Финальное положение системы

$$J_1 = c^T x(T) \longrightarrow \min. \tag{1.5}$$

Первая целевая функция предполагает минимизацию линейной комбинации фазовых переменных в конечный момент времени. Таким образом, это условие на финальное положение системы. Здесь c — некоторый n-мерный вектор. Задача (1.1)–(1.5) является задачей Майера [13]. Отметим, что рассматривать такую целевую функцию имеет смысл только при незафиксированном правом конце, то есть когда $m < n$, и соответственно матрица H в условии (1.2) является прямоугольной.

Простейшим примером такого функционала может служить критерий $x_j(T) \to \max$, требующий максимизации финального положения одной из компонент фазового вектора.

Расход ресурса

$$J_2 = \int_0^T \|u(t)\|_1 \, dt = \int_0^T \sum_{i=1}^{r} |u(t)| \, dt \longrightarrow \min. \tag{1.6}$$

Вторая целевая функция является интегралом от нормы вектора управлений. В данном случае в качестве нормы будем рассматривать «манхэттенское расстояние», поскольку оно является наиболее удобным для сведения к линейным оптимизационным задачам.

Квадратичный функционал

$$J_3 = \int_0^T u(t)^T Q u(t) \, dt \longrightarrow \min. \tag{1.7}$$

Третья целевая функция представляет собой интеграл от квадратичной формы от вектора управлений. Здесь Q — симметричная матрица размерности $r \times r$, ее элементы будем обозначать символом $q_{i_1 i_2}$. Будем также предполагать, что Q неотрицательно определенная матрица. Задача (1.1)–(1.4), (1.7) является задачей Лагранжа [13].

Примером использования такой целевой функции является минимизация интеграла от квадрата нормы управления: $\int_0^T \|u(t)\|_2^2 \, dt \to \min$. Физическим смыслом такого критерия является минимизация использования управляющих воздействий (минимизация расхода энергии) подобно второй целевой функции, но в другом нормированном пространстве.

1.2 Обзор литературы

Одним из основателей математической теории управления принято считать Рудольфа Калмана, внесшего существенный вклад в развитие этой области знаний [14].

Классическим методом решения задачи оптимального управления является принцип максимума Понтрягина [15]. Он был сформулирован в 1958 году коллективом ученых под руководством Л. С. Понтрягина в виде необходимых условий оптимального решения. Метод является эффективным для множества частных случаев, но не является универсальным ввиду необходимости решения нелинейных алгебраических уравнений.

Альтернативой принципу максимума является метод динамического программирования. Он основан на принципе оптимальности Беллмана [16], который гарантирует выполнение достаточных условий оптимальности.

Существенный вклад в развитие теории устойчивости движения, теории автоматического управления и теории оптимальных процессов внес В. И. Зубов [17]. Им был предложен метод последовательных приближений для нахождения оптимальных программных движений и для решения задачи синтеза оптимальных управлений [18]. Отдельные результаты были получены для задачи с линейной системой и квадратичным функционалом, оптимальное стабилизирующее управление находится как решение матричного уравнения Риккати методом последовательных приближений.

Также задача построения стабилизирующего управления для линейно-квадратичной задачи освещена и во многих зарубежных источниках [19; 20]. Среди современных работ можно выделить подход с применением матричных неравенств для решения выпуклой линейно-квадратичной задачи [21] и алгоритм построения кусочно-линейного управления для невыпуклой задачи [22].

Отдельным направлением в исследованиях задачи оптимального управления можно выделить абстрактную теорию оптимального управления, основоположником которой является В. А. Якубович [23; 24]. Ему удалось сформулировать условия оптимальности для различных классов задач.

Актуальным вектором развития приближенных методов для задачи оптимального управления является метод «предиктор-корректор» [25], суть которого состоит в построении управления на некоторый горизонт времени вперед. В 1960–1970-ых годах был разработан алгоритм для линейных систем [26], заключающийся в сведении к задаче линейного программирования. Современные работы посвящены нелинейным задачам [27; 28] и индустриальным приложениям [29; 30].

Базовой основой для данного исследования послужили наработки коллектива ученых под руководством Р. Ф. Габасова. Ими был предложен двухэтапный алгоритм нахождения оптимального управления для линейных систем: сведение задачи управления к интервальной задаче линейного программирования (ИЗЛП) [31], с последующим решением этой задачи специально разработанным адаптивным методом [32]. Коллективом был представлен метод решения нелинейной задачи оптимального управления, о нем будет рассказано в параграфе 4.2.

Метод Габасова для линейного случая был апробирован автором этой работы для задачи управления вращательным движением вала электродвигателя [1], управления движением четырехколесной тележки [2], распределения инвестиций в отрасли многопродуктовой экономики [3]. Для линеаризованной модели квадрокоптера было построено оптимальное управление в статье [4].

1.3 Сведение задачи оптимального управления к интервальной задаче линейного программирования

1.3.1 Сведение терминальных условий

Ограничение на левый конец траектории будет выполняться автоматически, поскольку является начальным условием задачи Коши. Для того, чтобы выразить конечное состояние объекта как функцию от управлений, выпишем формулу Коши решения системы (1.1) с начальным условием $x(0) = x_*$ в точке T:

$$x(T) = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)\left(B(t)u(t) + d(t)\right)dt.$$

Здесь и далее $Y(t)$ — матрицант однородной системы дифференциальных уравнений с матрицей $A(t)$, нормированный в точке 0:

$$\dot{Y}(t) = A(t)Y(t), \quad Y(0) = E.$$

Заменим управление на его представление в виде кусочно-постоянной функции (1.4):

$$x(T) = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)d(t)dt + \sum_{i=1}^{r}\sum_{k=1}^{N}\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)u_{ik}dt.$$

Можем вынести за интеграл значения управления u_{ik} на временных сегментах, поскольку эти значения не зависят от времени:

$$x(T) = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)d(t)dt + \sum_{i=1}^{r}\sum_{k=1}^{N}\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dtu_{ik}.$$

В пункте 3.1.3 будет рассмотрен вопрос вычисления интегралов вида

$$\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt,$$

а пока же заметим, что этот интеграл не содержит искомых переменных, следовательно, его значение является постоянной величиной, которую обозначим за $b^{i,k}$:

$$b^{i,k} = \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i dt, \quad i = \overline{1,r}, \quad k = \overline{1,N}.$$

Оставшиеся слагаемые в правой части формулы Коши также не зависят от u_{ik}. Обозначим их сумму символом d^0:

$$d^0 = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)d(t)dt,$$

а вопрос вычисления d^0 осветим позднее.

Таким образом, можем выписать представление конечного состояния объекта $x(T)$ как линейную функцию от переменных u_{ik}:

$$x(T) = d^0 + \sum_{i=1}^{r}\sum_{k=1}^{N} b^{i,k}u_{ik}. \tag{1.8}$$

Тогда граничное условие (1.2) предстанет в виде системы линейных ограничений:

$$\sum_{i=1}^{r}\sum_{k=1}^{N} Hb^{i,k}u_{ik} = g^0 - Hd^0.$$

После введения новых обозначений получим:

$$\sum_{i=1}^{r}\sum_{k=1}^{N} h_l^{i,k}u_{ik} = g_l, \quad l = \overline{1,m}. \tag{1.9}$$

Здесь $h^{i,k} = Hb^{i,k}$, а $g = g^0 - Hd^0$. В итоге терминальные условия (1.2) перешли в m линейных равенств с rN переменными (1.9).

1.3.2 Сведение прямых ограничений на управления

Для данного класса управлений сведение является тривиальным. Прямые ограничения (1.3) очевидным образом перейдут в двусторонние

ограничения на значения управлений на временных сегментах:

$$l_{*i} \leqslant u_{ik} \leqslant l_i^*, \quad i = \overline{1,r}, \quad k = \overline{1,N}. \tag{1.10}$$

1.3.3 Сведение целевых функций

Для сведения **первой целевой функции** (1.5) воспользуемся найденным представлением конечного состояния объекта (1.8). Тогда можем записать критерий (1.5) как линейную функцию от управлений:

$$J_1 = c^T d^0 + \sum_{i=1}^{r} \sum_{k=1}^{N} c^T b^{i,k} u_{ik} \longrightarrow \min\,.$$

Поскольку первое слагаемое является константой, можем удалить его из целевой функции, при этом имея в виду, что от этого изменится значение функционала:

$$J_1 = \sum_{i=1}^{r} \sum_{k=1}^{N} c_{ik} u_{ik} \longrightarrow \min, \tag{1.11}$$

где $c_{ik} = c^T b^{i,k}$. Таким образом, (1.11) представляет собой линейный функционал относительно переменных u_{ik}.

В случае **второй целевой функции** (1.6) для преобразования суммы модулей в линейную функцию введем в модель два типа дополнительных переменных:

$$\begin{aligned} &u_{ik} = u_{ik}^+ - u_{ik}^-, \\ &u_{ik}^+ \geqslant 0, u_{ik}^- \geqslant 0, \end{aligned} \quad k = \overline{1,N}, \quad i = \overline{1,r}. \tag{1.12}$$

Тогда функционал (1.6) перейдет в сумму новых переменных:

$$J_2 = h \sum_{i=1}^{r} \sum_{k=1}^{N} \left(u_{ik}^+ + u_{ik}^-\right) \longrightarrow \min\,. \tag{1.13}$$

Исходя из вида целевой функции (1.13), существует следующая взаимосвязь между переменными:

$$\begin{cases} \text{если } u_{ik} \geqslant 0, \ \text{ то } u_{ik} = u_{ik}^{+}, \\ \text{если } u_{ik} \leqslant 0, \ \text{ то } u_{ik} = -u_{ik}^{-}. \end{cases}$$

Поэтому можно сказать, что u_{ik}^{+} обозначает положительную часть числа u_{ik}, а u_{ik}^{-} — отрицательную.

Далее перейдем к **третьему критерию**. Заменим в (1.7) управление на его представление в виде кусочно-постоянной функции (1.4):

$$J_2 = \int_0^T u(t)^T Q u(t) dt = \sum_{k=1}^{N} \int_{t_{k-1}}^{t_k} \left(\sum_{i_1=1}^{r} \sum_{i_2=1}^{r} u_{i_1 k} q_{i_1 i_2} u_{i_2 k} \right) dt.$$

Тогда, учитывая, что $t_k - t_{k-1} = h$, выпишем итоговый вид выражения:

$$J_2 = h \sum_{i_1=1}^{r} \sum_{i_2=1}^{r} q_{i_1 i_2} \sum_{k=1}^{N} u_{i_1 k} u_{i_2 k} \longrightarrow \min. \tag{1.14}$$

Получили функционал (1.14) в виде квадратичной формы относительно переменных u_{ik}. В пункте 3.1.2 покажем, что это неотрицательно определенная квадратичная форма.

1.3.4 Результаты сведения

В итоге линейная задача оптимального управления (1.1)–(1.5) была сведена к задаче (1.9)–(1.11):

$$\sum_{i=1}^{r} \sum_{k=1}^{N} c_{ik} u_{ik} \longrightarrow \min,$$

$$\sum_{i=1}^{r} \sum_{k=1}^{N} h_l^{i,k} u_{ik} = g_l, \quad l = \overline{1,m},$$

$$l_{*i} \leqslant u_{ik} \leqslant l_i^*, \quad i = \overline{1,r}, \quad k = \overline{1,N},$$

обладающей следующими характеристиками:

– rN переменных,

– m линейных равенств,

– $2rN$ линейных неравенств,

– линейная целевая функция.

Поскольку для всех переменных заданы двусторонние ограничения, такую задачу можно классифицировать как интервальную задачу линейного программирования, которая, кроме классических методов, также может быть эффективна решена и с использованием адаптивного метода [32].

При смене функционала на интеграл от суммы моделей компонентов управления, задача (1.1)–(1.4), (1.6) сводится к (1.9), (1.10), (1.12), (1.13):

$$h\sum_{i=1}^{r}\sum_{k=1}^{N}\left(u_{ik}^{+}+u_{ik}^{-}\right)\longrightarrow\min,$$

$$\sum_{i=1}^{r}\sum_{k=1}^{N}h_{l}^{i,k}u_{ik}=g_{l},\quad l=\overline{1,m},$$

$$l_{*i}\leqslant u_{ik}\leqslant l_{i}^{*},\quad i=\overline{1,r},\quad k=\overline{1,N},$$

$$u_{ik}=u_{ik}^{+}-u_{ik}^{-},\quad i=\overline{1,r},\quad k=\overline{1,N},$$

$$u_{ik}^{+}\geqslant 0,u_{ik}^{-}\geqslant 0,\quad i=\overline{1,r},\quad k=\overline{1,N}.$$

Это задача линейного программирования со следующими размерностями:

– $3rN$ переменных,

– $rN+m$ линейных равенств,

– $4rN$ линейных неравенств,

– линейная целевая функция.

Наконец, при использовании функционала Лагранжа задача управления (1.1)–(1.4), (1.7) перейдет в задачу квадратичного программирования (1.9), (1.10), (1.14):

$$h\sum_{i_1=1}^{r}\sum_{i_2=1}^{r}q_{i_1i_2}\sum_{k=1}^{N}u_{i_1k}u_{i_2k}\longrightarrow\min,$$
$$\sum_{i=1}^{r}\sum_{k=1}^{N}h_l^{i,k}u_{ik}=g_l,\quad l=\overline{1,m},$$
$$l_{*i}\leqslant u_{ik}\leqslant l_i^*,\quad i=\overline{1,r},\quad k=\overline{1,N}.$$

Характеристики этой задачи выписаны ниже:

– rN переменных,

– m линейных равенств,

– $2rN$ линейных неравенств,

– выпуклая квадратичная целевая функция.

1.4 Решение задачи линейного и квадратичного программирования

При решении разнообразных задач оптимального управления возникает необходимость поиска оптимальных значений для различных задач математического программирования. В рамках данной диссертационной работы мы столкнемся с несколькими классами оптимизационных задач:

1. Линейное программирование (Linear Programming, LP),
2. Квадратичное программирование (Quadratic Programming, QP),
3. Программирование с квадратичными ограничениями (Quadratically Constrained Programming, QCP),
4. Частично целочисленное линейное программирование (Mixed Integer Linear Programming, MILP),
5. Частично целочисленное квадратичное программирование (Mixed Integer Quadratic Programming, MIQP),

6. Частично целочисленное программирование с квадратичными ограничениями (Mixed Integer Quadratically Constrained Programming, MIQCP).

Несмотря на тот факт, что одни задачи являются частным случаем других (см. рис. 1.1), эффективнее оказывается рассматривать методы решения для каждого конкретного класса. В данном параграфе опишем методы решения каждого из представленных классов и приведем примеры специализированных программ, которыми можно воспользоваться для поиска оптимальных решений этих задач.

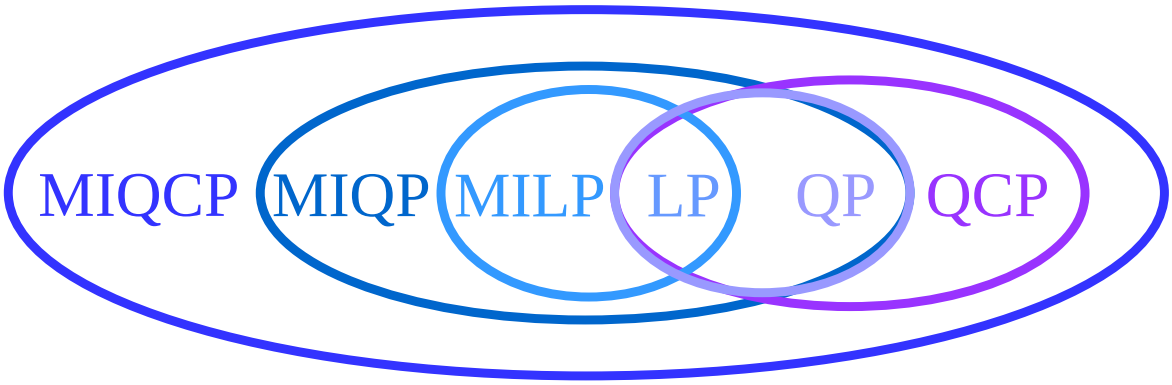

Рисунок 1.1 — Соотношение типов оптимизационных задач

1.4.1 Линейное программирование

Задача линейного программирования является классической оптимизационной задачей: ставится цель найти экстремум многомерной линейной функции при линейных ограничениях на переменные. С геометрической точки зрения, линейные ограничения порождают выпуклый многогранник в многомерном пространстве, а целевая функция определяет направляющий вектор. Оптимальным решением является некоторая вершина

многогранника, а в отдельных случаях оптимальными решениями могут быть все точки принадлежащие определенному ребру или грани.

Основоположником линейного программирования как отдельного класса оптимизационных задач принято считать Л. В. Канторовича, также предложившего метод решения поставленной задачи [33].

Сиплекс-метод. В 1947 году, работая над задачей об улучшении процесса планирования для ВВС, Дж. Б. Данциг сформулировал симплекс-метод [34] — эффективный метод поиска оптимального решения, считающийся одним из самых популярных подходов до сих пор.

Метод внутренней точки. Алгоритм был разработан 60-ых — 70-ых годах [35; 36] для задач выпуклой оптимизации с ограничениями. В 1984 году специально для задачи линейного программирования был адаптирован метод внутренней точки (Алгоритм Кармаркара [37]), решающий задачу за полиномиальное время.

Адаптивный метод. В начале 70-х годов Р. Ф. Габасовым совместно с Ф. М. Кирилловой был предложен и обоснован новый подход к решению задач линейного программирования. Это положило начало крупному направлению, известному как конструктивные методы оптимизации [38]. Результаты этого направления широко применялись в 70–80-х годах при решении прикладных задач.

Одним из преимуществ адаптивного метода можно выделить то, что его применение не ведёт к увеличению размерности задачи, что неизбежно при сведении исходной задачи к канонической форме задачи линейного программирования, пригодной для решения симплекс-методом [32].

Метод Эллипсоидов. Среди прочих методов дополнительно можно выделить метод эллипсоидов [39], ставший первым полиномиальным методом решения задачи линейного программирования, но на практике не оказавшийся достаточно эффективным.

1.4.2 Квадратичное программирование

В литературе принято отдельно рассматривать два класса квадратичных задач:

1. Квадратичное программирование — задача оптимизации с линейными ограничениями и квадратичной целевой функцией,
2. Программирование с квадратичными ограничениями — задача оптимизации с линейными и квадратичными ограничениями и линейной или квадратичной целевой функцией.

Важной характеристикой квадратичных задач является условие выпуклости ограничений и целевой функции. В выпуклом случае множество допустимых решений содержит единственный оптимум, являющийся глобальным. Сложность решения задачи существенно зависит от выполнения этого условия. В данной работе будем рассматривать оптимизационные задачи только с выпуклыми ограничениями и целевыми функциями.

Для решения задачи квадратичного программирования применяют методы активного набора [40] и методы внутренней точки [41]. Задача с квадратичными ограничениями является более сложной. Обычно ее рассматривают как частный случай более общей задачи конической оптимизации (Second-Order Cone Programming, SOCP) [42]. С помощью метода внутренней точки оптимальное решение этой задачи может быть найдено за полиномиальное время [43].

1.4.3 Частично целочисленное программирование

Более общим случаем линейных и квадратичных задач являются частично целочисленные задачи — когда некоторые переменные являются

бинарными или целочисленными. Опишем способы решения задачи MILP, подразумевая, что для частично целочисленных квадратичных задач справедливы аналогичные подходы.

Перебор значений. Простейшим подходом к решению поставленной задачи является полный перебор значений бинарных и целочисленных переменных. Значения дискретных переменных фиксируются, таким образом, задача сводится к задаче линейного программирования, которая решается стандартными методами. Среди всех наборов значений, для которых существует допустимое решение, выбирается тот, на котором достигается оптимум целевой функции.

Метод ветвей и границ. Алгоритм был предложен 1960-ых годах [44] и в каком-то смысле является более эффективным перебором значений дискретных переменных. На первом этапе решается релаксированная задача — все бинарные и целочисленные переменные рассматриваются как вещественные. После решения, в случае существования оптимального решения, возможны две ситуации:

1. Значения всех дискретных переменных являются целыми числами. Тогда оптимальное решение релаксированной задачи является оптимальным решением исходной задачи.
2. Если существует дискретная переменная, принявшая не целое значение $x_i = x_i^*$, то необходимо исходную задачу разбить на две. В первой подзадаче будет наложено дополнительное ограничение: $x_i \leqslant \lfloor x_i^* \rfloor$, во второй — дополнительное ограничение $x_i \geqslant \lceil x_i^* \rceil$. Обе подзадачи решается тем же алгоритмом, так образом, алгоритм является рекурсивным. В случае существования решений для обеих подзадач, решением исходной задачи будет решение подзадачи с лучшим значением целевой функции. Несмотря на теоретические преимущества метода — гарантированное схождение к оптимальному решению, на практике он зачастую бывает слишком медлительным.

Метод ветвей и отсечений. Одним из улучшений метода ветвей и границ является метод ветвей и отсечений, предложенный в середине 90-ых [45; 46]. Помимо ветвления, алгоритм предполагает для каждой подзадачи отсекать заведомо нецелочисленные решения путем введения дополнительных ограничений. Существует множество алгоритмов отсечений, одним из самых популярных является алгоритм Гомори [47; 48].

Метод ветвей и цен. Модификация метода ветвей и границ, основанная на методе генерации столбцов [49]. На первом шаге формируется неполная задача, содержащая не все переменные и ограничения исходной задачи. Это делается из предположения, что часть переменных примет нулевое значение, следовательно, их можно исключить из задачи. Обычно для переформулирования задачи используется декомпозиция Данцига—Вулфа [50]. Далее, в течение алгоритма ветвления, определенные переменные могут возвращаться в модель. Такой подход позволяет сократить время расчета и использование оперативной памяти. Существует также метод ветвей, цен и отсечений [51] — дополнительно закладывается этап отсечения нецелочисленных решений.

Эвристики. Для ускорения оптимизации может быть оправданно параллельно с методом ветвей и отсечений также использовать и эвристические алгоритмы. Основным преимуществом таких алгоритмов является способность быстро находить качественные решения, однако, недостаток состоит в отсутствии гарантии нахождения решения, и невозможность оценить их оптимальность. Примеры эвристических алгоритмов:

1. Генетический алгоритм [52],
2. Local Branching [53],
3. RINS (Relaxation Induced Neighborhood Search) [54].

1.4.4 Использование специализированного программного обеспечения

Существуют разнообразные программные комплексы для решения задач математического программирования, в которых реализованы перечисленные выше алгоритмы решения. Такие комплексы обычно представлены в виде программных библиотек, их принято называть решателями или солверами.

Для выбора конкретного солвера, необходимо учитывать специфику решаемой задачи и общую архитектуру проекта. Как правило, можно руководствоваться следующими критериями:

1. Типы решаемых задач,
2. Производитель, тип лицензии, стоимость лицензии,
3. Производительность на бенчмарках — наборе стандартных задач [55],
4. Наличие удобного руководства, поддержка производителей и сообщества пользователей,
5. Использование с помощью языков программирования и программных пакетов (API),
6. Возможность задавать стартовое решение — использование начального решения позволяет ускорить поиск оптимума,
7. Возможность управления настройками солвера,
8. Функционал анализа несовместных моделей — в случае отсутствия допустимых решений, алгоритм находит множество противоречащих ограничений.
9. Возможность применения пользовательских методов — поиск решения с помощью алгоритмов, учитывающих специфику задачи параллельно с основными методами солвера.

В таб. 1.1 приведено небольшое сравнение популярных солверов. Рассмотрены самые известные коммерческие решатели Gurobi [56] и CPLEX [57], выпущенный в 2019 году COPT [58], и их общедоступные аналоги SCIP [59] и CBC [60]. Также со списком солверов с классификацией по типам решаемых задач можно ознакомиться на сайте системы для математического моделирования GAMS [61].

Таблица 1.1 — Сравнение солверов

Метрики	**Gurobi**	**CPLEX**	**COPT**	**SCIP**	**CBC**
Производитель	Gurobi Optimization, США	IBM, США	Cardinal Operation, Китай	Zuse Institute Berlin, Германия	COIN-OR Foundation
Тип лицензии	Коммерческая и бесплатная академическая лицензия	Коммерческая и бесплатная академическая лицензия	Коммерческая и бесплатная академическая лицензия	Свободное использование для исследовательских целей членам некоммерческих организаций	Свободное распространение
Классы задач	(MI)LP, (MI)QP, (MI)QCP	(MI)LP, (MI)QP, (MI)QCP	(MI)LP, (MI)QP, (MI)QCP	(MI)LP, (MI)QP, (MI)QCP	(MI)LP
API	C/C++, Java, .NET, Python, MATLAB, R	C/C++, Java, .NET, Python, MATLAB	C/C++/C#, Java, Python	C/C++, Python, Julia, MATLAB, Java	C++, Mathematica, MATLAB, R, Python, Julia, Rust
Бенчмарки LP (Симплекс-метод)	1,43	—	1	44,9	9,52
Бенчмарки LP (Метод внутренней точки)	1,33	—	1	—	77,8
Бенчмарки MILP	1	—	2,34	8,39	10,2
Бенчмарки QCP	2,03	—	1	—	—
Бенчмарки MIQCP	1	—	—	13,1	—

Для бенчмарков указано среднее относительное время решения задач, нормированное по результату лидера. Пропуски в данных означают отсутствие информации по производительности на указанных бенчмарках.

В текущей работе для решения оптимизационных задач был выбран солвер COPT. Обращение к солверу производилось на языке программирования Python с помощью библиотеки *coptpy*, предоставляемой производителями солвера.

1.5 Пример. Демпфирования колебаний «спящего волчка» Лагранжа

В качестве иллюстрации работы алгоритма рассмотрим задачу демпфирования колебаний «спящего волчка» Лагранжа [62].

Движение волчка описывается системой обыкновенных дифференциальных уравнений с управлением:

$$\ddot{\xi} + B\dot{\eta} - A\xi = u_1,$$
$$\ddot{\eta} - B\dot{\xi} - A\eta = u_2,$$

где $\xi(t)$ и $\eta(t)$ обозначают координаты вершины волчка; $u_1(t)$, $u_2(t)$ — обобщенные силы; A и B — параметры модели, характеризующие физические свойства объекта. В данном примере положим $A = 1$, $B = 3$.

Терминальной задачей является перевод объекта из положения

$$(\xi = -10,\ \dot{\xi} = 5,\ \eta = 6,\ \dot{\eta} = 8)$$

в нулевое положение. Таким образом, в начальный момент времени объект находится в точке $(-10,\ 6)$ и имеет ненулевую скорость. Необходимо поместить объект в состояние покоя.

Значение параметра T установим равным 20, то есть будем рассматривать процесс на отрезке $[0,\ 20]$. Разобьем этот временной отрезок на 50 равных участков. Нижние границы установим равными -2, а верхние границы положим равными 2. Управление будем искать в классе кусочно-постоянных функций.

В качестве целевой функции будем рассматривать интеграл от квадрата нормы вектора управлений.

Тогда после замены переменных

$$x_1 = \xi, \quad x_2 = \dot{\xi}, \quad x_3 = \eta, \quad x_4 = \dot{\eta},$$

получим задачу оптимального управления с линейной системой и квадратичным функционалом:

$$\int_0^{20} \left(u_1^2(t) + u_2^2(t)\right) dt \longrightarrow \min,$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -3 \\ 0 & 0 & 0 & 1 \\ 0 & 3 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix},$$

$$\begin{pmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \\ x_4(0) \end{pmatrix} = \begin{pmatrix} -10 \\ 5 \\ 6 \\ 8 \end{pmatrix}, \quad \begin{pmatrix} x_1(20) \\ x_2(20) \\ x_3(20) \\ x_4(20) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$\begin{pmatrix} -2 \\ -2 \end{pmatrix} \leqslant \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} \leqslant \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \quad t \in [0,\, 20].$$

Процедура сведения была реализована на языке Python 3, а задача квадратичного программирования была решена с помощью решателя COPT (см. приложение А).

Общее время работы программы составило 0,32 секунд. Значение целевой функции равняется 92,80. Результаты приведены на рис. 1.2.

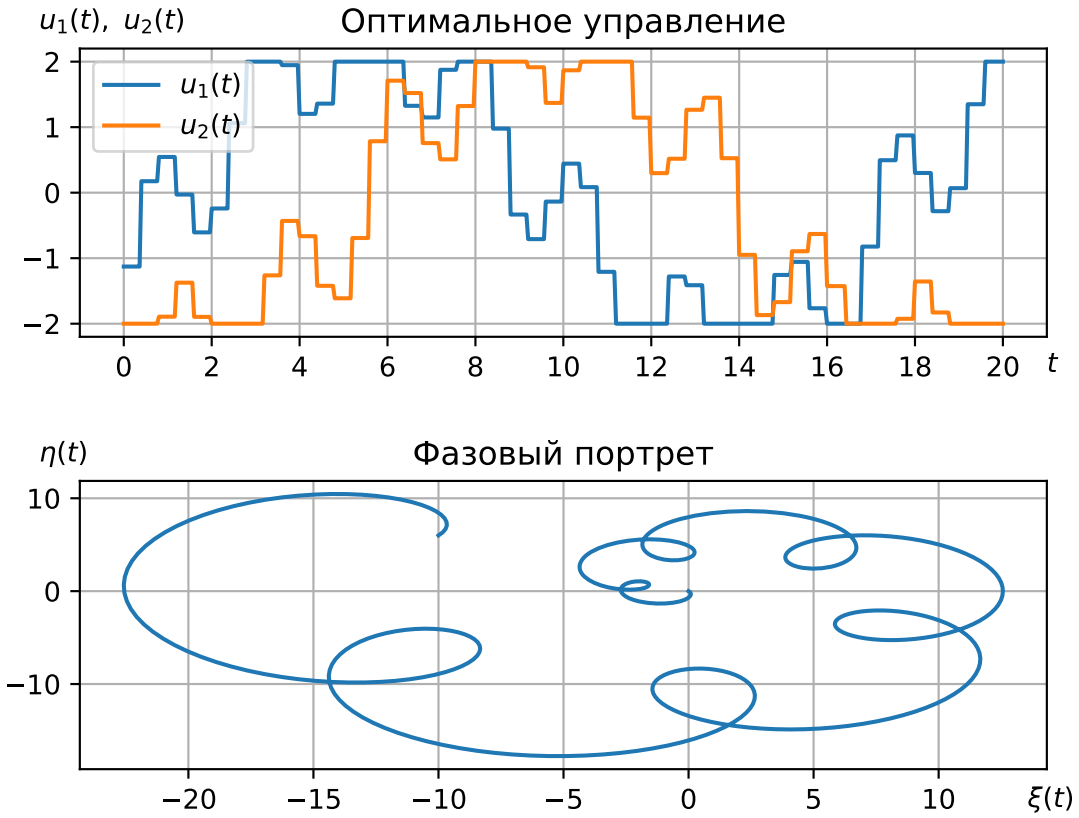

Рисунок 1.2 — Демпфирование колебаний «спящего волчка» Лагранжа при выборе управления в классе кусочно-постоянных функций

1.6 Выводы по главе 1

В главе 1 был описан алгоритм решения линейной задачи оптимального управления, состоящий из двух основных этапов:

1. **Сведение к задаче математического программирования.** Такой переход осуществляется с использованием формулы Коши и возможен благодаря выбору управлений из класса кусочно-постоянных функций. Этап сведения подробно описан в параграфе 1.3. В зависимости от вида целевой функции формируется задача линейного или выпуклого квадратичного программирования.
2. **Решение задачи математического программирования.** Поскольку задача линейного программирования и выпуклая задача

квадратичного программирования хорошо изучены, для решения можно воспользоваться одним из специализированных программных продуктов (солвером). В параграфе 1.4 описаны идеи основных алгоритмов решения таких задач, приведен обзор существующих солверов, осуществлено сравнение скорости их работы для различных классов задач.

Изложенные в главе 1 результаты опираются на теоретическую базу, сформированную научной школой Р. Ф. Габасова [31; 32; 38]. С точки зрения общей структуры диссертационной работы, текущая глава является основой, на которую мы будем опираться в последующих разделах. Было выбрано несколько направления развития, которые являются актуальными, но содержат неразрешенные вопросы, в следствие чего представляют интерес для исследований:

1. Построение множеств достижимости и управляемости для базовой задачи (1.1)–(1.4).
2. Выбор управления из альтернативных классов функций: кусочно-линейного и кусочно-квадратичного.
3. Добавление в базовую модель дополнительных выпуклых и невыпуклых ограничений на управление и фазовые переменные.
4. Замена линейной системы обыкновенных дифференциальных уравнений на нелинейную.
5. Перестроение управления в режиме реального времени при отклонении объекта от траектории программного режима.

ГЛАВА 2. ПОСТРОЕНИЕ МНОЖЕСТВ ДОСТИЖИМОСТИ И УПРАВЛЯЕМОСТИ

Практическая реализация решения задач оптимального управления с переходом к линейному программированию показала теоретическую важность обоснования целесообразности решения задачи, то есть существования оптимального решения. Это привело к задаче анализа множеств достижимости и управляемости исходной задачи. В главе 2 рассматривается вопрос о построении этих множеств. В параграфе 2.1 приведены определения множеств достижимости и управляемости. В 2.3 показано, что задачи построения этих множеств эквиваленты и могут быть сведены к задаче линейного отображения многомерного куба. Далее в параграфе 2.5 анализируются существующие подходы к решению поставленной задачи. Поскольку они чрезмерно трудоемки, возникает вопрос о создании более эффективного алгоритма. В 2.6 предложен алгоритм построения искомых множеств как системы линейных неравенств. В виде теоремы доказана корректность алгоритма. Далее оценена сложность представленного подхода.

2.1 Постановка задачи

Целью диссертационной работы является решение задачи оптимального управления для различных классов систем. Однако заранее может быть не известно, существует ли решение поставленной задачи. Для ответа на этот вопрос будет полезно построить два множества: **множество достижимости** — все возможные состояния объекта, в которые он может попасть за определенное время под воздействием допустимого управления, и **множество управляемости** — все начальные положения системы, из

которых можно попасть за указанное время в заданное положение с помощью допустимого управления.

Рассмотрим задачу управления (1.1)–(1.4). Ее можно дополнить любой из целевых функций (1.5)–(1.7), однако, они не будут оказывать влияние на существование допустимых управлений. Поэтому в текущей главе будет идти речь о задаче терминального управления без какого-либо критерия.

Теперь формализуем понятия множества достижимости и множества управляемости.

Определение 1. Пусть в момент $\hat{t}$ $(0 \leqslant \hat{t} < T)$ управляемый объект находится в состоянии $\hat{x}$. Будем называть *множеством достижимости* $X_1(t, \hat{t}, \hat{x})$ задачи (1.1)–(1.4) множество всевозможных состояний $x^{(1)}(t)$, где $t \in (\hat{t},\ T]$, в которые система (1.1) может перейти за время $t - \hat{t}$ при начальном условии $x(\hat{t}) = \hat{x}$ и выборе управления вида (1.3), удовлетворяющего ограничению (1.4):

$$X_1(t, \hat{t}, \hat{x}) = \{x^{(1)}(t) = x(t) \mid \dot{x}(\tau) \equiv A(\tau)x(\tau) + B(\tau)u(\tau) + d(\tau),$$
$$x(\hat{t}) = \hat{x},\ \tau \in [\hat{t},\ t],\ u(\tau) \text{ удовлетворяет (1.3), (1.4)}\}.$$

Определение 2. Пусть в момент $\hat{t}$ $(0 < \hat{t} \leqslant T)$ управляемый объект должен попасть в состояние $\hat{x}$. Будем называть *множеством управляемости* $X_2(t, \hat{t}, \hat{x})$ задачи (1.1)–(1.4) множество всевозможных состояний $x^{(2)}(t)$, где $t \in [0,\ \hat{t})$, из которых система (1.1) может перейти за время $\hat{t} - t$ в состояние $x(\hat{t}) = \hat{x}$ и выборе управления вида (1.3), удовлетворяющего ограничению (1.4):

$$X_2(t, \hat{t}, \hat{x}) = \{x^{(2)}(t) = x(t) \mid \dot{x}(\tau) \equiv A(\tau)x(\tau) + B(\tau)u(\tau) + d(\tau),$$
$$x(\hat{t}) = \hat{x},\ \tau \in [t,\ \hat{t}],\ u(\tau) \text{ удовлетворяет (1.3), (1.4)}\}.$$

Задача текущей главы заключается в описании этих множеств в удобном для анализа виде.

Несмотря на то, что рассматривается линейная управляемая система, методы решения нелинейных задач зачастую состоят в использовании

линеаризованных систем. Поэтому предполагается, что полученные здесь результаты будут применены в алгоритмах поиска оптимального управления в нелинейном системах в главе 4.

2.2 Обзор литературы

Задача построения множеств достижимости и управляемости была неоднократно рассмотрена в классических источниках [15; 63; 64], в частности, в них представлены конструктивные методы построения множеств достижимости в виде эллипсоидов.

В современной литературе можно найти работы, посвященные построению множеств для линейной задачи [65; 66]. Также существуют подходы к построению множеств достижимости для нелинейных систем [67–69], однако, описываемые подходы представляют трудоемкие алгоритмы, состоящие, например, в решении задачи оптимального управления.

Отличие этой работы — акцент на специальном виде управления: оно выбирается из класса кусочно-постоянных функций, благодаря чему, как будет показано ниже, множества достижимости и управляемости являются выпуклыми многогранниками и описываются системами линейных неравенств. Целью является реализация конструктивного алгоритма построения этих множеств, с помощью которого можно оперативно проверить, принадлежит ли та или иная точка множеству достижимости или управляемости.

Результаты представленные в текущей главе были опубликованы в статье [11].

2.3 Переход к задаче линейного отображения

Покажем, что задачи построения множеств $X_1(t, \hat{t}, \hat{x})$ и $X_2(t, \hat{t}, \hat{x})$ являются эквивалентными и могут быть сведены к задаче линейного отображения выпуклого многогранника.

2.3.1 Сведение для множества достижимости

Выпишем для задачи (1.1) с начальным условием $x(\hat{t}) = \hat{x}$ формулу Коши, обозначив $\hat{Y}(t)$ матрицант системы, нормированный в точке $\hat{t}$:

$$x(t) = \hat{Y}(t) \left(\hat{x} + \int_{\hat{t}}^{t} \hat{Y}(t)\hat{Y}^{-1}(\tau) \left(Bu(\tau) + d(\tau)\right) d\tau \right) \tag{2.1}$$

и воспользуемся представлением управления в виде (1.3).

Среди чисел $\overline{1, N}$ найдем k_1 такое, что $(k_1-1)h \leqslant \hat{t} < k_1 h$. Аналогично получим число k_2 таким образом, что либо $t \in [0, T)$, тогда среди чисел $\overline{1, N}$ можем определить k_2, для которого $(k_2 - 1)h \leqslant t < k_2 h$, либо $t = T$, тогда положим $k_2 = N$.

Зададим последовательность

$$s_k = \begin{cases} \hat{t} & \text{при } k = k_1 - 1, \\ kh & \text{при } k = \overline{k_1, k_2 - 1}, \\ t & \text{при } k = k_2. \end{cases} \tag{2.2}$$

Теперь перепишем (2.1), раскрыв скобки и воспользовавшись представлением управления в виде (1.3) с учетом (2.2):

$$x(t) = \hat{Y}(t)\hat{x} + \sum_{i=1}^{r}\sum_{k=k_1}^{k_2}\left(\int_{s_{k-1}}^{s_k}\hat{Y}(t)\hat{Y}^{-1}(\tau)b_i(\tau)d\tau u_{i,k}\right) + \\ + \int_{\hat{t}}^{t}\hat{Y}(t)\hat{Y}^{-1}(\tau)d(\tau)d\tau. \tag{2.3}$$

Здесь b_i обозначает i-й столбец матрицы B.

Произведем замены:

$$U_1 = (u_{1,k_1}, u_{2,k_1}, \ldots, u_{r,k_1}, u_{1,k_1+1}, u_{2,k_1+1}, \ldots, u_{r,k_2})^T, \\ b_{i,k}^{(1)} = \int_{s_{k-1}}^{s_k}\hat{Y}(t)\hat{Y}^{-1}(\tau)b_i d\tau, \\ B_1 = \left(b_{1,k_1}^{(1)}, b_{2,k_1}^{(1)}, \ldots, b_{r,k_1}^{(1)}, b_{1,k_1+1}^{(1)}, b_{2,k_1+1}^{(1)}, \ldots, b_{r,k_2}^{(1)}\right), \\ q_1 = \hat{Y}(t)\hat{x} + \int_{\hat{t}}^{t}\hat{Y}(t)\hat{Y}^{-1}(\tau)d(\tau)d\tau. \tag{2.4}$$

Тогда равенство (2.3) с учетом (2.4) перепишем следующим образом:

$$x = B_1U_1 + q_1. \tag{2.5}$$

Замечание 2.1. Вообще говоря, матрица B_1 и вектор q_1, а следовательно, и вектор x зависят от параметров t, $\hat{t}$, $\hat{x}$. Более того, выбор t и $\hat{t}$ влияет на размер вектора U_1 и количество столбцов в B_1. Будем считать их зафиксированными, и для простоты опустим обозначения зависимостей.

Таким образом, получили систему линейных алгебраических уравнений, в которых матрица B_1 имеет размерность $n \times R$, где $R = r(k_2 - k_1 + 1)$. Конечное значение R определяется выбором точек t и $\hat{t}$, например, при $t = 0$ и $\hat{t} = T$ имеем $R = rN$.

Введем $L_1^{(1)}$ и $L_2^{(1)}$ как R-мерные векторы, состоящие из $k_2 - k_1 + 1$ повторений векторов $(l_{*1}, \ldots, l_{*r})^T$ и $(l_1^*, \ldots, l_r^*)^T$ соответственно.

Тогда прямые ограничения на компоненты вектора U_1 можно переписать в виде

$$L_1^{(1)} \leqslant U_1 \leqslant L_2^{(1)}. \tag{2.6}$$

Как видно, задача построения множества достижимости состоит в нахождении всех векторов $x(t)$ таких, что существует вектор U_1, компоненты которого удовлетворяют двусторонним ограничениям (2.6), и выполняется линейное равенство (2.5).

Немного упростим вид задачи (2.5), (2.6). Сделаем замену вектора управлений и вектора фазовых переменных:

$$\begin{gathered} v_i = \frac{2U_{1,j} - \left(L_{1,j}^{(1)} + L_{2,j}^{(1)}\right)}{L_{1,j}^{(2)} - L_{1,j}^{(1)}}, \quad j = \overline{1,R}, \\ y = x - \frac{1}{2} B_1 \left(L_1^{(1)} + L_2^{(1)}\right) - q_1. \end{gathered} \tag{2.7}$$

Также введем матрицу P размера $n \times R$, состоящую из столбцов

$$p_j = \frac{1}{2} B_{1,j} \left(L_{2,j}^{(1)} - L_{1,j}^{(1)}\right),$$

где $B_{1,j}$ — j-й столбец матрицы B_1.

В новых переменных (2.7) задача (2.5), (2.6) примет вид

$$\begin{gathered} y = Pv, \\ -1 \leqslant v_j \leqslant 1,\ j = \overline{1,R}. \end{gathered} \tag{2.8}$$

Таким образом, для построения множества достижимости необходимо произвести линейное отображение R-мерного куба в n-мерное пространство. В качестве матрицы линейного оператора выступает матрица P размерности $n \times R$.

2.3.2 Сведение для множества управляемости

Аналогичным образом сведем задачу нахождения множества управляемости. Для задачи (1.1) с конечным условием $x(\hat{t}) = \hat{x}$ рассмотрим формулу Коши, выразив $\hat{x}$:

$$\hat{x} = \hat{Y}^{-1}(t)x(t) + \int_t^{\hat{t}} \hat{Y}^{-1}(\tau) \left(B(\tau)u(\tau) + d(\tau)\right) d\tau. \tag{2.9}$$

Среди чисел $\overline{1, N}$ найдем k_1 и k_2 такие, что $(k_1 - 1)h \leqslant t < k_1 h$, и либо $\hat{t} \in [0,\ T)$, тогда существует k_2 такое, что $(k_2 - 1)h \leqslant \hat{t} < k_2 h$, либо $\hat{t} = T$, тогда $k_2 = N$.

Зададим последовательность

$$s_k = \begin{cases} t & \text{при } k = k_1 - 1, \\ kh & \text{при } k = \overline{k_1, k_2 - 1}, \\ \hat{t} & \text{при } k = k_2. \end{cases} \tag{2.10}$$

Перепишем (2.9), раскрыв скобки и воспользовавшись представлением управления в виде (1.3) с учетом (2.10):

$$\begin{aligned} x(t) = \hat{Y}(t)\hat{x} - \sum_{i=1}^{r} \sum_{k=k_1}^{k_2} \left(\int_{s_{k-1}}^{s_k} \hat{Y}(t)\hat{Y}^{-1}(\tau) b_i(\tau) d\tau u_{ik} \right) - \\ - \int_{t}^{\hat{t}} \hat{Y}(t)\hat{Y}^{-1}(\tau) d(\tau) d\tau. \end{aligned} \tag{2.11}$$

Введем обозначения:

$$\begin{gathered} U_2 = (u_{1k_1}, u_{2k_1}, \ldots, u_{rk_1}, u_{1k_1+1}, u_{2k_1+1}, \ldots, u_{rk_2})^T, \\ b_{ik}^{(2)} = - \int_{s_{k-1}}^{s_k} \hat{Y}(t)\hat{Y}^{-1}(\tau) b_i d\tau, \\ B_2 = \left(b_{1k_1}^{(2)}, b_{2k_1}^{(2)}, \ldots, b_{rk_1}^{(2)}, b_{1k_1+1}^{(2)}, b_{2k_1+1}^{(2)}, \ldots, b_{rk_2}^{(2)} \right), \\ q_2 = \hat{Y}(t)\hat{x} - \int_{t}^{\hat{t}} \hat{Y}(t)\hat{Y}^{-1}(\tau) d(\tau) d\tau. \end{gathered} \tag{2.12}$$

Тогда равенство (2.11) с учетом (2.12) перепишем как

$$x = B_2 U_2 + q_2. \tag{2.13}$$

Равенства (2.5), (2.13) имеют одинаковую форму. Тогда с учетом прямых ограничений задачи построения множества достижимости и множества управляемости могут быть сведены к задаче (2.8). В этом смысле они эквивалентны.

2.4 Свойства множеств

Здесь и далее будем рассматривать свойства множества достижимости, подразумевая, что они также справедливы и для множества управляемости.

С учетом результатов сведения в параграфе 2.3 введем два множества.

Определение 3. Будем называть *множеством допустимых управлений* множество $\mathbb{V} = [-1;\ 1]^R$, а *множеством допустимых состояний* — множество $\mathbb{Y} = \{y \in \mathbb{R}^n |\ \ \exists v \in \mathbb{V} : y = Pv\}$.

Свойство 1. $\mathbb{Y}$ — непустое, ограниченное, замкнутое, выпуклое множество.

Оно следует из того, что множество $\mathbb{V}$ является непустым, ограниченным, замкнутым и выпуклым, а линейное отображение сохраняет вышеперечисленные свойства.

Свойство 2. $\mathbb{Y}$ — выпуклая оболочка точек $\{p^1, \ldots, p^{2^R}\}$. Здесь $p^k = Pv^k$, а v^k — вершины $\mathbb{V}$.

Множество $\mathbb{V}$, очевидно, является выпуклой оболочкой своих вершин, т. е. $\forall v \in V\ \exists \lambda_k,\ k = \overline{1,2^R}$: $v = \sum\limits_{k=1}^{2^R} \lambda_k v^k$, где $\sum\limits_{k=1}^{2^R} \lambda_k = 1,\ \lambda_k \geqslant 0$.

Тогда $\forall y \in Y\ \exists \lambda_k$ такой, что

$$y = \sum_{k=1}^{2^R} \lambda_k P v^k = \sum_{k=1}^{2^R} \lambda_k p^k, \quad \sum_{k=1}^{2^R} \lambda_k = 1, \quad \lambda_k \geqslant 0, \quad k = \overline{1,2^R}.$$

Следовательно, множество $\mathbb{Y}$ — выпуклая оболочка векторов $p^k = Pv^k$.

Свойство 3. $\mathbb{Y}$ — выпуклый многогранник, вершинами которого является подмножество набора векторов $\{p^1, \ldots, p^{2^R}\}$, где $p^k = Pv^k$, а v^k — вершины $\mathbb{V}$.

Оно вытекает из свойства 2 и свойств выпуклых оболочек.

Свойство 4. Если $y \in \mathbb{Y}$, то и $-y \in \mathbb{Y}$.

Если $y \in \mathbb{Y}$, то $\exists v \in \mathbb{V}: \ y = Pv$. Но и $-v \in \mathbb{V}$, а следовательно, $P(-v) = -y \in \mathbb{Y}$.

Свойство 5. Множество $\mathbb{Y}$ описывается системой неравенств типа $-b \leqslant Ay \leqslant b$.

Это множество является выпуклым многогранником и может быть описано системой линейных неравенств. По свойству 4, если выполняется $a_i y \leqslant b_i$, то это неравенство будет справедливо и для $-y : a_i(-y) \leqslant b_i$. Тогда $-b_i \leqslant a_i y$.

2.5 Обзор алгоритмов

Существует несколько подходов к решению задачи (2.8). Коротко опишем их ниже.

1) Построение выпуклой оболочки. Как было показано в свойстве 3, многогранник $\mathbb{Y}$ есть выпуклая оболочка точек Pv^k, где v^k — вершины куба $\mathbb{V}$. Однако уже при достаточно небольшой размерности R такой подход выглядит нереализуемым, учитывая вычислительную сложность алгоритмов построения выпуклых оболочек. Так, при $R = 100$ будет получено $2^{100} \approx 10^{30}$ вершин куба.

2) Суммы Минковского. Отложим в пространстве $\mathbb{R}^n$ R отрезков: от точки $-p^1$ до точки p^1, от точки $-p^2$ до точки p^2 и т. д. Тогда найдем множество $\mathbb{Y}_1$ как геометрическое место точек, принадлежащих первому отрезку: $\mathbb{Y}_1 = \{y \in \mathbb{R}^n |\ y = \gamma p^1, -1 \leqslant \gamma \leqslant 1\}$. Множество $\mathbb{Y}_2$ определим как сумму Минковского [70] множества $\mathbb{Y}_1$ и второго отрезка:

$$\mathbb{Y}_2 = \{y_1 + y_2 \in \mathbb{R}^n |\ y_1 \in \mathbb{Y}_1,\ y_2 = \gamma p^2, -1 \leqslant \gamma \leqslant 1\}.$$

Продолжим строить суммы для всех отрезков, и в конце получим множество $\mathbb{Y}$:

$$\mathbb{Y} = \left\{ y_1 + y_2 \in \mathbb{R}^n | \; y_1 \in \mathbb{Y}_{R-1}, \; y_2 = \gamma p^R, -1 \leqslant \gamma \leqslant 1 \right\}$$

К сожалению, и такой подход также имеет экспоненциальную сложность. На первом шаге будет построен отрезок, соединяющий две точки, на втором — многоугольник с четырьмя вершинами, а на последнем — многогранник, который может содержать 2^R вершин.

3) Алгоритм Фурье — Моцкина. Более перспективными выглядят алгоритмы, работающие не с вершинами многогранника, а с его гранями. К ним относится метод элиминации Фурье — Моцкина [71]. Его суть заключается в исключении переменных из системы неравенств. Метод итеративно удаляет по одной переменной из неравенств. Тогда, если рассмотреть систему (2.8) как систему неравенств относительно переменных y и v, задача сведется к исключению R переменных v_i из этой системы.

Тем не менее и у такого подхода есть недостаток — при исключении переменных генерируются слишком много избыточных ограничений, и, если не отсеивать их на каждом шаге, в концу алгоритма будет получена система из $n \cdot 2^R$ неравенств.

2.6 Метод построения множеств. Теорема

Зададимся целью построения более эффективного алгоритма. На первом шаге необходимо удалить лишние (линейно зависимые) переменные состояния. Если ранг матрицы P меньше количества строк, выделим $n - \operatorname{rank} P$ компонент вектора y и выразим их через остальные. Таким образом, будет получено $n - \operatorname{rank} P$ ограничений типа равенств. В дальнейшем будем предполагать, что этот этап пройден, и в распоряжении имеется мат-

рица P полного ранга: $\operatorname{rank} P = n \leqslant R$. Кроме того, будем подразумевать, что $n \geqslant 2$ (при $n = 1$ алгоритм построения множества довольно очевиден).

Теперь будем перебирать столбцы матрицы P. Выберем $n-1$ линейно независимых столбцов и найдем фундаментальную систему решений относительно вектора-строки $\tilde{a}$ для уравнения

$$\tilde{a}\left(p^{j_1}, \ldots, p^{j_{n-1}}\right) = 0. \tag{2.14}$$

Это однопараметрическое семейство решений. Выберем из него какое-нибудь значение. Например, для единственности решения можно добавить ограничение $\|\tilde{a}\| = 1$. По сути вектор $\tilde{a}$ является нормальным вектором плоскости в пространстве $\mathbb{R}^n$, построенной по векторам $p^{j_1}, \ldots, p^{j_{n-1}}$.

Теперь умножим $\tilde{a}$ на матрицу P и найдем максимум произведения полученного выражения на вектор $v \in \mathbb{V}$:

$$\tilde{\beta} = \max_{v \in \mathbb{V}} \tilde{a} P v = \sum_{j=1}^{R} |\tilde{a} p^j|, \tag{2.15}$$

где $\tilde{a}P$ представляет собой R-мерную вектор-строку, содержащую значения проекций вектора $\tilde{a}$ на столбцы матрицы P. Причем по построению проекции на векторы $p^{j_1}, \ldots, p^{j_{n-1}}$ равны нулю. Поскольку компоненты вектора v изменяются от -1 до 1 и не зависят друг от друга, то максимальное значение слагаемого $\tilde{a}p^j v_j$ достигается при $v = \operatorname{sgn}(ap^j)$ и равно $|ap^j|$.

Введем матрицу A, состоящую из векторов-строк $\tilde{a}$, построенных для всех наборов линейно независимых (л.н.з.) столбцов $p^{j_1}, \ldots, p^{j_{n-1}}$ матрицы P, и вектор b, состоящий из чисел $\tilde{\beta}$, соответствующих строкам $\tilde{a}$. Также введем множество

$$\bar{\mathbb{Y}} = \{y \in \mathbb{R}^n | \ -b \leqslant Ay \leqslant b\}.$$

Теорема 2.1. *$y \in \mathbb{Y}$ тогда и только тогда, когда $y \in \bar{\mathbb{Y}}$.*

Доказательство. *Необходимость.* Пусть вектор $y \in \mathbb{Y}$. Значит, существует вектор v такой, что $y = Pv$ и $v \in \mathbb{V}$. Покажем, что $-b \leqslant Ay \leqslant b$.

Подставляя в это неравенство выражение Pv вместо y, для строки l получим

$$-\beta_l \leqslant a_l P v \leqslant \beta_l,$$

или, что то же,

$$-\sum_{j=1}^{R} |a_l p^j| \leqslant \sum_{j=1}^{R} a_l p^j v_j \leqslant \sum_{j=1}^{R} |a_l p^j|.$$

А поскольку $v_j \in [-1;1]$, $j = \overline{1,R}$, неравенство, очевидно, выполняется для любого $l = \overline{1,k}$, а значит, $y \in \bar{\mathbb{Y}}$.

Достаточность.

I. Рассмотрим матрицу A, размерность которой $k \times n$, где k — количество различных наборов из $n-1$ линейно независимых столбцов P. Для начала покажем, что $k \geqslant n$. Поскольку $\operatorname{rank} P = n$, то в матрице P найдется n л.н.з. столбцов. Тогда из них можно составить n наборов по $n-1$ л.н.з. столбцов в каждом.

Теперь покажем, что $\operatorname{rank} A = n$. Составим матрицу $\bar{P}$ из n л.н.з столбцов матрицы P: $\bar{P} = (p^{j_1}, \ldots, p^{j_n})$. По построению в матрице A найдутся такие n строк $a_{l_1}, \ldots, a_{l_n}$, что

$$a_{l_i} p^{j_1} = \cdots = a_{l_i} p^{j_{i-1}} = a_{l_i} p^{j_{i+1}} = \cdots = a_{l_i} p^{j_n} = 0, \quad i = \overline{1,n}.$$

Составим из этих строк матрицу $\bar{A}$. Тогда

$$\bar{A}\bar{P} = \begin{pmatrix} a_{l_1} p^{j_1} & 0 & \cdots & 0 \\ 0 & a_{l_2} p^{j_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{l_n} p^{j_n} \end{pmatrix}.$$

Далее предположим, что $a_{l_i} p^{j_i} = 0$. Это означает, что $a_{l_i}\bar{P} = (0\,0\,\ldots\,0)$, из чего следует линейная зависимость столбцов матрицы $\bar{P}$, что противоречит условиям. Потому $a_{l_i} p^{j_i} \neq 0 \ \forall i = \overline{1,n}$. Тогда $\det(\bar{A}\bar{P}) \neq 0$, следовательно, $\det(\bar{A}) \neq 0$, или, что то же, $\operatorname{rank} \bar{A} = n$.

Так как матрица $\bar{A}$ составлена из строк матрицы A, то $\operatorname{rank} A \geqslant n$. Но поскольку A содержит n столбцов, то $\operatorname{rank} A = n$.

II. Для доказательства достаточности стоит показать: для любой точки $y \in \bar{\mathbb{Y}}$ найдется такая точка $v \in \mathbb{V}$, что будет справедливо $y = Pv$. Для начала докажем это для вершин множества $\bar{\mathbb{Y}}$.

Выберем такую точку $\hat{y}$ из пространства $\mathbb{R}^n$, что выполняется $-b \leqslant A\hat{y} \leqslant b$. При этом среди данных неравенств существуют n таких ограничений, что

1. Строки $a_{l_1}, \ldots, a_{l_n}$, порождающие эти неравенства, л.н.з.
2. В точке $\hat{y}$ эти неравенства переходят в равенства по одному из двусторонних ограничений. Не умаляя общности, будем считать, что в равенства переходят верхние ограничения: $a_{l_i}\hat{y} = \beta_{l_i}, i = \overline{1,n}$.

При таких условиях точка $\hat{y}$ будет являться вершиной многогранника $\bar{\mathbb{Y}}$.

Поскольку $\operatorname{rank} P = \operatorname{rank}\{P, \hat{y}\} = n$, система уравнений $Pv = \hat{y}$ будет иметь по крайней мере одно решение. Необходимо показать существование решения при $v \in \mathbb{V}$.

Предположим от противного: для любого вектора v, являющегося решением системы $Pv = \hat{y}$, найдется номер j^*, что $|v_{j^*}| > 1$.

III. Определим вектор $\hat{v} : P\hat{v} = \hat{y}$ так, чтобы он обладал рядом удобных свойств. Для этого выберем некоторое равенство $a_{l_i}\hat{y} = \beta_{l_i}$. Для удобства обозначим $\bar{a} = a_{l_i}$, $\bar{\beta} = \beta_{l_i}$.

Столбцы матрицы P можно разделить на два класса: ортогональные и неортогональные со строкой $\bar{a}$. Пусть число R_1 обозначает количество ортогональных столбцов. Из свойств строки $\bar{a}$ и матрицы P следует, что $R_1 \geqslant n-1$, но при этом существует по крайней мере один неортогональный столбец. Для простоты переставим столбцы P таким образом, что сначала будут идти ортогональные:

$$\begin{cases} \bar{a}p^j = 0, \ j = \overline{1, R_1}, \\ \bar{a}p^j \neq 0, \ j = \overline{R_1 + 1, R}. \end{cases}$$

Теперь положим, что $\hat{v}_j = \operatorname{sgn}(\bar{a}p^j)$, $j = \overline{R_1+1, R}$. В таком случае равенство $\bar{a}\hat{y} = \bar{\beta}$ будет выполняться вне зависимости от значений остальных $\hat{v}_j$:

$$\bar{a}\hat{y} = \bar{a}P\hat{v} = \sum_{j=1}^{R} \bar{a}p^j\hat{v}_j = \sum_{j=R_1+1}^{R} \bar{a}p^j \operatorname{sgn}(\bar{a}p^j) = \sum_{j=R_1+1}^{R} |\bar{a}p^j| = \bar{\beta}.$$

Далее зададимся вопросом нахождения $\hat{v}_1, \ldots, \hat{v}_{R_1}$. По построению $\bar{a}$ известно, что существует набор из $n-1$ л.н.з. столбцов матрицы P, ортогональных со строкой $\bar{a}$. Следовательно, $\operatorname{rank}\{p^1, \ldots, p^{R_1}\} = n-1$. Тогда можно выделить $n-1$ столбцов, как базис этого подпространства, и выразить через них остальные столбцы. Пусть базисными будут столбцы $p^1, \ldots, p^{n-1}$:

$$c_{1,j}p^1 + \cdots + c_{n-1,j}p^{n-1} + p^j = 0, \quad j = \overline{n, R_1}. \tag{2.16}$$

Далее введем обозначения:

$$\hat{A} = \begin{pmatrix} a_{l_1} \\ \vdots \\ a_{l_{i-1}} \\ a_{l_{i+1}} \\ \vdots \\ a_{l_n} \end{pmatrix}, \quad \hat{b} = \begin{pmatrix} \beta_{l_1} - \sum_{j=R_1+1}^{R} a_{l_1}p^j\hat{v}_j \\ \vdots \\ \beta_{l_{i-1}} - \sum_{j=R_1+1}^{R} a_{l_{i-1}}p^j\hat{v}_j \\ \beta_{l_{i+1}} - \sum_{j=R_1+1}^{R} a_{l_{i+1}}p^j\hat{v}_j \\ \vdots \\ \beta_{l_n} - \sum_{j=R_1+1}^{R} a_{l_n}p^j\hat{v}_j \end{pmatrix}.$$

Тогда имеет место система из $n-1$ уравнений с R_1 неизвестными:

$$\hat{A}(p^1\hat{v}_1 + \cdots + p^{R_1}\hat{v}_{R_1}) = \hat{b}. \tag{2.17}$$

Будем называть *базисными переменными* те $\hat{v}_j$, которые стоят при столбцах, входящих в базис подпространства, а решение системы (2.17) *базисным*, если значения небазисных переменных равны нулю. Дополнительно обозначим множество индексов базисных переменных символом B, а небазисных — NB.

Лемма 2.1. *Если для любого решения системы* (2.17) *найдется номер* j^*, *что* $|\hat{v}_{j^*}| > 1$, *то существуют такое базисное решение* $\tilde{v}$ *и номер* j', *что* $|\tilde{v}_{j'}| > 1 + \sum_{j \in NB} |c_{j',j}|$.

Доказательство. Для простоты пусть $B = \{1, \ldots, n-1\}$, а тогда $NB = \{n, n+1, \ldots R_1\}$. Введем матрицу P_1, состоящую из столбцов $(p^1, \ldots, p^{n-1})$.

Поскольку матрица $\hat{A}P_1$ имеет размер $(n-1) \times (n-1)$ и ее ранг равен $n-1$, то для заданного базиса система (2.17) имеет единственное решение. Найдем его и обозначим $\tilde{v}_1, \ldots, \tilde{v}_{n-1}$.

Докажем лемму методом индукции в зависимости от количества небазисных столбцов.

1. Пусть $R_1 = n - 1$. По условию леммы существует такое j^*, что $|\tilde{v}_{j^*}| > 1$.
2. Пусть $R_1 = s$, $s > n-1$ и существуют базис $\{1, \ldots, n-1\}$ и номер j_1, что $|\tilde{v}_{j_1}| > 1 + \sum_{j=n}^{s-1} |c_{j_1,j}|$. Необходимо показать, что есть такое базисное решение $\tilde{v}$ и номер j', что $|\tilde{v}_{j'}| > 1 + \sum_{j \in NB} |c_{j',j}|$.

Перепишем (2.17), подставив базисное решение:

$$\hat{A}(p^1\tilde{v}_1 + \cdots + p^{n-1}\tilde{v}_{n-1}) = \hat{b}.$$

В левой части равенства вычтем и прибавим выражение $\hat{A}p^s\lambda_s$, где λ_s — некоторый параметр:

$$\hat{A}(p^1\tilde{v}_1 + \cdots + p^{n-1}\tilde{v}_{n-1} - p^s\lambda_s + p^s\lambda_s) = \hat{b}.$$

Теперь для вычитаемой части представим p^s в виде линейной комбинации столбцов $p^1, \ldots, p^{n-1}$ согласно (2.16):

$$\hat{A}\left(p^1(\tilde{v}_1 + c_{1,s}\lambda_s) + \cdots + p^{n-1}(\tilde{v}_{n-1} + c_{n-1,s}\lambda_s) + p^s\lambda_s\right) = \hat{b}.$$

Таким образом, получим новое решение системы (2.17), зависящее от параметра λ_s:

$$\hat{v}_j = \begin{cases} \tilde{v}_j + c_{j,s}\lambda_s, \;\; j = \overline{1,n-1}, \\ 0, \;\; j = \overline{n,s-1}, \\ \lambda_s, \;\; j = s. \end{cases}$$

Будем рассматривать λ_s в пределах от -1 до 1.

По условию существует такой номер j_1, что $|\tilde{v}_{j_1}| > \mu$, где $\mu = 1 + \sum_{j=n}^{s-1} |c_{j_2,j}|$. Рассмотрим, что произойдет с числом $\hat{v}_{j_1}$ при варьировании λ_s. Не умаляя общности, предположим, что $|\hat{v}_{j_1}|$ возрастает при увеличении λ_s, т. е. $|\tilde{v}_{j_1} + c_{j_1,s}| = |\tilde{v}_{j_1}| + |c_{j_1,s}|$. Тогда возможны два варианта:

а) При уменьшении λ_s до -1 абсолютное значение $\hat{v}_j$ остается больше μ. Следовательно, $|\tilde{v}_{j_1} - c_{j_1,s}| = |\tilde{v}_{j_1}| - |c_{j_1,s}| > 1 + \sum_{j=n}^{s-1} |c_{j_1,j}|$. В таком случае лемма доказана и $j' = j_1$;

б) Найдется число λ_s^1 в пределах $(-1, 0)$, что $|\tilde{v}_{j_1} + \lambda_s^1 c_{j_1,s}| = \mu$. В этом случае по условию леммы при $\lambda_s = \lambda_s^1$ существует номер j_2, для которого справедливо $|\tilde{v}_{j_2} + \lambda_s^1 c_{j_2,s}| > \mu$. Тогда посмотрим, в каком направлении изменяется $|\hat{v}_{j_2}|$ при стремлении λ_s к -1. Если уменьшается, то по аналогии с j_1 либо $j' = j_2$, либо есть число λ_s^2 $(-1 < \lambda_s^2 < \lambda_s^1)$, для которого выполняется равенство $|\tilde{v}_{j_2} + \lambda_s^2 c_{j_2,s}| = \mu$. При этом найдется номер j_3 такой, что $|\tilde{v}_{j_3} + \lambda_s^2 c_{j_3,s}| > \mu$. Продолжая по той же логике, для некоторого номера j_k окажется, что $|\tilde{v}_{j_k} - c_{j_k,s}| > \mu$, следовательно, $j' = j_k$.

Если же значение $|\hat{v}_{j_2}|$ возрастает при уменьшении λ_s, т. е. $|\tilde{v}_{j_2} - c_{j_2,s}| = |\tilde{v}_{j_2}| + |c_{j_2,s}|$, то с учетом непрерывной зависимости $\hat{v}_{j_1}$ и $\hat{v}_{j_2}$ от λ_s можно определить небольшое положительное число ε, для которого справедлива система

$$\begin{cases} \left|\tilde{v}_{j_1} + (\lambda_s^1 + \varepsilon) c_{j_1,s}\right| = |\tilde{v}_{j_1}| + (\lambda_s^1 + \varepsilon)|c_{j_1,s}| > \mu, \\ \left|\tilde{v}_{j_2} + (\lambda_s^1 + \varepsilon) c_{j_2,s}\right| = |\tilde{v}_{j_2}| - (\lambda_s^1 + \varepsilon)|c_{j_2,s}| > \mu. \end{cases}$$

Тогда умножим второе неравенство на $\left|\frac{c_{j_1,s}}{c_{j_2,s}}\right|$ и прибавим к первому:

$$|\tilde{v}_{j_1}| + \left|\frac{c_{j_1,s}}{c_{j_2,s}}\right| |\tilde{v}_{j_2}| > \mu \left(1 + \left|\frac{c_{j_1,s}}{c_{j_2,s}}\right|\right). \tag{2.18}$$

Теперь необходимо перестроить базис — внести столбец p^s вместо p^{j_2}:

$$B_s = B\backslash\{j_2\} \cup \{s\}, \quad NB_s = NB \cup \{j_2\}\backslash\{s\}.$$

Для этого нужно найти такое λ_s, что $\hat{v}_{j_2}$ обратится в нуль. Новое базисное решение можно выразить через старое следующим образом:

$$\tilde{v}^{(s)} = \begin{cases} \tilde{v}_j - \frac{c_{j,s}}{c_{j_2,s}}\tilde{v}_{j_2}, & j \in B_s\backslash\{s\}, \\ -\frac{1}{c_{j_2,j}}\tilde{v}_{j_2}, & j = s, \\ 0, & j = j \in NB_s. \end{cases}$$

Также изменится представление небазисных столбцов через базисные:

$$\sum_{k \in B_1} c_{k,j}^{(s)} p^k + p^j = 0 \quad \forall j \in NB_1,$$

где

$$c_{k,j}^{(s)} = \begin{cases} c_{k,j} - c_{j_2,j}\frac{c_{k,s}}{c_{j_2,s}}, & \forall k \in B_s\backslash\{s\} \quad \forall j \in NB_s\backslash\{j_2\}, \\ -c_{j_2,j}\frac{1}{c_{j_2,s}}, & k = s, \quad \forall j \in NB_s\backslash\{j_2\}, \\ \frac{c_{k,s}}{c_{j_2,s}}, & \forall k \in B_s\backslash\{s\}, \quad j = j_2, \\ \frac{1}{c_{j_2,s}}, & k = s, \quad j = j_2. \end{cases}$$

Следовательно, необходимо показать: в базисном решении $\hat{v}_j$ существует такой номер j', что $|\tilde{v}_{j'}^{(s)}| > 1 + \sum_{j \in NB_s} \left|c_{j',j}^{(s)}\right|$.

Из выдвинутых ранее предположений известно, что $\tilde{v}_{j_1} c_{j_1,s} > 0$, а $\tilde{v}_{j_2} c_{j_2,s} < 0$. Отсюда можно сделать вывод, что $\tilde{v}_{j_1}\left(\frac{c_{j_1,s}}{c_{j_2,s}}\tilde{v}_{j_2}\right) < 0$, а потому $|\tilde{v}_{j_1}^{(s)}| = \left|\tilde{v}_{j_1} - \frac{c_{j_1,s}}{c_{j_2,s}}\tilde{v}_{j_2}\right| = |\tilde{v}_{j_1}| + \left|\frac{c_{j_1,s}}{c_{j_2,s}}\tilde{v}_{j_2}\right|$. Используя (2.18), получим неравенство

$$|\tilde{v}_{j_1}^{(s)}| > \left(1 + \sum_{j=n}^{s-1} |c_{j_2,j}|\right)\left(1 + \left|\frac{c_{j_1,s}}{c_{j_2,s}}\right|\right) > 1 + \left|\frac{c_{j_1,s}}{c_{j_2,s}}\right| + \sum_{j=n}^{s-1} \left|c_{j_2,j}\frac{c_{j_1,s}}{c_{j_2,s}}\right|,$$

или в других обозначениях:

$$|\tilde{v}_{j_1}^{(s)}| > 1 + \left|c_{j_1,j_2}^{(s)}\right| + \sum_{j=n}^{s-1} \left|c_{j_1,j}^{(s)}\right| = \sum_{j \in NB_s} \left|c_{j_1,j}^{(s)}\right|.$$

Для этого случая $j' = j_1$. Таким образом, лемма доказана, и можем вернуться к доказательству основной теоремы. □

Для простоты определим базисными переменными $\tilde{v}_1, \ldots, \tilde{v}_{n-1}$. Также положим $j' = 1$, а $c_{j',j}$ переобозначим как c_j. Тогда можем доопределить вектор $\hat{v}$:

$$\begin{cases} \hat{v}_1 > 1 + \sum_{j=n}^{R_1} |c_j|, \\ \hat{v}_j = 0, \ j = \overline{n, R_1}. \end{cases}$$

IV. Покажем, что среди строк матрицы A существует строка a^* такая, что

1. $a^* p^2 = a^* p^3 = \cdots = a^* p^{n-1} = 0$.
2. $\operatorname{sgn}(a^* p^1) = \operatorname{sgn}(\hat{v}_1)$.
3. $(a^* p^j)(\bar{a} p^j) \geqslant 0 \quad \forall j = \overline{1,R}$.

Известно, что $\operatorname{rank}\{p^1, p^2, \ldots p^{n-1}\} = n-1$. Поскольку $\operatorname{rank} P = n$, то в этой матрице найдется такой столбец p^s, что $\operatorname{rank}\{p^1, p^2, \ldots p^{n-1}, p^s\} = n$, а тогда $\operatorname{rank}\{p^2, \ldots p^{n-1}, p^s\} = n-1$.

По построению в матрице A существует такая строка a, что $ap^2 = ap^3 = \cdots = ap^{n-1} = ap^s = 0$, но $ap^1 \neq 0$.

Благодаря тому, что неравенства $-\beta_l \leqslant a_l y \leqslant \beta_l$ инвариантны относительно нуля, в матрице A можно заменить строку a_l строкой $-a_l$. Следовательно, можно добиться выполнения $ap^1 \hat{v}_1 > 0$.

Таким образом показано существование вектора a, удовлетворяющего требованиям 1 и 2. Выполнение требования 3 покажем по индукции.

База. для $j = \overline{1, n-1}$ выполняется очевидно, поскольку $\bar{a} p^j = 0$.

Шаг индукции. Пусть $(ap^j)(\bar{a}p^j) \geqslant 0$ для $j = \overline{1,k-1}$, но $(ap^k)(\bar{a}p^k) < 0$. Тогда существуют два неотрицательных числа γ_1 и γ_2, для которых выполняется $\gamma_1 ap^k + \gamma_2 \bar{a} p^k = 0$. Введем строку

$a^* = \gamma_1 a + \gamma_2 \bar{a}$, в этом случае $a^* p^k = 0$. Поскольку $\bar{a}$ ортогональна с векторами $p^1, p^2, \ldots, p^{n-1}$, а, в свою очередь, a ортогональна с векторами $p^2, \ldots, p^{n-1}, p^s$, то справедливо

$$a^* p^j = (a + \bar{a}) p^j = 0 \quad \forall j = \overline{2, n-1}.$$

При этом $a^* p^1 \hat{v}_1 = a p^1 \hat{v}_1 > 0$. Это означает, что строка a^* удовлетворяет требованиям 1 и 2.

Кроме того, при $\forall j = \overline{1,k}$

$$(a^* p^j)(\bar{a} p^j) = (\gamma_1 a p^j + \gamma_2 \bar{a} p^j)(\bar{a} p^j) = \gamma_1 (a p^j)(\bar{a} p^j) + \gamma_2 (\bar{a} p^j)(\bar{a} p^j) \geqslant 0.$$

Осталось показать, что a^* входит в матрицу A. В матрице A существует строка a_l, построенная как решение системы из $n-1$ уравнений:

$$a_l p^2 = \cdots = a_l p^{n-1} = a_l p^k = 0.$$

Поскольку решение данной системы представляет собой однопараметрическое семейство, и a^* является её решением, то a_l и a^* совпадают с точностью до множителя. Так как множитель не влияет на выполнение свойств (и строку всегда можно умножить на -1), можно говорить о том, что a^* является строкой матрицы A.

Таким образом, по индукции показано, что найдется строка a^*, удовлетворяющая трем требованиям.

V. Найдем, чему равняется $\beta^* = \max\limits_{v \in \mathbb{V}} (a^* P v)$. Ранее было определено, как небазисные столбцы выражаются через базисные (2.16). Воспользуемся тем фактом, что $a^* p^j = 0$ для $j = \overline{2, n-1}$. Умножив равенства (2.16) на a^* слева, получим равенства

$$a^* p^j = -c_j a^* p^1, \ \forall j = \overline{n, R_1}.$$

Тогда

$$\beta^* = a^* p^1 \operatorname{sgn}\left(a^* p^1\right) + \sum_{j=n}^{R_1} \left(-c_j a^* p^1\right) \operatorname{sgn}\left(-c_j a^* p^1\right) + \sum_{j=R_1+1}^{R} \left|a^* p^j\right| =$$

$$= \left|a^* p^1\right| \left(1 + \sum_{j=n}^{R_1} |c_j|\right) + \sum_{j=R_1+1}^{R} \left|a^* p^j\right|.$$

Теперь найдем значение $a^* P\tilde{v}$. Поскольку $a^* p^j = 0,\ j = \overline{2,n-1}$, $\hat{v}_j = 0,\ j = \overline{n,R_1}$, и $\hat{v}_j = \operatorname{sgn}(\bar{a} p^j),\ j = \overline{R_1+1,R}$, получим выражение

$$a^* P\hat{v} = a^* p^1 \hat{v}_1 + \sum_{j=R_1+1}^{R} a^* p^j \operatorname{sgn}(\bar{a} p^j).$$

Поскольку $(a^* p^j)(\bar{a} p^j) > 0$, то $\operatorname{sgn}(a^* p^j) = \operatorname{sgn}(\bar{a} p^j)$. Перепишем предыдущее равенство с учетом $a^* P v_1 > 0$:

$$a^* P\hat{v} = \left|a^* p^1\right| |\hat{v}_1| + \sum_{j=R_1+1}^{R} \left|a^* p^j\right|.$$

Наконец, оценим выражение $a^* P\hat{v} - \beta^*$:

$$a^* P\hat{v} - \beta^* = \left|a^* p^1\right| \left(|\hat{v}_1| - 1 - \sum_{j=n}^{R_1} |c_j|\right) > 0.$$

Следовательно, $a^* P\hat{v} > \beta^*$, а так как $\hat{y} = P\hat{v}$, то $a^*\hat{y} > \beta^*$. Таким образом, показано противоречие: $\hat{y} \notin \bar{\mathbb{Y}}$. Значит, существует $\hat{v}$ такое, что $\hat{y} = P\hat{v}$ и $\hat{v} \in \mathbb{V}$.

VI. Было показано, что вершины многогранника $\bar{\mathbb{Y}}$ принадлежат множеству $\mathbb{Y}$, обозначим вершины $y^1, \ldots, y^h$.

Известно, что многогранник может быть описан как выпуклая оболочка своих вершин, т. е. $y \in \bar{\mathbb{Y}}$ тогда и только тогда, когда $y = \sum_{i=1}^{h} \lambda_i y^i$, где $\sum_{i=1}^{h} \lambda_i = 1, \lambda_i \geqslant 0$. Так как $y^i \in \mathbb{Y}$, то существует вектор $v^i \in \mathbb{V}$, для которого $y^i = Pv^i$. Значит, $y = \sum_{i=1}^{h} \lambda_i y^i = \sum_{i=1}^{h} \lambda_i P v^i = Pv$.

Поскольку $\mathbb{V}$ — выпуклое множество, то выпуклая оболочка точек, принадлежащих ему, входит в его состав. Следовательно, $v = \sum_{i=1}^{h} v^i \in \mathbb{V}$, поэтому $y \in \mathbb{Y}$. В результате показано, что если $y \in \bar{\mathbb{Y}}$, то $y \in \mathbb{Y}$. $\square$

2.7 Анализ алгоритма

В итоге алгоритм построения множества достижимости (управляемости) задачи (1.1)–(1.4) состоит из следующих шагов:

1. Переход от задачи построения множества $X_1(t, \hat{t}, \hat{x})$ или $X_2(t, \hat{t}, \hat{x})$ к задаче линейного отображения R-мерного куба в n-мерное пространство (2.8).
2. Исключение линейно-зависимых уравнений из системы $y = Pv$. Для этого можно воспользоваться методом Гаусса. В случае $\operatorname{rank} P = n_1$, то будет получено $n - n_1$ линейных уравнений относительно y.
3. Перебор всех совокупностей из $n_1 - 1$ линейно независимых столбцов матрицы P. Для каждой такой совокупности столбцов вычисляется вектор-строка $\tilde{a}$ — некоторое ненулевое решение однородной системы линейных алгебраических уравнений (СЛАУ) с транспонированной матрицей, состоящей из этих столбцов (2.14). Для нахождения $\tilde{a}$ можно воспользоваться методом Гаусса или любым другим методом решения СЛАУ.
4. Вычисление значений $\tilde{\beta}$ для строк $\tilde{a}$ (2.15). Далее формируется матрица A, состоящая из всех найденных $\tilde{a}$, и соответствующий вектор b из чисел $\tilde{\beta}$. Система неравенств $-b \leqslant A \leqslant b$ является результатом отображения (2.8).
5. Возврат от переменных y к x. В итоге множество достижимости $X_1(t, \hat{t}, \hat{x})$ или множество управляемости $X_2(t, \hat{t}, \hat{x})$ описывается как система из n_1 линейных уравнений и ряда двусторонних линейных неравенств.

Максимально возможное количество строк в матрице A — число сочетаний $C_R^{n_1}$. Найти строки $\tilde{a}$ можно как решение системы (2.14) или как обобщение векторного произведения на n_1-мерное пространство. Напри-

мер, для трехмерного пространства строку $\tilde{a}$ для столбцов p^{j_1} и p^{j_2} можно найти как определитель:

$$\tilde{a} = \begin{vmatrix} \vec{i} & p_1^{j_1} & p_1^{j_2} \\ \vec{j} & p_2^{j_1} & p_2^{j_2} \\ \vec{k} & p_3^{j_1} & p_3^{j_2} \end{vmatrix}.$$

Таким образом, задачу можно свести к подсчету $C_R^{n_1}$ n_1-мерных определителей.

2.8 Выводы по главе 2

Предложен алгоритм построения множеств достижимости и управляемости для линейной задачи управления с двусторонними ограничениям. Алгоритм состоит в переходе от задачи (1.1)–(1.4) к системе (2.8), построении системы линейных неравенств относительно y и обратному переходу к переменным $x(t)$. В результате найдено множество достижимости (управляемости), заданное системой линейных ограничений.

Кроме непосредственного использования этих множеств, существуют еще два полезных применения:

1. В задаче управления с внешним возмущением, которое может вносить существенные коррективы в движение управляемого объекта, а управление в связи с этим периодически перестраивается в режиме реального времени [7], может быть полезно при каждом перестроении оценивать, является ли желаемое положение системы достижимым.
2. Нелинейные задачи оптимального управления могут решаться с помощью линеаризаций системы. Так, в главе 4 будет описан алгоритм построения управления для нелинейной системы, состоящий в последовательной линеаризации системы и построении

управления уже для линейной системы. Однако из-за неточности линеаризаций в какой-то момент может оказаться, что желаемое финальное состояние не достижимо. Вероятность возникновения этой проблемы можно минимизировать, если на каждом шаге оценивать, как далеко финальное состояние находится от границы множества достижимости.

ГЛАВА 3. ПОСТРОЕНИЕ ОПТИМАЛЬНЫХ УПРАВЛЕНИЙ С УЧЕТОМ ОГРАНИЧЕНИЙ

В данной главе поставим задачу изучить возможности учета дополнительных ограничений на управление и фазовые переменные для линейной терминальной задачи (1.1)–(1.3) с линейной целевой функцией (1.5) или с квадратичным функционалом (1.7).

В параграфе 3.1 будет подробно рассмотрен вопрос выбора класса допустимых управлений. Кроме класса кусочно-постоянных функций (1.4), выбранного основным для диссертационной работы, интерес также представляют управления в виде кусочно-линейных и кусочно-квадратичных функций. Использование более общих классов приводит к усложнению решения задачи, но при этом можно накладывать дополнительные условия на непрерывность и гладкость. Например, для задачи управления колесными мобильными роботами, которые применяются для транспортировки людей с ограниченными физическими возможностями, гладкость управляющих воздействий является важнейшим критерием [72].

В статье [10] был предложен метод управления в классе квадратичных сплайнов. Результаты, представленные в параграфе 3.1, являются развитием изложенного подхода.

В параграфе 3.2 будет изучена задача управления с дополнительными ограничениями на управления и фазовые переменные. Сначала будут рассмотрены линейные ограничения на управления, затем рассмотрен вопрос построения управления для невыпуклых множеств допустимых управлений. Далее будет освещена задача с ограничениями уже на фазовые переменные. Отметим, что задача построения управления при выпуклых линейных ограничениях на фазовые переменные в заданные моменты времени была проработана в статье [73].

3.1 Ограничения на класс управляющих функций

В данном параграфе рассматривается задача управления линейной системой при различных ограничениях на класс управления. Наибольший интерес представляют три класса:

1. Управление в виде кусочно-постоянной функции.
2. Управление в виде кусочно-линейной функции.
3. Управление в виде кусочно-квадратичной функции.

Выбор таких классов обусловлен удобством решения задачи при сохранении эффективности. При этом рассмотрение управления в одном из заданных классов позволяет добиться кусочной непрерывности, непрерывности или гладкости управлений, что зачастую является важным требованием, вытекающим из физических свойств исследуемых процессов.

Так, выбор управления в первом классе позволяет добиться кусочной непрерывности, во втором классе — кусочной непрерывности или непрерывности, в третьем классе — кусочной непрерывности, непрерывности или же гладкости.

Класс кусочно-постоянных функций (1.4) был подробно освещен в главе 1. Сейчас сосредоточимся на альтернативных вариантах выбора функций управления.

3.1.1 Линейная задача с управлением в виде кусочно-линейной функции

Рассмотрим случай, в котором компоненты вектора управлений являются кусочно-линейными функциями:

$$u_i(t) = \begin{cases} p_{ik}^{(1)} t + p_{ik}^{(0)}, & t \in [t_{k-1},\ t_k), \\ k = \overline{1, N}, \end{cases} \qquad i = \overline{1,r}. \tag{3.1}$$

где $t_k = kh$.

Покажем алгоритм сведения задачи оптимального управления (1.1)–(1.3), (3.1) с целевой функцией (1.5) или (1.7) к задаче математического программирования.

Сведение условий на непрерывность

Так как управления представлены в виде кусочно-непрерывных функций, разрывы возможны только в узлах. Поэтому необходимо добавить условия на непрерывность на стыках временных сегментов:

$$p_{ik}^{(1)} t_k + p_{ik}^{(0)} = p_{ik+1}^{(1)} t_k + p_{ik+1}^{(0)}, \quad k = \overline{1,N-1}, \quad i = \overline{1,r}. \tag{3.2}$$

Таким образом, условия на непрерывность были сведены к $r(N-1)$ линейным равенствам относительно переменных $p_{ik}^{(0)}$ и $p_{ik}^{(1)}$.

Если по условию задачи не требуется непрерывность в узлах, стоит отключить ограничения (3.2).

Сведение терминальных условий

Условия для начального состояния объекта всегда выполняются, поскольку являются частью задачи Коши. Определим условия на конечное состояние.

Выпишем формулу Коши для системы (1.1) в финальный момент времени:

$$x(T) = Y(T)x_* + \int\limits_0^T Y(T)Y^{-1}(t)\left(B(t)u(t) + d(t)\right)dt.$$

Разделим отрезок времени $[0,\ T]$ на N сегментов одинаковой длины и применим представление управления в виде кусочно-линейных функций (3.1):

$$x(T) = Y(T)x_* + \int\limits_0^T Y(T)Y^{-1}(t)d(t)dt +$$

$$+\sum_{i=1}^{r}\sum_{k=1}^{N}\left[\left(\int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)tdt\right)p_{ik}^{(1)} + \left(\int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i dt\right)p_{ik}^{(0)}\right].$$

В пункте 3.1.3 рассмотрим вопрос вычисления интегралов вида

$$\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i dt, \quad \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i t dt,$$

а пока переобозначим их, имея в виду, что они являются константами:

$$\begin{aligned} b^{i,k,0} &:= \int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt, \\ b^{i,k,1} &:= \int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)tdt, \end{aligned} \qquad k = \overline{1,N}, \quad i = \overline{1,r},$$

Дополнительно введем обозначение:

$$d^0 = Y(T)x_* + \int\limits_0^T Y(T)Y^{-1}(t)d(t)dt.$$

Тогда можем выписать представление $x(T)$ в виде линейной функции от переменных $p_{ik}^{(0)}$ и $p_{ik}^{(1)}$:

$$x(T) = d^0 + \sum_{i=1}^{r}\sum_{k=1}^{N}\left(b^{i,k,0}p_{ik}^{(0)} + b^{i,k,1}p_{ik}^{(1)}\right). \tag{3.3}$$

Подставим выражение (3.3) в граничное условие (1.2):

$$Hd^0 + \sum_{i=1}^{r}\sum_{k=1}^{N}\left(Hb^{i,k,0}p_{ik}^{(0)} + Hb^{i,k,1}p_{ik}^{(1)}\right) = g^0,$$

и введем ряд новых обозначений:

$$\begin{aligned} h^{i,k,0} &= Hb^{i,k,0}, \\ h^{i,k,1} &= Hb^{i,k,1}, \end{aligned} \quad k = \overline{1,N}, \quad i = \overline{1,r},$$

$$g = g^0 - Hd^0.$$

Теперь выпишем полученные условия:

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(h_l^{i,k,0}p_{ik}^{(0)} + h_l^{i,k,1}p_{ik}^{(1)}\right) = g_l, \quad l = \overline{1,m}. \tag{3.4}$$

Таким образом терминальные условия (1.2) перешли в m линейных равенств (3.4) с $2rN$ переменными.

Сведение прямых ограничений на управления

В соответствии с прямыми ограничениями (1.3) и представлением управляющих функций в виде (3.1), должны выполняться следующие условия:

$$l_i^{(1)} \leqslant p_{ik}^{(1)}t + p_{ik}^{(0)} \leqslant l_i^{(2)} \quad \forall t \in [t_{k-1},\ t_k), \quad k = \overline{1,N}, \quad i = \overline{1,r}.$$

Поскольку линейные функции являются монотонными, достаточно учесть данные неравенства на концах рассматриваемых отрезков времени:

$$\begin{aligned} l_i^{(1)} &\leqslant p_{ik}^{(1)}t_{k-1} + p_{ik}^{(0)} \leqslant l_i^{(2)}, \\ l_i^{(1)} &\leqslant p_{ik}^{(1)}t_k + p_{ik}^{(0)} \leqslant l_i^{(2)}, \end{aligned} \quad k = \overline{1,N}, \quad i = \overline{1,r}. \tag{3.5}$$

Следовательно, прямые ограничения (1.3) сведены к $4rN$ линейным неравенствам (3.5) от $2rN$ переменных $p_{ik}^{(0)}$ и $p_{ik}^{(1)}$.

Сведение целевых функций

Для сведения первой целевой функции воспользуемся представлением финального состояния объекта (3.3). Тогда можем записать критерий (1.5) как линейную функцию от переменных $p_{ik}^{(0)}$ и $p_{ik}^{(1)}$:

$$J_1 = c^T d^0 + \sum_{i=1}^{r} \sum_{k=1}^{N} \left(c^T b^{i,k,0} p_{ik}^{(0)} + c^T b^{i,k,1} p_{ik}^{(1)} \right).$$

Поскольку первое слагаемое является константой, можем удалить его из целевой функции, при этом имея в виду, что от этого может поменяться значение функционала:

$$J_1 = \sum_{i=1}^{r} \sum_{k=1}^{N} \left(c^{i,k,0} p_{ik}^{(0)} + c^{i,k,1} p_{ik}^{(1)} \right) \longrightarrow \min. \tag{3.6}$$

В (3.6) используются обозначения

$$\begin{aligned} c^{i,k,0} &= c^T b^{i,k,0}, \\ c^{i,k,1} &= c^T b^{i,k,1}, \end{aligned} \qquad k = \overline{1,N}, \quad i = \overline{1,r}.$$

Таким образом, (3.6) представляет собой линейный функционал относительно переменных $p_{ik}^{(0)}$ и $p_{ik}^{(1)}$.

Второй критерий (1.6) не будет рассмотрен для данного типа управления, поскольку нахождение модуля от кусочно-линейной функции требует дополнительных вычислений, выходящих за рамки данной работы.

Перейдем к третьему критерию. Заменим управление $u(t)$ кусочно-линейными функциями (3.1) в целевой функции (1.7):

$$J_2 = \sum_{k=1}^{N} \sum_{i_1=1}^{r} \sum_{i_2=1}^{r} \int_{t_{k-1}}^{t_k} q_{i_1 i_2} \left(p_{i_1 k}^{(1)} t + p_{i_1 k}^{(0)} \right) \left(p_{i_2 k}^{(1)} t + p_{i_2 k}^{(0)} \right) dt.$$

После вычисления интегралов и введения новых обозначений, получим квадратичную форму относительно переменных $p_{ik}^{(0)}$ и $p_{ik}^{(1)}$:

$$J_2 = \sum_{k=1}^{N} \sum_{i_1=1}^{r} \sum_{i_2=1}^{r} \left(h_{i_1 i_2 k}^{(1)} p_{i_1 k}^{(0)} p_{i_2 k}^{(0)} + \right.$$
$$\left. + h_{i_1 i_2 k}^{(2)} \left(p_{i_1 k}^{(1)} p_{i_2 k}^{(0)} + p_{i_1 k}^{(0)} p_{i_2 k}^{(1)} \right) + h_{i_1 i_2 k}^{(3)} p_{i_1 k}^{(1)} p_{i_2 k}^{(1)} \right), \tag{3.7}$$

где

$$h_{i_1 i_2 k}^{(\alpha)} := q_{i_1 i_2} \frac{1}{\alpha} \frac{T^\alpha}{N^\alpha} \left(k^\alpha - (k-1)^\alpha \right),$$
$$k = \overline{1,N}, \quad i_1 = \overline{1,r}, \quad i_2 = \overline{1,r}, \quad \alpha = \overline{1,3}.$$

В пункте 3.1.2 покажем, что построенная квадратичная форма является неотрицательно определенной.

Итоги

Было показано, как при выборе управления в виде кусочно-линейной функции можно свести линейную задачу оптимального управления (1.1)–(1.3), (3.1), (1.5) к задаче линейного программирования (3.2), (3.4)–(3.6):

$$\sum_{i=1}^{r} \sum_{k=1}^{N} \left(c^{i,k,0} p_{ik}^{(0)} + c^{i,k,1} p_{ik}^{(1)} \right) \longrightarrow \min$$
$$p_{ik}^{(1)} t_k + p_{ik}^{(0)} = p_{ik+1}^{(1)} t_k + p_{ik+1}^{(0)}, \quad k = \overline{1,N-1}, \quad i = \overline{1,r}$$
$$\sum_{i=1}^{r} \sum_{k=1}^{N} \left(h_l^{i,k,0} p_{ik}^{(0)} + h_l^{i,k,1} p_{ik}^{(1)} \right) = g_l, \quad l = \overline{1,m}$$
$$\begin{array}{l} l_i^{(1)} \leqslant p_{ik}^{(1)} t_{k-1} + p_{ik}^{(0)} \leqslant l_i^{(2)}, \\ l_i^{(1)} \leqslant p_{ik}^{(1)} t_k + p_{ik}^{(0)} \leqslant l_i^{(2)}, \end{array} \quad k = \overline{1,N}, \quad i = \overline{1,r}.$$

Данная задача имеет следующие характеристики:

– $2rN$ переменных,

– $r(N-1)+m$ линейных равенств,

– $4rN$ линейных неравенств,

– линейная целевая функция.

Также выше было показано, как задачу оптимального управления с линейной системой и квадратичным функционалом (1.1)–(1.3), (3.1), (1.7) можно свести к выпуклой задаче квадратичного программирования (3.2), (3.4), (3.5), (3.7):

$$\sum_{k=1}^{N}\sum_{i_1=1}^{r}\sum_{i_2=1}^{r}\left(h^{(1)}_{i_1i_2k}p^{(0)}_{i_1k}p^{(0)}_{i_2k}+h^{(2)}_{i_1i_2k}\left(p^{(1)}_{i_1k}p^{(0)}_{i_2k}+p^{(0)}_{i_1k}p^{(1)}_{i_2k}\right)+h^{(3)}_{i_1i_2k}p^{(1)}_{i_1k}p^{(1)}_{i_2k}\right)\to\min$$

$$p^{(1)}_{ik}t_k+p^{(0)}_{ik}=p^{(1)}_{ik+1}t_k+p^{(0)}_{ik+1},\quad k=\overline{1,N-1},\quad i=\overline{1,r}$$

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(h^{i,k,0}_{l}p^{(0)}_{ik}+h^{i,k,1}_{l}p^{(1)}_{ik}\right)=g_l,\quad l=\overline{1,m}$$

$$\begin{matrix}l^{(1)}_i\leqslant p^{(1)}_{ik}t_{k-1}+p^{(0)}_{ik}\leqslant l^{(2)}_i,\\ l^{(1)}_i\leqslant p^{(1)}_{ik}t_k+p^{(0)}_{ik}\leqslant l^{(2)}_i,\end{matrix}\qquad k=\overline{1,N},\quad i=\overline{1,r}.$$

Такая задача имеет следующие характеристики:

– $2rN$ переменных,

– $r(N-1)+m$ линейных равенств,

– $4rN$ линейных неравенств,

– выпуклая квадратичная целевая функция.

Пример. Демпфирование колебаний «спящего волчка» Лагранжа

Решим задачу демпфирования спящего волчка, описанную в параграфе 1.5, построив управление в классе кусочно-линейных функций.

Процедура сведения была реализована на языке Python 3, задача квадратичного программирования была решена с помощью пакета для оптимизации COPT (см. приложение А).

Общее время работы программы составило 0,57 секунд. Значение целевой функции равняется 91,47. Результаты приведены на рис. 3.1.

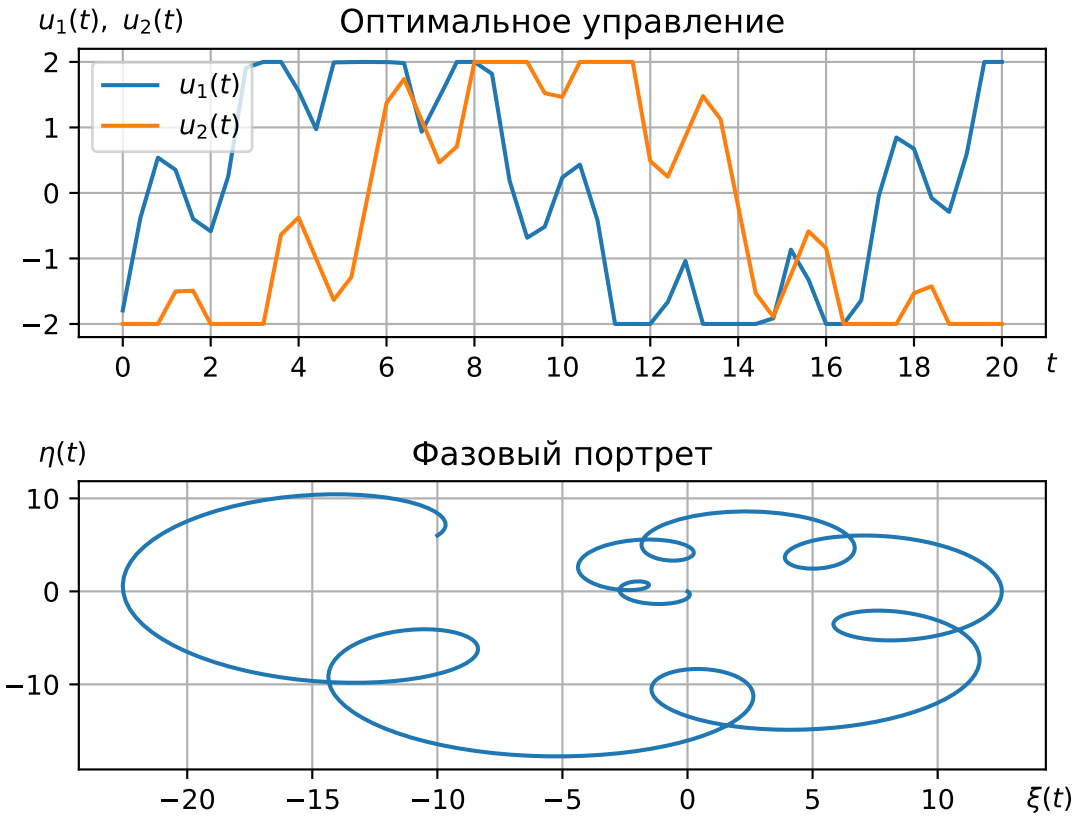

Рисунок 3.1 — Демпфирование колебаний «спящего волчка» Лагранжа при выборе управления в классе кусочно-линейных функций

3.1.2 Линейная задача с управлением в виде кусочно-квадратичной функции

Теперь предположим, что компоненты вектора управлений являются кусочно-квадратичными функциями или что то же квадратичными сплайнами:

$$u_i(t) = \begin{cases} p_{ik}^{(2)} t^2 + p_{ik}^{(1)} t + p_{ik}^{(0)}, \quad t \in [t_{k-1},\ t_k), \\ k = \overline{1, N}, \end{cases} \qquad i = \overline{1,r}. \tag{3.8}$$

где $t_k = kh$.

Покажем алгоритм сведения задачи оптимального управления (1.1)–(1.3), (3.8) с целевой функцией (1.5) или (1.7) к задаче математического программирования.

Сведение условий на гладкость

Поскольку управления представлены в виде квадратичных сплайнов, у нас гарантирована гладкость между узлами. Однако, необходимо дополнительно обеспечить непрерывность и гладкость в узлах (точках перехода от одной квадратичной функции к другой).

Непрерывность в узлах для управляющих функций:

$$\begin{gathered} p_{ik}^{(2)} t_k^2 + p_{ik}^{(1)} t_k + p_{ik}^{(0)} = p_{ik+1}^{(2)} t_k^2 + p_{ik+1}^{(1)} t_k + p_{ik+1}^{(0)}, \\ k = \overline{1,N-1}, \quad i = \overline{1,r}. \end{gathered} \tag{3.9}$$

Непрерывность в узлах для производных от управляющих функций:

$$2p_{ik}^{(2)} t_k + p_{ik}^{(1)} = 2p_{ik+1}^{(2)} t_k + p_{ik+1}^{(1)}, \quad k = \overline{1,N-1}, \quad i = \overline{1,r}. \tag{3.10}$$

Таким образом, условия на гладкость были сведены к $2r(N-1)$ линейным равенствам относительно переменных $p^{(0)}$, $p^{(1)}$, $p^{(2)}$.

Заметим, что если по условию задачи не требуется гладкость в узлах, мы можем отключить ограничения (3.9). Если непрерывность также не требуется, нужно к тому же отключить ограничения (3.10).

Сведение терминальных условий

Условия на начальное состояние объекта всегда выполняются, поскольку являются частью задачи Коши. Определим условия на конечное состояние.

Выпишем формулу Коши для системы (1.1) в финальный момент времени:

$$x(T) = Y(T)x_* + \int\limits_0^T Y(T)Y^{-1}(t)\left(B(t)u(t) + d(t)\right)dt.$$

Разделим отрезок времени $[0,\ T]$ на N сегментов одинаковой длины и применим представление управления в виде квадратичных сплайнов (3.8):

$$x(T) = Y(T)x_* + \int\limits_0^T Y(T)Y^{-1}(t)d(t)dt + \\ + \sum_{i=1}^{r}\sum_{k=1}^{N}\left[\left(\int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)t^2dt\right)p_{ik}^{(2)} + \right. \\ \left. + \left(\int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)tdt\right)p_{ik}^{(1)} + \left(\int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt\right)p_{ik}^{(0)}\right].$$

В пункте 3.1.3 рассмотрим вопрос вычисления интегралов вида

$$\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt,\ \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)tdt,\ \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)t^2dt,$$

а пока переобозначим их, учитывая что это константы:

$$b^{i,k,0} := \int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt,$$
$$b^{i,k,1} := \int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)tdt, \qquad k = \overline{1,N}, \quad i = \overline{1,r},$$
$$b^{i,k,2} := \int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)t^2dt,$$

Аналогично п. 3.1.1 введем параметр d^0:

$$d^0 = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)d(t)dt.$$

После этого можем выписать представление $x(T)$ в виде линейной функции от переменных $p^{(0)},\ p^{(1)},\ p^{(2)}$:

$$x(T) = d^0 + \sum_{i=1}^{r}\sum_{k=1}^{N}\left(b^{i,k,0}p_{ik}^{(0)} + b^{i,k,1}p_{ik}^{(1)} + b^{i,k,2}p_{ik}^{(2)}\right). \tag{3.11}$$

Теперь подставим выражение (3.11) в граничное условие (1.2):

$$Hd^0 + \sum_{i=1}^{r}\sum_{k=1}^{N}\left(Hb^{i,k,0}p_{ik}^{(0)} + Hb^{i,k,1}p_{ik}^{(1)} + Hb^{i,k,2}p_{ik}^{(2)}\right) = g^0,$$

и введем ряд новых обозначений:

$$\begin{aligned} h^{i,k,0} &= Hb^{i,k,0},\\ h^{i,k,1} &= Hb^{i,k,1}, \quad k = \overline{1,N}, \quad i = \overline{1,r},\\ h^{i,k,2} &= Hb^{i,k,2},\\ g &= g^0 - Hd^0. \end{aligned}$$

Выпишем полученные условия:

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(h_l^{i,k,0}p_{ik}^{(0)} + h_l^{i,k,1}p_{ik}^{(1)} + h_l^{i,k,2}p_{ik}^{(2)}\right) = g_l, \quad l = \overline{1,m}. \tag{3.12}$$

Таким образом терминальные условия (1.2) перешли в m линейных равенств с $3rN$ переменными (3.12).

Сведение прямых ограничений на управления

В соответствии с прямыми ограничениями (1.3) и представлением управляющих функций в виде (3.9), должны выполняться следующие ограничения:

$$l_i^{(1)} \leqslant p_{ik}^{(2)}t^2 + p_{ik}^{(1)}t + p_{ik}^{(0)} \leqslant l_i^{(2)} \quad \forall t \in [t_{k-1},\ t_k), \quad k = \overline{1,N}, \quad i = \overline{1,r}.$$

Они могут быть заменены экстремальными условиями

$$\begin{aligned} l_i^{(1)} &\leqslant \inf_{t\in[t_{k-1},\ t_k)} \left(p_{ik}^{(2)}t^2 + p_{ik}^{(1)}t + p_{ik}^{(0)}\right), \\ l_i^{(2)} &\geqslant \sup_{t\in[t_{k-1},\ t_k)} \left(p_{ik}^{(2)}t^2 + p_{ik}^{(1)}t + p_{ik}^{(0)}\right), \end{aligned} \qquad k = \overline{1,N}, \quad i = \overline{1,r}.$$

В таком случае на каждом из отрезков $[t_{k-1},\ t_k)$ необходимо рассмотреть лишь три точки подозрительные на экстремум: $t_{k-1},\ t_k$ и $-p_{ik}^{(1)}/2p_{ik}^{(2)}$.

Это приводит к ограничением на узловые точки:

$$\begin{aligned} l_i^{(1)} \leqslant p_{ik}^{(2)}t_{k-1}^2 + p_{ik}^{(1)}t_{k-1} + p_{ik}^{(0)} \leqslant l_i^{(2)}, \\ l_i^{(1)} \leqslant p_{ik}^{(2)}t_k^2 + p_{ik}^{(1)}t_k + p_{ik}^{(0)} \leqslant l_i^{(2)}, \end{aligned} \qquad k = \overline{1,N}, \quad i = \overline{1,r}. \tag{3.13}$$

Куда более сложная ситуация с вершиной параболы $-p_{ik}^{(1)}/2p_{ik}^{(2)}$. Было бы неправильно сказать, что значение экстремума должно удовлетворять двусторонним ограничениям, поскольку допустимо нарушать границы, если аргумент экстремума находится справа или слева от сегмента $[t_{k-1},\ t_k)$.

Для простоты изложения переобозначим некоторые переменные:

$$p := p_{ik}^{(2)}, \quad \tau_1 := t_{k-1}, \quad \tau_2 := t_k, \quad l_1 := l_i^{(1)}, \quad l_2 := l_i^{(2)}.$$

И введем обозначения для значений квадратичного полинома в узлах:

$$y_1 := p_{ik}^{(2)}t_{k-1}^2 + p_{ik}^{(1)}t_{k-1} + p_{ik}^{(0)}, \quad y_2 := p_{ik}^{(2)}t_k^2 + p_{ik}^{(1)}t_k + p_{ik}^{(0)}.$$

Теперь можно выразить переменные $p_{ik}^{(1)}$ и $p_{ik}^{(0)}$ через p, y_1 и y_2:

$$p_{ik}^{(1)} = \frac{y_2 - y_1}{\tau_2 - \tau_1} - p(\tau_1 + \tau_2), \quad p_{ik}^{(0)} = p\tau_1\tau_2 - \frac{y_2 - y_1}{\tau_2 - \tau_1}\tau_1 + y_1.$$

После этого можем найти условия, при которых вершина полинома равняется верхней границе l_2:

$$\left(\frac{y_2 - y_1}{\tau_2 - \tau_1} - p(\tau_1 + \tau_2)\right)^2 - 4p\left(p\tau_1\tau_2 - \frac{y_2 - y_1}{\tau_2 - \tau_1}\tau_1 + y_1 - l_2\right) = 0.$$

После ряда преобразований получим квадратное уравнение относительно переменной p:

$$(\tau_2 - \tau_1)^2 p^2 + (-2(y_1 + y_2) + 4l_2)\, p + \left(\frac{y_2 - y_1}{\tau_2 - \tau_1}\right)^2 = 0,$$

Оно имеет два решения:

$$p = -\left(\frac{\sqrt{l_2 - y_1} \pm \sqrt{l_2 - y_2}}{\tau_2 - \tau_1}\right)^2.$$

Таким образом, была решена следующая задача: найти параметры квадратичного полинома в зависимости от его значений в двух заданных точках (y_1 в точке τ_1 и y_2 в точке τ_2) и условии, что значение экстремума зафиксировано и равно некоторому заранее заданному числу (l_2). Анализируя полученные результаты, можно понять, что решение с наибольшим абсолютным значением соответствует параболе, аргумент вершины которой находится между двумя заданными точками (внутри отрезка $[\tau_1,\ \tau_2]$). Оставшееся решение соответствует параболе, аргумент вершины которой расположен слева или справа от заданных точек.

Как уже было сказано ранее, ограничение на экстремум квадратичного полинома имеет место быть только, если он расположен внутри рассматриваемого отрезка. Поэтому в данном случае нас интересует только решение со знаком «плюс» в числителе:

$$p = -\left(\frac{\sqrt{l_2 - y_1} + \sqrt{l_2 - y_2}}{\tau_2 - \tau_1}\right)^2.$$

Проделав то же самое для нижней границы, найдем двусторонние ограничения на p:

$$-\left(\frac{\sqrt{l_2 - y_1} + \sqrt{l_2 - y_2}}{\tau_2 - \tau_1}\right)^2 \leqslant p \leqslant \left(\frac{\sqrt{y_1 - l_1} + \sqrt{y_2 - l_1}}{\tau_2 - \tau_1}\right)^2. \qquad (3.14)$$

Теперь вернемся к изначальным обозначениям, введем новые переменные $s_{ik}^{(1)}$, $s_{ik}^{(2)}$, $s_{ik}^{(3)}$, $s_{ik}^{(4)}$ и добавим ограничения на них:

$$\begin{aligned}
&s_{ik}^{(1)} = p_{ik}^{(2)} t_{k-1}^2 + p_{ik}^{(1)} t_{k-1} + p_{ik}^{(0)} - l_i^{(1)},\\
&s_{ik}^{(2)} = p_{ik}^{(2)} t_k^2 + p_{ik}^{(1)} t_k + p_{ik}^{(0)} - l_i^{(1)},\\
&s_{ik}^{(3)} = l_i^{(2)} - p_{ik}^{(2)} t_{k-1}^2 - p_{ik}^{(1)} t_{k-1} - p_{ik}^{(0)}, \qquad k = \overline{1,N}, \quad i = \overline{1,r}. \qquad (3.15)\\
&s_{ik}^{(4)} = l_i^{(2)} - p_{ik}^{(2)} t_k^2 - p_{ik}^{(1)} t_k - p_{ik}^{(0)},\\
&s_{ik}^{(1)} \geqslant 0, \quad s_{ik}^{(2)} \geqslant 0, \quad s_{ik}^{(3)} \geqslant 0, \quad s_{ik}^{(4)} \geqslant 0,
\end{aligned}$$

После этого перепишем неравенства (3.14):

$$-\left(\frac{\sqrt{s^{(3)}}+\sqrt{s^{(4)}}}{T/N}\right)^2 \leqslant p_{ik}^{(2)} \leqslant \left(\frac{\sqrt{s^{(1)}}+\sqrt{s^{(2)}}}{T/N}\right)^2, \quad k=\overline{1,N}, \quad i=\overline{1,r}.$$

Далее, раскрыв скобки, получим следующие нелинейные ограничения:

$$\begin{aligned} &T^2 p_{ik}^{(2)} \leqslant N^2\left(s_{ik}^{(1)}+s_{ik}^{(2)}+2\sqrt{s_{ik}^{(1)}s_{ik}^{(2)}}\right), \\ &T^2 p_{ik}^{(2)} \geqslant -N^2\left(s_{ik}^{(3)}+s_{ik}^{(4)}+2\sqrt{s_{ik}^{(3)}s_{ik}^{(4)}}\right), \end{aligned} \qquad k=\overline{1,N}, \quad i=\overline{1,r}. \tag{3.16}$$

Формально, можно было бы остановиться на ограничениях (3.16), однако, такие нелинейные неравенства, содержащие квадратные корни, крайне неудобные для поиска оптимального решения, поскольку вносят невыпуклость в рассчитываемую модель и сложны для восприятия солверами. Поэтому попытается свести их к выпуклым ограничениям.

Введем неотрицательные переменные $r_{ik}^{(1)}$ и $r_{ik}^{(2)}$ и выразим нелинейную часть (3.16) через них:

$$\begin{aligned} &\left(r_{ik}^{(1)}\right)^2 \leqslant 4 s_{ik}^{(1)} s_{ik}^{(2)}, \\ &\left(r_{ik}^{(2)}\right)^2 \leqslant 4 s_{ik}^{(3)} s_{ik}^{(4)}, \qquad k=\overline{1,N}, \quad i=\overline{1,r}. \\ &r_{ik}^{(1)} \geqslant 0, \quad r_{ik}^{(2)} \geqslant 0, \end{aligned} \tag{3.17}$$

Отметим, что ограничения (3.17) принадлежат классу конических уравнений второго порядка, можно доказать, что такие ограничения имеют выпуклое множество решений. Данный факт серьезно упрощает оптимизационную задачу. Этот результат является главным достижением текущего параграфа.

Наконец выпишем окончательное представление ограничений (3.16):

$$\begin{aligned} &T^2 p_{ik}^{(2)} \leqslant N^2\left(s_{ik}^{(1)}+s_{ik}^{(2)}+r_{ik}^{(1)}\right), \\ &T^2 p_{ik}^{(2)} \geqslant -N^2\left(s_{ik}^{(3)}+s_{ik}^{(4)}+r_{ik}^{(2)}\right), \end{aligned} \qquad k=\overline{1,N}, \quad i=\overline{1,r}. \tag{3.18}$$

В итоге показано, как прямые двусторонние ограничения (1.3) были преобразованы в линейные ограничения (3.13), (3.15), (3.18) и квадратичные неравенства (3.17).

Сведение целевых функций

Для сведения первой целевой функции воспользуемся представлением финального состояния объекта (3.11). Тогда можем записать критерий (1.5) как линейную функцию от переменных $p^{(0)},\ p^{(1)},\ p^{(2)}$:

$$J_1 = c^T d^0 + \sum_{i=1}^{r}\sum_{k=1}^{N}\left(c^T b^{i,k,0} p_{ik}^{(0)} + c^T b^{i,k,1} p_{ik}^{(1)} + c^T b^{i,k,2} p_{ik}^{(2)}\right).$$

Поскольку первое слагаемое является константой, можем удалить его из целевой функции, при этом имея в виду, что от этого может поменяться значение функционала:

$$J_1 = \sum_{i=1}^{r}\sum_{k=1}^{N}\left(c^{i,k,0} p_{ik}^{(0)} + c^{i,k,1} p_{ik}^{(1)} + c^{i,k,2} p_{ik}^{(2)}\right) \longrightarrow \min, \qquad (3.19)$$

где

$$\begin{aligned} c^{i,k,0} &= c^T b^{i,k,0}, \\ c^{i,k,1} &= c^T b^{i,k,1}, \quad k = \overline{1,N}, \quad i = \overline{1,r}. \\ c^{i,k,2} &= c^T b^{i,k,2}, \end{aligned}$$

Таким образом, (3.19) представляет собой линейный функционал относительно переменных $p^{(0)},\ p^{(1)},\ p^{(2)}$.

Второй критерий (1.6) не будет рассмотрен для данного типа управления, поскольку нахождение модуля от кусочно-линейной функции требует дополнительных вычислений, выходящих за рамки данной работы.

Перейдем к третьему критерию. Заменим управление $u(t)$ квадратичными сплайнами (3.8) в целевой функции (1.7):

$$J_2 = \sum_{k=1}^{N}\sum_{i_1=1}^{r}\sum_{i_2=1}^{r}\int_{t_{k-1}}^{t_k} q_{i_1 i_2}\left(p_{i_1 k}^{(2)} t^2 + p_{i_1 k}^{(1)} t + p_{i_1 k}^{(0)}\right)\left(p_{i_2 k}^{(2)} t^2 + p_{i_2 k}^{(1)} t + p_{i_2 k}^{(0)}\right) dt.$$

После вычисления интегралов и введения новых обозначений, получим квадратичную форму относительно переменных $p_{ik}^{(0)}$, $p_{ik}^{(1)}$ и $p_{ik}^{(2)}$:

$$J_2 = \sum_{k=1}^{N} \sum_{i_1=1}^{r} \sum_{i_2=1}^{r} \sum_{\alpha_1=0}^{2} \sum_{\alpha_2=0}^{2} \left(h_{i_1 i_2 k}^{\alpha_1,\alpha_2} p_{i_1 k}^{(\alpha_1)} p_{i_2 k}^{(\alpha_2)} \right), \tag{3.20}$$

где

$$h_{i_1 i_2 k}^{\alpha_1,\alpha_2} := q_{i_1 i_2} \frac{1}{\beta} \frac{T^\beta}{N^\beta} \left(k^\beta - (k-1)^\beta \right), \quad \beta = \alpha_1 + \alpha_2 + 1,$$
$$k = \overline{1,N}, \quad i_1 = \overline{1,r}, \quad i_2 = \overline{1,r}, \quad \alpha_1 = \overline{0,2}, \quad \alpha_2 = \overline{0,2}.$$

Поскольку Q — неотрицательно определенная матрица, в три шага может быть доказано, что квадратичная форма в формуле (3.20) также является неотрицательно определенной:

1. Итоговая матрица является блочно-диагональной матрицей, состоящей из N блоков. Поэтому необходимо показать, что каждый блок является неотрицательно определенной матрицей.
2. Если для каждого k зафиксировать t, то окажется, что $u_i(t)$ — линейная функция относительно $p_{ik}^{(0)}$, $p_{ik}^{(1)}$ и $p_{ik}^{(2)}$. Далее можно показать, что суперпозиция $G(F(\xi))$ является неотрицательно определенной квадратичной формой, если $G(\eta)$ — неотрицательно определенная квадратичная форма, а $F(\xi)$ — линейная функция. Отсюда мы получаем, что под интегралом стоит неотрицательно определенная квадратичная форма относительно переменных $p_{ik}^{(0)}$, $p_{ik}^{(1)}$ и $p_{ik}^{(2)}$ для каждого момента t.
3. Далее используем тот факт, что результатом интегрирования неотрицательно определенной квадратичной формы по параметру t является неотрицательно определенная квадратичная форма.

Замечание 3.1. Так как кусочно-постоянные и кусочно-линейные функции являются частыми случаями кусочно-квадратичных функций, из этого следует, что в (1.14) и (3.7) также были получены неотрицательно определенные квадратичные формы.

Итоги

Было показано, как при выборе управления в виде кусочно-квадратичной функции можно свести линейную задачу оптимального управления (1.1)–(1.3), (3.8), (1.5) к задаче программирования с квадратичными ограничениями (3.9), (3.10), (3.12), (3.13), (3.15), (3.17)–(3.19):

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(c^{i,k,0}p_{ik}^{(0)}+c^{i,k,1}p_{ik}^{(1)}+c^{i,k,2}p_{ik}^{(2)}\right)\longrightarrow\min$$

$$p_{ik}^{(2)}t_k^2+p_{ik}^{(1)}t_k+p_{ik}^{(0)}=p_{ik+1}^{(2)}t_k^2+p_{ik+1}^{(1)}t_k+p_{ik+1}^{(0)},\quad k=\overline{1,N-1},\quad i=\overline{1,r}.$$

$$2p_{ik}^{(2)}t_k+p_{ik}^{(1)}=2p_{ik+1}^{(2)}t_k+p_{ik+1}^{(1)},\quad k=\overline{1,N-1},\quad i=\overline{1,r}.$$

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(h_l^{i,k,0}p_{ik}^{(0)}+h_l^{i,k,1}p_{ik}^{(1)}+h_l^{i,k,2}p_{ik}^{(2)}\right)=g_l,\quad l=\overline{1,m}.$$

$$\left.\begin{array}{l} l_i^{(1)}\leqslant p_{ik}^{(2)}t_{k-1}^2+p_{ik}^{(1)}t_{k-1}+p_{ik}^{(0)}\leqslant l_i^{(2)},\\ l_i^{(1)}\leqslant p_{ik}^{(2)}t_k^2+p_{ik}^{(1)}t_k+p_{ik}^{(0)}\leqslant l_i^{(2)},\end{array}\right.\quad k=\overline{1,N},\quad i=\overline{1,r}.$$

$$\left.\begin{array}{l} s_{ik}^{(1)}=p_{ik}^{(2)}t_{k-1}^2+p_{ik}^{(1)}t_{k-1}+p_{ik}^{(0)}-l_i^{(1)},\\ s_{ik}^{(2)}=p_{ik}^{(2)}t_k^2+p_{ik}^{(1)}t_k+p_{ik}^{(0)}-l_i^{(1)},\\ s_{ik}^{(3)}=l_i^{(2)}-p_{ik}^{(2)}t_{k-1}^2-p_{ik}^{(1)}t_{k-1}-p_{ik}^{(0)},\\ s_{ik}^{(4)}=l_i^{(2)}-p_{ik}^{(2)}t_k^2-p_{ik}^{(1)}t_k-p_{ik}^{(0)},\end{array}\right.\quad k=\overline{1,N},\quad i=\overline{1,r}.$$

$$s_{ik}^{(1)}\geqslant 0,\quad s_{ik}^{(2)}\geqslant 0,\quad s_{ik}^{(3)}\geqslant 0,\quad s_{ik}^{(4)}\geqslant 0,$$

$$\left.\begin{array}{l}\left(r_{ik}^{(1)}\right)^2\leqslant 4s_{ik}^{(1)}s_{ik}^{(2)},\\ \left(r_{ik}^{(2)}\right)^2\leqslant 4s_{ik}^{(3)}s_{ik}^{(4)},\\ r_{ik}^{(1)}\geqslant 0,\quad r_{ik}^{(2)}\geqslant 0,\end{array}\right.\quad k=\overline{1,N},\quad i=\overline{1,r}.$$

$$\left.\begin{array}{l} T^2p_{ik}^{(2)}\leqslant N^2\left(s_{ik}^{(1)}+s_{ik}^{(2)}+r_{ik}^{(1)}\right),\\ T^2p_{ik}^{(2)}\geqslant -N^2\left(s_{ik}^{(3)}+s_{ik}^{(4)}+r_{ik}^{(2)}\right),\end{array}\right.\quad k=\overline{1,N},\quad i=\overline{1,r}.$$

Данная задача обладает следующими характеристиками:

Переменные

1. $3rN$ основных вещественных переменных: $p_{ik}^{(0)}$, $p_{ik}^{(1)}$, $p_{ik}^{(2)}$, $k = \overline{1,N}$, $i = \overline{1,r}$.
2. $6rN$ вещественных неотрицательных переменных: $s_{ik}^{(1)}$, $s_{ik}^{(2)}$, $s_{ik}^{(3)}$, $s_{ik}^{(4)}$, $r_{ik}^{(1)}$, $r_{ik}^{(2)}$, $k = \overline{1,N}$, $i = \overline{1,r}$.

Ограничения

1. $6rN-2r+m$ линейных уравнений: (3.9), (3.10), (3.12), (3.15). Таким образом, число переменных может быть сокращено.
2. $6rN$ линейных неравенств: (3.13), (3.18).
3. $2rN$ квадратичных неравенств: (3.17). Несмотря на то, что квадратичные ограничения не представлены неотрицательно определенными квадратичными формами, они являются коническими уравнениями второго порядка. А тогда может быть доказано, что в случае неотрицательности переменных $s_{ik}^{(1)}$, $s_{ik}^{(2)}$, $s_{ik}^{(3)}$, $s_{ik}^{(4)}$, множество решений для этих ограничений является выпуклым множеством.

Соответственно, для данной оптимизационной задачи имеет место выпуклое множество допустимых решений.

Целевая функция

Целевая функция (3.19) является линейной функцией от переменных $p_{ik}^{(0)}$, $p_{ik}^{(1)}$, $p_{ik}^{(2)}$, $k = \overline{1,N}$, $i = \overline{1,r}$.

Задачу оптимального управления (1.1)–(1.3), (3.8), (1.7) с линейной системой и квадратичном критерием можно свести к задаче программирования с квадратичными ограничениями (3.9), (3.10), (3.12), (3.13), (3.15), (3.17), (3.18), (3.20).

Данная оптимизационная задача имеет те же характеристики, что и предыдущий вариант за тем исключением, что целевой функцией является выпуклая квадратичная функция (3.20) от переменных $p_{ik}^{(0)}$, $p_{ik}^{(1)}$, $p_{ik}^{(2)}$, $k = \overline{1,N}$, $i = \overline{1,r}$.

Пример. Демпфирование колебаний «спящего волчка» Лагранжа

Вернемся к задаче демпфирования спящего волчка, описанную в параграфе 1.5, построив управление в классе кусочно-квадратичных функций.

Процедура сведения была реализована на языке Python 3, задача квадратичного программирования была решена с помощью солвера COPT (см. приложение А).

Общее время работы программы составило 1,11 секунд. Значение целевой функции равняется 91,56. Результаты приведены на рис. 3.2.

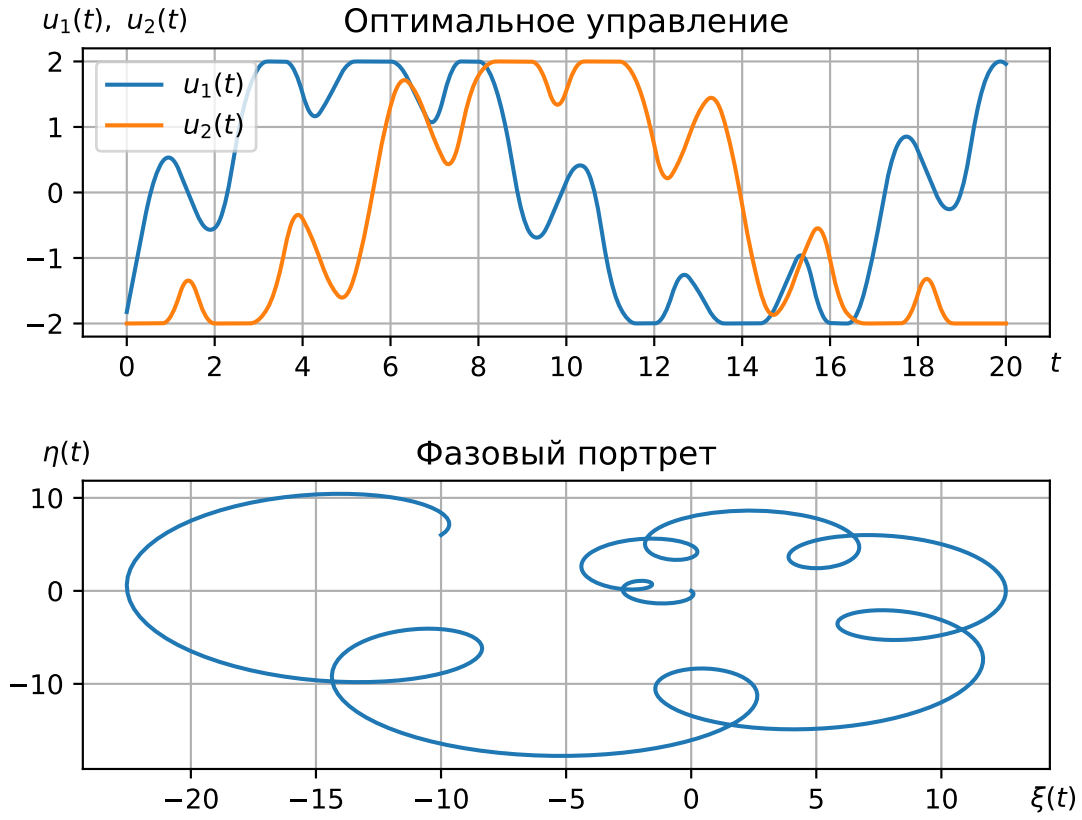

Рисунок 3.2 — Демпфирование колебаний «спящего волчка» Лагранжа при выборе управления в классе кусочно-квадратичных функций

3.1.3 Численная реализация

Интегрирование ОДУ и вычисление определенных интегралов

В ходе этапа сведения возник вопрос вычисления интегралов типа

$$\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt.$$

Для начала определимся со способом вычисления матрицы $Y(T)Y^{-1}(t)$. Обозначим символом $Y_{t_1}(t)$ матрицант однородной системы с матрицей $A(t)$, нормированный в точке t_1:

$$\dot{Y}_{t_1}(t) = A(t)Y_{t_1}(t), \quad Y_{t_1}(t_1) = E.$$

Теперь преобразуем произведение $Y_0(T)Y_0^{-1}(t)$, воспользовавшись свойствами матрицанта [74]:

$$Y_0(T)Y_0^{-1}(t) = Y_0(T)Y_t(0) = Y_t(T) = Y_T^{-1}(t).$$

Рассмотрим сопряженную систему

$$\dot{z} = -A^T(t)z. \tag{3.21}$$

Пусть $Z_{t_1}(t)$ — матрицант этой системы, нормированный в t_1. Его основное свойство:

$$Z_{t_1}^T(t) = Y_{t_1}^{-1}(t)$$

Тогда подставляя $t_1 = T$, получим соотношение $Z_T^T(t) = Y_T^{-1}(t)$. Следовательно, матрица $Y(T)Y^{-1}(t)$ является транспонированным матрицантом сопряженной системы (3.21), нормированным в точке T. Для его вычисления, необходимо n раз решить задачу Коши с системой (3.21) и различными начальными условиями $z(T) = \vec{e}_j,\ j = \overline{1,n}$. Предполагается, что решения

будут найдены численно. Например, при подготовке данной работы для этих целей использовалась функция *odeint* из пакета *scipy.integrate*, подключаемого к Python.

После вычисления функций $Z_T(t)$ останется вычислить определенные интегралы $\int\limits_{t_{k-1}}^{t_k} Z_T^T(t)b_i(t)dt$. Их можно рассчитать в квадратурах используя функцию *quad* из того же пакета *scipy.integrate*.

Интегралы типа $\int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i t dt$ и $\int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i t^2 dt$ можно вычислить по аналогии либо с использованием найденных значений $\int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i dt$ и формул интегрирования по частям.

Кроме вычисления вышеупомянутых интегралов, на этапе сведения возникает проблема определения значения

$$Y(T)x_* + \int\limits_0^T Y(T)Y^{-1}(t)d(t)dt.$$

Для этого с помощью библиотеки *odeint* требуется вычислить решение задачи Коши для системы (1.1) с нулевым управлением

$$\dot{\tilde{x}} = A(t)\tilde{x} + d(t), \quad x(0) = x_*$$

и найти значение $\tilde{x}(T)$.

Схема решения

Решение поставленной задачи было реализовано на языке Python 3. Код программ доступен в приложении А, сейчас же приведем верхнеуровневую схему решения.

1. Задание условий задачи оптимального управления (1.1)–(1.3) с ЦФ (1.5) или (1.7).
2. Выбор класса управления: кусочно-постоянный (1.4), кусочно-линейный (3.1) или кусочно-квадратичный (3.8).

3. Вычисление параметров задачи математического программирования, в том числе с использованием
   - алгоритмов решения систем линейных обыкновенных дифференциальных уравнений с помощью функции *odeint*.
   - формул вычисления определенных интегралов с помощью функции *quad*.
4. Решение задачи математического программирования при помощи пакета для оптимизации *coptpy*.
5. Формирование функций управления, исходя из решения задачи математического программирования.
6. Поиск решения исходной системы ОДУ, замкнутой оптимальным управлением, при помощи функции *odeint*.
7. Построение графиков решения через функции пакета *matplotlib*.

3.2 Оптимальное управление с ограничениями на фазовые переменные и компоненты управления

В этом параграфе рассмотрим задачу оптимального управления при наличии различных выпуклых и невыпуклых ограничений на фазовый вектор и на вектор управления.

Базовой моделью будем считать задачу управления (1.1)–(1.5) с линейной системой обыкновенных дифференциальных уравнений, терминальными условиями, прямыми ограничениями на управления, кусочно-постоянными управлениями и линейным целевым функционалом.

Данная задача сводится к задаче линейного программирования (1.9)–(1.11). Все выведенные в данной главе ограничения будем считать дополнительными к основной модели.

3.2.1 Ограничения на компоненты управления

Выпуклые линейные ограничениями

Дополним линейную задачу оптимального управления набором ограничений относительно вектора управлений u:

$$Du \leqslant d.$$

Здесь D — матрица размера $q \times r$, состоящая из столбцов d_i, а d — q-мерный вектор.

Тогда можем свести эти ограничения к линейным неравенствам относительно переменных u_{ik}:

$$\sum_{i=1}^{r} d_i u_{ik} \leqslant d, \quad k = \overline{1, N}. \tag{3.22}$$

Таким образом получили qN линейных неравенств, которые необходимо добавить к задаче линейного (квадратичного) программирования.

Невыпуклые линейные ограничениями

Пусть теперь на управления наложены ограничения типа «ИЛИ»:

$$\begin{bmatrix} D^{(1)}u \leqslant d^{(1)}, \\ D^{(2)}u \leqslant d^{(2)}, \\ \dots \\ D^{(s)}u \leqslant d^{(s)}. \end{bmatrix} \tag{3.23}$$

Учитывая, что управление имеет двусторонние границы (1.3), для любой строки $\boldsymbol{\gamma}$ можем найти такой параметр M, что при любых допустимых

управлениях будет выполняться неравенство

$$\gamma u \leqslant M.$$

Для этого достаточно определить M следующим образом:

$$M = \sum_{\substack{i=1 \\ \gamma_i<0}}^{r} \gamma_i l_{*i} + \sum_{\substack{i=1 \\ \gamma_i>0}}^{r} \gamma_i l_i^*.$$

Подобным образом можем определить вектор $\tilde{d}^{(\alpha)}$ для матрицы $D^{(\alpha)}$. Тогда для любого допустимого управления будет выполняться

$$D^{(\alpha)} u \leqslant \tilde{d}^{(\alpha)}, \quad \alpha = \overline{1, s}.$$

Тогда, учитывая, что блоки ограничений из (3.22) не обязательно должны всегда выполняться, введем бинарные функции $\delta_\alpha(t)$, являющиеся индикаторами того, какой блок α должен выполняться в момент t:

$$\delta_\alpha(t) = \begin{cases} 1, \text{ если неравенства } D^{(\alpha)} u(t) \leqslant d^{(\alpha)} \text{ должны выполняться,} \\ 0, \text{ если неравенства } D^{(\alpha)} u(t) \leqslant d^{(\alpha)} \text{ могут нарушаться,} \end{cases}$$

где $\alpha = \overline{1,s}$. При этом всегда должен быть один активный блок:

$$\sum_{\alpha=1}^{s} \delta_\alpha(t) = 1. \tag{3.24}$$

Тогда можно переписать (3.23) как систему неравенств:

$$\begin{cases} D^{(1)} u \leqslant d^{(1)} \delta_1(t) + \tilde{d}^{(1)}(1 - \delta_1(t)), \\ D^{(2)} u \leqslant d^{(2)} \delta_2(t) + \tilde{d}^{(2)}(1 - \delta_2(t)), \\ \dots \\ D^{(s)} u \leqslant d^{(s)} \delta_s(t) + \tilde{d}^{(s)}(1 - \delta_s(t)). \end{cases} \tag{3.25}$$

Теперь можем перейти от динамических ограничений к статическим. Поскольку управление выбирается из класса кусочно-постоянных функций, не умаляя общности, можно рассматривать и $\delta_\alpha(t)$ как кусочно-постоянную функцию:

$$\delta_\alpha(t) = \delta_{\alpha k}, \quad k = \overline{1,N}, \quad \alpha = \overline{1,s},$$

где $\delta_{\alpha k}$ — бинарные переменные. Тогда равенство (3.24) перейдет в N линейных уравнений:

$$\sum_{\alpha=1}^{s} \delta_{\alpha k} = 1, \quad k = \overline{1,N}. \tag{3.26}$$

А систему (3.25) можно переписать в виде:

$$\sum_{i=1}^{r} d^{i,\alpha} u_{ik} \leqslant d^{(\alpha)} \delta_{\alpha k} + \tilde{d}^{(\alpha)}(1 - \delta_{\alpha k}), \quad k = \overline{1,N}, \quad \alpha = \overline{1,s}. \tag{3.27}$$

Здесь $d^{i,\alpha}$ — i-ый столбец $D^{(\alpha)}$.

Пример

Пусть для некоторой задачи управления задан двумерный вектор управлений со следующими прямыми ограничениями:

$$0 \leqslant u_1(t) \leqslant 10, \quad 0 \leqslant u_2(t) \leqslant 10.$$

При этом дополнительно заданы два линейных ограничения:

$$\begin{cases} u_1(t) + u_2(t) \leqslant 17, \\ u_1(t) - 2u_2(t) \leqslant 6. \end{cases}$$

Кроме того, управления не должны принадлежать прямоугольнику, описываемому следующими неравенствами:

$$\begin{cases} 2 \leqslant u_1(t) \leqslant 5, \\ 2 \leqslant u_2(t) \leqslant 7. \end{cases}$$

На рис. 3.3 множество допустимых управлений изображено синим цветом.

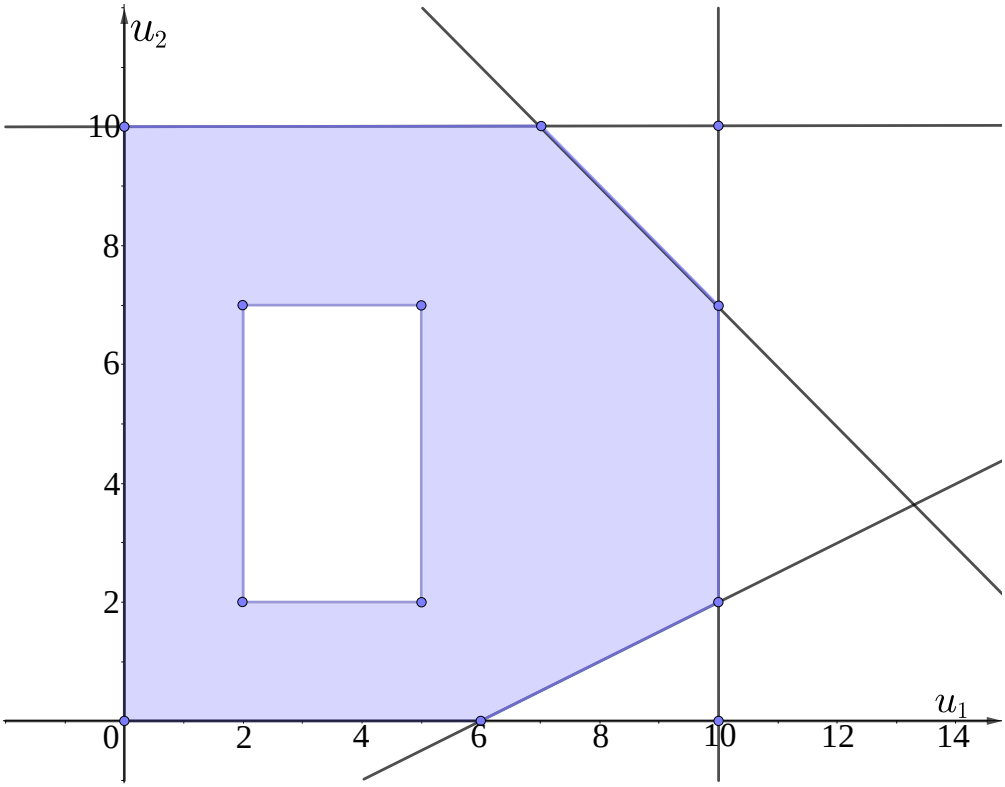

Рисунок 3.3 — Невыпуклое множество допустимых управлений

Переведем требования на не принадлежность управлений множеству к четырем ограничениям типа «ИЛИ»:

$$\left[\begin{array}{l} u_1(t) \leqslant 2, \\ u_1(t) \geqslant 5, \\ u_2(t) \leqslant 2, \\ u_2(t) \geqslant 7. \end{array}\right. \tag{3.28}$$

Для получения совокупности уравнений введем бинарные функции $\delta_1(t)$, $\delta_2(t)$, $\delta_3(t)$, $\delta_4(t)$, сумма которых равна единице:

$$\delta_1(t) + \delta_2(t) + \delta_3(t) + \delta_4(t) = 1.$$

Теперь можем переписать (3.28) в виде

$$\left\{\begin{array}{l} u_1(t) \leqslant 2\delta_1(t) + 10(1 - \delta_1(t)), \\ -u_1(t) \leqslant -5\delta_2(t), \\ u_2(t) \leqslant 2\delta_3(t) + 10(1 - \delta_3(t)), \\ -u_2(t) \leqslant -7\delta_4(t). \end{array}\right.$$

В итоге, множество допустимых управлений будет описываться следующей системой:

$$\begin{cases} 0 \leqslant u_1(t) \leqslant 10, \\ 0 \leqslant u_2(t) \leqslant 10, \\ u_1(t) + u_2(t) \leqslant 17, \\ u_1(t) - 2u_2(t) \leqslant 6, \\ u_1(t) \leqslant 2\delta_1(t) + 10(1 - \delta_1(t)), \\ -u_1(t) \leqslant -5\delta_2(t), \\ u_2(t) \leqslant 2\delta_3(t) + 10(1 - \delta_3(t)), \\ -u_2(t) \leqslant -7\delta_4(t), \\ \delta_1(t) + \delta_2(t) + \delta_3(t) + \delta_4(t) = 1, \\ \delta_1(t),\ \delta_2(t),\ \delta_3(t),\ \delta_4(t) \in \{0, 1\}. \end{cases}$$

После этого, необходимо свести такую систему к системе линейных алгебраических уравнений по алгоритму, описанному в предыдущих пунктах. Поскольку такой переход достаточно тривиальный, не будем приводить его в данном примере.

3.2.2 Ограничения на фазовые переменные

По аналогии с предыдущим пунктом рассмотрим выпуклые и невыпуклые ограничения на фазовые переменные в узлах — точках скачков кусочно-постоянных функций управления.

Также отдельно уделим внимание ограничениям на всем отрезке времени $[0,\ T]$. Выполнение таких ограничений является более трудоемкой операцией.

Выпуклые линейные ограничениями

Пусть в каждой точке t_k имеют место линейные ограничения на фазовые переменные

$$\hat{H}_k x(t_k) \leqslant \hat{g}_k^0, \quad k = \overline{1,N}. \tag{3.29}$$

Такие ограничения сильно напоминают терминальные условия (1.2). Будем преобразовывать их таким же образом. Выпишем формулу Коши в точке t_k:

$$x(t_k) = Y(t_k)x_* + \int_0^{t_k} Y(t_k)Y^{-1}(t)\left(B(t)u(t) + d(t)\right)dt.$$

Подставим кусочно-постоянную функцию управления вместо $u(t)$:

$$\begin{gathered} x(t_k) = Y(t_k)x_* + \int_0^{t_k} Y(t_k)Y^{-1}(t)d(t)dt + \\ + \sum_{i=1}^{r}\sum_{\varkappa=1}^{k}\int_{t_{\varkappa-1}}^{t_\varkappa} Y(t_k)Y^{-1}(t)b_i(t)dt u_{i\varkappa}. \end{gathered} \tag{3.30}$$

Внесем полученную формулу в неравенство (3.29):

$$\begin{gathered} \sum_{i=1}^{r}\sum_{\varkappa=1}^{k}\hat{H}_k\int_{t_{\varkappa-1}}^{t_\varkappa} Y(t_k)Y^{-1}(t)b_i(t)dt u_{i\varkappa} \leqslant \\ \leqslant \hat{g}_k^0 - \hat{H}_k Y(t_k)x_* - \hat{H}_k\int_0^{t_k} Y(t_k)Y^{-1}(t)d(t)dt. \end{gathered}$$

После замены переменных получим линейные неравенства относительно переменных $u_{i\varkappa}$:

$$\sum_{i=1}^{r}\sum_{\varkappa=1}^{k}\hat{h}_k^{i\varkappa}u_{i\varkappa} \leqslant \hat{g}_k, \quad k = \overline{1,N}, \tag{3.31}$$

при обозначениях

$$\hat{h}_k^{i\varkappa} = \hat{H}_k \int\limits_{t_{\varkappa-1}}^{t_\varkappa} Y(t_k)Y^{-1}(t)b_i(t)dt,$$

$$\hat{g}_k = \hat{g}_k^0 - \hat{H}_k Y(t_k)x_* - \hat{H}_k \int\limits_0^{t_k} Y(t_k)Y^{-1}(t)d(t)dt.$$

В итоге условия (3.29) были сведены к линейным ограничениям (3.31).

Замечание 3.2. Таким же способом можно учесть ограничения на фазовые переменные в произвольных точках отрезка $[0,\, T]$.

Невыпуклые линейные ограничениями

Теперь рассмотрим требования на фазовые переменные, в которых присутствуют ограничения типа «ИЛИ»:

$$\begin{bmatrix} A_k^{(1)}x(t_k) \leqslant b_k^{(1)}, \\ A_k^{(2)}x(t_k) \leqslant b_k^{(2)}, \\ \dots \\ A_k^{(v)}x(t_k) \leqslant b_k^{(v)}, \end{bmatrix} \quad k = \overline{1, N}. \tag{3.32}$$

Ограничения таких видов появляются в невыпуклых пространствах. Например, когда объект управления должен обогнуть некоторый сторонний объект.

Основным отличием данного случая от невыпуклых ограничений на управления является отсутствие прямых ограничений для фазовых переменных. Однако, можно определить границы на переменные одним из двух способов:

– вычислить аналитически, исходя из некоторых логических предположений,

– найти минимум и максимум для каждой компоненты $x(t_k)$, выписав ее представление через $u_{i\varkappa}$, и применив знания о границах для управлений.

Будем считать, что удалось найти такие векторы $\tilde{b}_k^{(\beta)}$, что для любого положения объекта $x(t_k)$, которое можно достигнуть при выборе допустимого управления, выполняется

$$A_k^{(\beta)} x(t_k) \leqslant \tilde{b}_k^{(\beta)}, \quad k = \overline{1, N}, \quad \beta = \overline{1, v}.$$

Теперь введем бинарные переменные $\sigma_{\beta k}$, сигнализирующие, что в момент t_k активно ограничение $A_k^{(\beta)} x(t_k) \leqslant b_k^{(\beta)}$. Тогда перейдем от требований (3.32) к системе ограничений:

$$\begin{cases} A_k^{(1)} x(t_k) \leqslant b_k^{(1)} \sigma_{1k} + \tilde{b}_k^{(1)} (1 - \sigma_{1k}), \\ A_k^{(2)} x(t_k) \leqslant b_k^{(2)} \sigma_{2k} + \tilde{b}_k^{(2)} (1 - \sigma_{2k}), \\ \dots \\ A_k^{(v)} x(t_k) \leqslant b_k^{(v)} \sigma_{vk} + \tilde{b}_k^{(v)} (1 - \sigma_{vk}), \end{cases} \quad k = \overline{1, N}. \tag{3.33}$$

Подставив правую часть формулы (3.30) вместо $x(t_k)$ в (3.33), получим линейные неравенства следующего типа:

$$\begin{gathered} \sum_{i=1}^{r} \sum_{\varkappa=1}^{k} a_k^{i\varkappa\beta} u_{i\varkappa} \leqslant q_k^{(\beta)} + b_k^{(\beta)} \sigma_{\beta k} + \tilde{b}_k^{(\beta)} (1 - \sigma_{\beta k}), \\ k = \overline{1, N}, \quad \beta = \overline{1, v}, \end{gathered} \tag{3.34}$$

где

$$a_k^{i\varkappa\beta} = A_k^{(\beta)} \int_{t_{\varkappa-1}}^{t_\varkappa} Y(t_k) Y^{-1}(t) b_i(t) dt,$$

$$q_k^{(\beta)} = -A_k^{(\beta)} Y(t_k) x_* - A_k^{(\beta)} \int_0^{t_k} Y(t_k) Y^{-1}(t) d(t) dt.$$

Последнее, что нужно сделать — указать, что в каждый момент t_k должен быть активен один блок неравенств (3.33):

$$\sum_{\beta=1}^{v} \sigma_{\beta k} = 1, \quad k = \overline{1,N}. \tag{3.35}$$

Таким образом было показано, как требования (3.32) можно свести к системе линейных ограничений с бинарными переменными (3.34), (3.35).

Непрерывные ограничения

Большой практический интерес представляют ограничения на фазовые переменные на всем отрезке времени:

$$Gx(t) \leqslant q, \quad t \in [0,\ T].$$

Но, к сожалению, такая задача крайне сложна и в общем случае не может быть сведена к удобной задаче математического программирования.

При этом можно предложить несколько подходов, позволяющих повысить шансы на выполнение этих неравенств для любого t:

– Увеличить количество моментов времени, на которых накладываются ограничения

$$Gx(t_s) \leqslant q, \quad t_s = \frac{sT}{N_1}, \quad s = \overline{1, N_1}. \quad N_1 \to \infty.$$

– Участить ограничения на тех сегментах, в которых предполагается сильное изменение состояния объекта.

– Оценить максимальное возможное отклонение состояния объекта и заложить эти отклонения в ограничениях.

$$\begin{array}{l} x_j(t_s) - m_j^1 \leqslant x_j(t) \leqslant x_j(t_s) + m_j^2, \\ j = \overline{1,n}, \quad t \in [t_{s-1},\ t_s] \end{array} \Rightarrow \left\{ \begin{array}{l} G\left(x(t_s) - m^1\right) \leqslant q, \\ G\left(x(t_s) + m^2\right) \leqslant q, \end{array} \right. \quad s = \overline{1, N_1}.$$

Отметим, что задача с непрерывными ограничениями также была исследована в работе [75].

3.3 Выводы по главе 3

В параграфе 3.1 были представлены методы сведения задачи оптимального управления к задаче оптимизации в случае кусочно-линейных и кусочно-квадратичных управлений. Такие классы функций рассматривались в качестве альтернативы к основному классу кусочно-постоянных функций, представленному в главе 1.

При выборе управления в классе кусочно-линейных функций, задача с линейной системой и линейным (квадратичным) критерием качества может быть сведена к задаче линейного (квадратичного) программирования. А при рассмотрении управления в классе кусочно-квадратичных функций, задача с линейной системой и линейным (квадратичным) критерием качества сводится к выпуклой задаче программирования с квадратичными ограничениями и линейной (квадратичной) целевой функцией. Отметим, что для вариантов с квадратичным функционалом при переходе к задачам математического программирования сохраняется выпуклость, если матрица Q является неотрицательно определенной.

На вопрос, какой же класс управления стоит выбрать, нужно ориентироваться на свойства и критерии искомого решения. Так, например, гладкость управляющих воздействий может быть важным требованиям к разрабатываемой системе. В случае свободы в выборе класса, скорее всего, стоит воспользоваться более простыми классами, поскольку решения для них имеют меньшую сложность, а увеличивая параметр N можно добиться качественных результатов за короткое время.

Дальнейшее усложнение класса управлений представляется неоправданным, поскольку ведет к непропорциональному увеличению сложности задачи. Уже при рассмотрении кусочно-кубических управлений будет крайне затруднительно свести прямые границы к статическим ограничениям.

Дополнительно стоит отметить, что переход к классам более высоких порядков не обязательно приводит к лучшим решениям, потому что могут добавляться дополнительные условия на гладкость. На примере с демпфированием колебаний волчка можно заметить, что наименьшее значение функционала было получено при использовании кусочно-линейного класса.

В параграфе 3.2 была рассмотрена линейная задача оптимального управления с управлением из класса кусочно-постоянных функций. На эту задачу накладывались дополнительные ограничения.

- **Выпуклые ограничения на управления.** Такие ограничения очевидным образом сводятся к линейным неравенствам относительно u_{ik} и легко встраиваются в задачу линейного программирования.
- **Невыпуклые ограничения на управления.** Для описания невыпуклого множества используются линейные ограничения типа «ИЛИ» — из нескольких блоков уравнений необходимо выполнить хотя бы один. Для каждого блока вводится бинарная переменная. В итоге в задачу линейного программирования вносятся дополнительные линейные ограничения с бинарными переменными.
- **Выпуклые ограничения на фазовые переменные.** В точках переключения управления t_k задаются линейные неравенства по переменным x. Подобно терминальным условиям, данные условия сводятся к линейным ограничениям по u_{ik}. Также такие ограничения могут быть заданы для любого конечного числа заранее заданных моментов времени. Таким образом, число ограничений, добавляемых в модель, зависит от числа рассматриваемых точек и количества неравенств в каждой из точек.
- **Невыпуклые ограничения на фазовые переменные.** Для описания невыпуклых множеств в узлах t_k задаются линейные ограничения типа «ИЛИ». Подобно ограничениям на управления для каждого блока вводится бинарная переменная и в модель добавляются линейные ограничения, содержащие эти переменные.

При использовании всех четырех типов ограничений исходная задача будет сведена к оптимизационной задаче, описываемой уравнениями (1.9)–(1.11), (3.22), (3.26), (3.27), (3.31), (3.34), (3.35). По общепринятой классификации, такая модель принадлежит классу задач частично целочисленного линейного программирования (MILP) и может быть решена при помощи различных солверов (см. параграф 1.4). Дополнительно отметим, что при замене линейной целевой функции (1.5) на квадратичную (1.7) будет получена задача частично целочисленного квадратичного программирования (1.9), (1.10), (1.11), (3.22), (3.26), (3.27), (3.31), (3.34), (3.35).

Была затронута и тема учета неравенств на фазовые переменные на всем отрезке $[0, T]$. Однако, для подхода, рассматриваемого в данной диссертационной работе, не представляется возможны свести эти условия к статическим уравнениям для добавления в задачу математического программирования.

Наконец, затронем тему построения множеств достижимости и управляемости при дополнительных ограничениях. В главе 2 задача построения множеств для задачи (1.1)–(1.4) была сведена к задаче линейного отображения r-мерного куба, поскольку были заданы только двусторонние ограничения на элементы вектора управлений. При наличии же дополнительных ограничений либо при рассмотрении управления в классе кусочно-линейных функций, задача определения множеств будет состоять в линейном отображении многогранника. При использовании кусочно-квадратичного управления задача будет еще более трудоемкой. Поэтому вопрос построения множеств для таких случаев выходит за рамки текущей работы.

ГЛАВА 4. НЕЛИНЕЙНАЯ ЗАДАЧА ОПТИМАЛЬНОГО УПРАВЛЕНИЯ

4.1 Постановка задачи

Будем считать, что процесс описывается с помощью системы дифференциальных уравнений, нелинейной по фазовым переменным и линейной относительно управления:

$$\dot{x} = f(t, x) + B(t)u. \tag{4.1}$$

Предположим, что существуют частные производные $\frac{\partial f_i(t,x)}{\partial x_j}$, а матрица Якоби $F(t, x)$ функции $f(t,x)$ ограничена:

$$\|F(t, x)\| \leqslant M, \quad t \in [0, T], \quad x \in \mathbb{R}^n. \tag{4.2}$$

При таком ограничении будет обеспечено выполнение условие Липшица. Следовательно, существует единственное решение задачи Коши для системы (4.1) с начальным условием $x(0) = x_*$ для любого заданного управления $u(t)$, удовлетворяющего условиям (1.3), (1.4).

Добавив терминальные условия (1.2) и ограничения на управления (1.3), (1.4), а также один из критерий качества (1.5)–(1.7), мы сформируем нелинейную задачу оптимального управления.

Замечание 4.1. По умолчанию в качестве целевой функции будем рассматривать комбинацию фазовых переменных (1.5), подразумевая при этом, что ее можно заменить на критерий расхода ресурса (1.6) или же на квадратичный функционал (1.7).

Цель главы 4 — разработать численный метод нахождения управления для поставленной задачи (4.1), (1.2)–(1.5). Как будет показано далее, управление в некотором смысле может считаться субоптимальным. В итоге будут предложены алгоритмы для двух режимов:

– *Программный режим.* Перед началом движения управляемого объекта за несколько итераций алгоритма находятся приближенная функция управления $u(t)$ и соответствующая ей динамика изменения фазовых переменных $x(t)$.
– *Режим управления в реальном времени (позиционный режим).* Определяется начальная функция управления. Объект движется под воздействием этого управления до ближайшей точки переключения t_k. Затем управление пересчитывается исходя из текущей позиции объекта, учитывая отклонения в фазовой траектории, возникшие из-за неточности аппроксимации нелинейной функции и внешних возмущений. До следующей точки t_{k+1} объект движется под воздействием нового управления.

4.2 Обзор литературы

В продолжение обзора, представленного в параграфе 1.2, опишем здесь основные направления в решении нелинейных задач оптимального управления. Упомянутые ранее классические подходы (принцип максимума Понтрягина [15], динамическое программирование [16] и метод последовательных приближений [18]) могут быть распространены и на нелинейные задачи.

В современной иностранной литературе можно выделить следующие основные алгоритмы решения нелинейной задачи с кусочно-постоянным управлением [27]:

1. *Direct Single Shooting* [76]. С учетом вида управления (1.4) численными методами находится решение задачи Коши с системой (4.1) как системы обыкновенных дифференциальных уравнений с параметрами u_{ik}. Таким образом определяется нелинейная зависимость

положения $x(T)$ от значений управления. После этого, задача (4.1), (1.2)–(1.5) чаще всего сводится к невыпуклой задаче нелинейного программирования с Nr неизвестными.

2. *Direct Multiple Shooting* [77]. Рассматривается задача Коши с системой (4.1) на интервале $[t_{k-1}, t_k)$:

$$\dot{x}^{(k)} = f(t, x^{(k)}) + \sum_{i=1}^{r} b_i(t) u_{ik}, \quad x^{(k)}(t_{k-1}) = s_k, \quad t \in [t_{k-1}, t_k).$$

В данном контексте $x^{(k)}$ обозначает решение указанной задачи. Оно может быть найдено численно как решение задачи с $n + r$ параметрами s_k и u_{ik}. Далее добавляются условия непрерывности:

$$x^{(k-1)}(t_{k-1}) = s_k, \quad k = \overline{2, N}.$$

После этого исходная задача сводится к задаче нелинейного программирования с $N(n + r)$ неизвестными.

3. *Direct Collocation* [78]. Семейство алгоритмов, базовая идея которых состоит в замене фазовой переменной на полиномы. Так, например, пользуются популярностью реализации с заменой $x(t)$ на кусочно-линейную функцию или на кубический сплайн.
4. *Pseudospectral Discretization* [79]. Фазовые переменные заменяются линейной комбинацией базовых функций, которые, в отличие от предыдущего подхода, не обязательно являются полиномами.

Задача построения оптимального управления для задачи (4.1), (1.2)–(1.5) была изучена в работах научного коллектива Р. Ф. Габасова [38; 80; 81]. Предложены алгоритмы для двух случаев:

1. *Оптимизация «в малом»:* в случае малых отклонений функции $f(x)$ от своей линеаризации в начальной точке x_*, данная нелинейная функция заменяется линейным приближением и решается линейная задача оптимального управления, после чего полученное решение корректируется.

2. *Оптимизация «в большом»:* в случае больших отклонений нелинейной функции от своего линейного приближения, пространство фазовых переменных разбивается на области, и в каждой области функция $f(x)$ аппроксимируется линейной функцией. Тогда исходная нелинейная задача заменяется кусочно-линейной задачей оптимального управления. После решения осуществляется коррекция полученного управления. Примеры применения такого подхода описаны в статьях [80; 82].

Кроме того, в работе [83] описаны условия оптимальности решения рассматриваемой нелинейной задачи.

Автором диссертационной работы в статье [7] был сформулирован метод позиционного управления нелинейной системой на основе последовательных линеаризаций нелинейной системы, после чего он был в подробно освещен в статье [8].

Дополнительно отметим класс задач с билинейными системами, что является частным случаем нелинейности, где, в отличие от системы (4.1), допускаются произведения фазовых переменных и элементов управления $u_i(t)x(t)$. Для таких систем решается задача стабилизации совокупности программных управлений (многопрограммная стабилизация) [84–86].

Наконец, существует ряд ПО для решения задачи оптимального управления. Выделим некоторые из них:

1. **CasADI** [87]. Открытый пакет для решения задач нелинейной оптимизации. В том числе могут быть вычислены решения для моделей оптимального управления и задач типа «предиктор-корректор». Для нахождения оптимального управления используются алгоритмы direct single shooting, direct multiple shooting и direct collocation. Пакет предоставляется в виде библиотеки на языках Matlab, Python и C++.

2. **ACADO** [88]. Инструмент для работы на Matlab, в котором реализованы различные алгоритмы оптимального управления, в том числе direct single shooting и direct multiple shooting.
3. **DIDO** [89]. Инструмент для решения задачи оптимального управления на Matlab, в основе лежат псевдоспектральные алгоритмы.

4.3 Метод построения управления

В данном параграфе описаны основные результаты главы — алгоритмы построения управления в программном режиме и режиме реального времени.

4.3.1 Вспомогательная линейная задача

Рассмотрим нелинейную функцию $n+1$ переменной $f(t,x)$. Поскольку существуют производные $\frac{\partial f_i(t,x)}{\partial x_j}$, в качестве линейного приближения функции относительно переменных x можно взять сумму первых двух членов ряда Тейлора:

$$f(t,x) \approx f\left(t,x^{(0)}\right) + F\left(t,x^{(0)}\right)\left(x - x^{(0)}\right),$$

где $F\left(t,x^{(0)}\right) = \left.\frac{\partial f(t,x)}{\partial x}\right|_{x=x^{(0)}}$. Заметим, что в правой части содержится функция, линейная относительно x и нелинейная относительно t. Если же вместо точки $x^{(0)}$ подставить некоторую функцию $x^{(0)}(t)$, то свойство линейности по x сохранится:

$$f(t,x) \approx f\left(t,x^{(0)}(t)\right) + F\left(t,x^{(0)}(t)\right)\left(x - x^{(0)}(t)\right). \tag{4.3}$$

Будем называть правую часть формулы (4.3) линейным приближением функции $f(t,x)$ вдоль траектории $x^{(0)}(t)$.

Рассмотрим линейную неоднородную систему обыкновенных дифференциальных уравнений:

$$\dot{x} = F\left(t, x^{(0)}(t)\right)\left(x - x^{(0)}(t)\right) + f\left(t, x^{(0)}(t)\right) + B(t)u. \tag{4.4}$$

По сравнению с уравнением (4.1) нелинейная фунция была заменена линейным приближением вдоль траектории $x^0(t)$ (4.3). Если проводить аналогии с уравнением (1.1), то можно сопоставить

$$A(t) = F\left(t, x^{(0)}(t)\right),$$
$$d(t) = f\left(t, x^{(0)}(t)\right) - F\left(t, x^{(0)}(t)\right)x^0(t).$$

Таким образом ясно, что задача (4.4), (1.2)–(1.5) является линейной задачей оптимального управления. Решение для нее может быть найдено с помощью подхода, описанного в главе 1. Будем называть эту задачу вспомогательной по отношению к основной задаче (4.1), (1.2)–(1.5).

4.3.2 Алгоритм построения программного режима

Пусть задана некоторая начальная траектория движения $x^{(0)}(t)$ для $t \in [0, T]$. При этом для нее не гарантировано выполнение терминальных условий $Hx(T) = g^0$.

Шаг 1. Запишем систему с линейным приближением вдоль траектории $x^{(0)}(t)$:

$$\dot{\xi}^{(1)} = F\left(t, x^{(0)}(t)\right)\left(\xi^{(1)} - x^{(0)}(t)\right) + f\left(t, x^{(0)}(t)\right) + B(t)u. \tag{4.5}$$

Наложим на искомую траекторию $\xi^{(1)}(t)$ терминальные условия:

$$\xi^{(1)}(0) = x_*, \quad H\xi^{(1)}(T) = g^0, \tag{4.6}$$

а в качестве целевой функции рассмотрим

$$c^T\xi^{(1)}(T) \longrightarrow \min. \tag{4.7}$$

Тогда задача управления (4.5), (4.6), (1.3), (1.4), (4.7) является аналогичной задаче (1.1)–(1.5) и может быть решена методом, описанном в главе 1: сведением к задаче линейного программирования с последующим решением задачи ЛП стандартными методами.

Обозначим оптимальное решение данной вспомогательной задачи символом $u^{(1)}(t)$. Далее замкнем систему (4.1) этим управлением и найдем решение задачи Коши:

$$\dot{x}^{(1)} = f\left(t, x^{(1)}\right) + B(t)u^{(1)}(t),$$
$$x^{(1)}(0) = x_*.$$

Решение $x^{(1)}(t)$ является движением управляемого объекта под действием управления $u^{(1)}(t)$. Процесс можно продолжить итеративно.

Шаг s. Пусть после $s-1$ итераций найдено движение $x^{(s-1)}(t)$. Запишем вспомогательную линейную задачу оптимального управления с линейным приближением функции $f(t,x)$ вдоль траектории $x^{(s-1)}(t)$.

$$\begin{gathered} c^T\xi^{(s)}(T) \longrightarrow \min, \\ \dot{\xi}^{(s)} = F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)} - x^{(s-1)}(t)\right) + f\left(t, x^{(s-1)}(t)\right) + B(t)u, \\ \xi^{(s)}(0) = x_*, \quad H\xi^{(s)}(T) = g^0, \\ l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1,r}, \\ u_i(t) = u_{ik}, \quad t \in [t_{k-1},\ t_k), \quad k = \overline{1,N}, \quad i = \overline{1,r}. \end{gathered} \tag{4.8}$$

После нахождения оптимального управления задачи (4.8) — функции $u^{(s)}(t)$, сформируем замкнутую систему:

$$\begin{gathered} \dot{x}^{(s)} = f\left(t, x^{(s)}\right) + B(t)u^{(s)}(t), \\ x^{(s)}(0) = x_*. \end{gathered} \tag{4.9}$$

Таким образом, имея на входе вектор-функцию $x^{(s-1)}(t)$, рассчитанную на предыдущей итерации, необходимо сначала решить линейную задачу оптимального управления (4.8), построив для нее оптимальное управление $u^{(s)}(t)$, а затем вычислить решение задачи Коши (4.9) — функцию $x^{(s)}(t)$.

Начальная траектория

В качестве начальной траектории $x^{(0)}(t)$ можно взять некое приближенное решение, основанное на известных сведений об исходной задаче. В случае отсутствия таких знаний, можно принять $x^{(0)}(t) \equiv x_*$. Тогда на первом шаге алгоритма будет рассматриваться линейное приближение $f(x)$ в начальной точке x_*. Альтернативный вариант состоит в задании

$$x^{(0)}(t) = (\hat{x} - x_*)\,\frac{t}{T} + x_*,$$

где $\hat{x}$ — некоторая точка, удовлетворяющая системе $H\hat{x} = g^0$. В этом случае траектория $x^{(0)}(t)$ представляет отрезок, проведенный из начального положения в заданную плоскость, таким образом, для нее выполняются терминальные условия (1.2).

Критерий останова

В качестве главных метрик сходимости алгоритма предлагается рассмотреть два функционала:

$$\begin{gathered} \left\| Hx^{(s)}(T) - g^0 \right\|, \\ \left| J_s - J_{s-1} \right|. \end{gathered} \tag{4.10}$$

Первый функционал обозначает расстояние финального положения объекта, вычисленного на s-ой итерации алгоритма, $x^{(s)}(T)$ от плоскости $Hx = g^0$, на которой должен был оказаться объект согласно терминальному условию (1.2). Тем не менее, возможно нарушение данного требования из-за ошибки аппроксимации нелинейной функции $f(t, x)$: управление $u^{(s)}(t)$ гарантирует выполнение (1.2) для вспомогательной системы — $H\xi^{(s)}(T) = g^0$, но не для решения замкнутой системы $x^{(s)}(T)$.

В случае ожидаемого схождения $\left\|Hx^{(s)}(T) - g^0\right\| \to 0$ при $s \to \infty$, можно задать константу δ_T и считать терминальное условие выполненным при соблюдении ограничения

$$\left\|Hx^{(s)}(T) - g^0\right\| \leqslant \delta_T.$$

Второй функционал в (4.10) означает изменение значения целевого функционала по сравнению с результатом на предыдущей итерации. J_s — значение целевой функции для траектории $x^{(s)}(t)$ и функции управления $u^{(s)}(t)$. Так, для критерия (1.5)

$$J_s = c^T x^{(s)}(T),$$
$$|J_s - J_{s-1}| = \left|c^T \left(x^{(s)}(T) - x^{(s-1)}(T)\right)\right|.$$

Сходимость к единому значению целевой функции может говорить о попадании в оптимум (возможно локальный). Поэтому предусмотрим остановку алгоритма при выполнении

$$|J_s - J_{s-1}| \leqslant \delta,$$

где δ — некоторая заранее заданная константа, значение которой зависит от специфики задачи, размерностей величин и требований к точности результата.

В итоге завершение алгоритма происходит при выполнении хотя бы одного из трех условий:

- попадание решения в «область сходимости», т. е. значения критериев не превышают заданные пределы,
- достижение максимального заранее определенного количества итераций алгоритма S_l,
- окончание выделенного времени на выполнение расчета T_l.

Существование решения

Важным является вопрос существования допустимых решений. Причем он может быть рассмотрен в нескольких контекстах:

1. Существует ли решение исходной задачи (4.1), (1.2)–(1.5)?
2. Существует ли решение вспомогательной линейной задачи (4.8)?
3. Если на некоторой итерации s для вспомогательной задачи (4.8) множество допустимых решений пусто, означает ли это, что не существует решений и задачи (4.1), (1.2)–(1.5)?

Для начала дадим ответ на вопрос 2. Для простоты предположим, что матрица H является квадратной, тогда ограничение на правый конец можно представить как требование попасть в заданное положение: $x(T) = x^*$. Тогда согласно выкладкам, приведенным в главе 2, можно построить множество достижимости вида

$$\hat{b}_* \leqslant \hat{A}x(T) \leqslant \hat{b}^*. \tag{4.11}$$

Эта система неравенств описывает множество всевозможных состояний, в которое объект, движение которого описывается вспомогательной линейной системой, может попасть из положения x_* за время T под действием кусочно-постоянного управления, удовлетворяющего (1.3), (1.4).

Следовательно, для проверки существования допустимых решений достаточно подставить положение в систему (4.11). Кроме того, можно оценить близость желаемого положения к границе i-го ограничения:

$$\min\left(\hat{a}_i x^* - \hat{b}_{i*}, \hat{b}_i^* - \hat{a}_i x^*\right). \tag{4.12}$$

В случае, если $\operatorname{rank} H < n$, то есть в момент T объект должен попасть на заданную плоскость, то необходимо проверить, существует ли решение

системы неравенств:

$$\hat{b}_* \leqslant \hat{A}\chi \leqslant \hat{b}^*,$$
$$H\chi = g^0,$$

относительно неизвестного n-мерного вектора χ.

Теперь займемся вопросами 1 и 3. К сожалению, построение множества достижимости нелинейной задачи представляется слишком сложной задачей, поэтому остается судить об этом только по неким косвенным признакам, один из которых — оценивать множество достижимости вспомогательных линейных задач.

Если на некоторой итерации алгоритма вспомогательная линейная задача оказалась несовместной, то необходимо оценить значение критерия $\left\|Hx^{(s)}(T) - g^0\right\|$. В случае, когда значение оказалось достаточно близким к нулю, можно завершить алгоритм и выбрать решение, полученное на предыдущей итерации, в качестве лучшего найденного (не оптимального) допустимого решения. Если значение метрики на всех проделанных итерациях превышает заданный порог, можно сделать вывод, что допустимое решение не может быть найдено.

В случае незафиксированного правого конца ($\operatorname{rank} H < n$) можно предположить, что чем дальше значение фазовых переменных решения вспомогательной задачи в финальный момент времени $\xi^{(s)}(T)$ от границ множества допустимых решений, тем меньше будет отклонение решения замкнутой системы $\left\|Hx^{(s)}(T) - g^0\right\|$. Логика состоит в следующем: нахождение точки $x^{(s)}(T)$ в близости границы множества, описываемого (4.11) говорит, о том, что для достижения этой точки управление используется «на пределе», т.е. при небольшом изменении управления $u(t)$, оно может выйти за пределы (1.4). Тогда при переходе от одного линейного приближения к другому существует риск, что параметры задачи $A(t)$ и $d(t)$ будут перестроены так, что не удастся найти допустимое решение. Поэтому можно предположить, что «безопаснее» получить правый конец подальше от

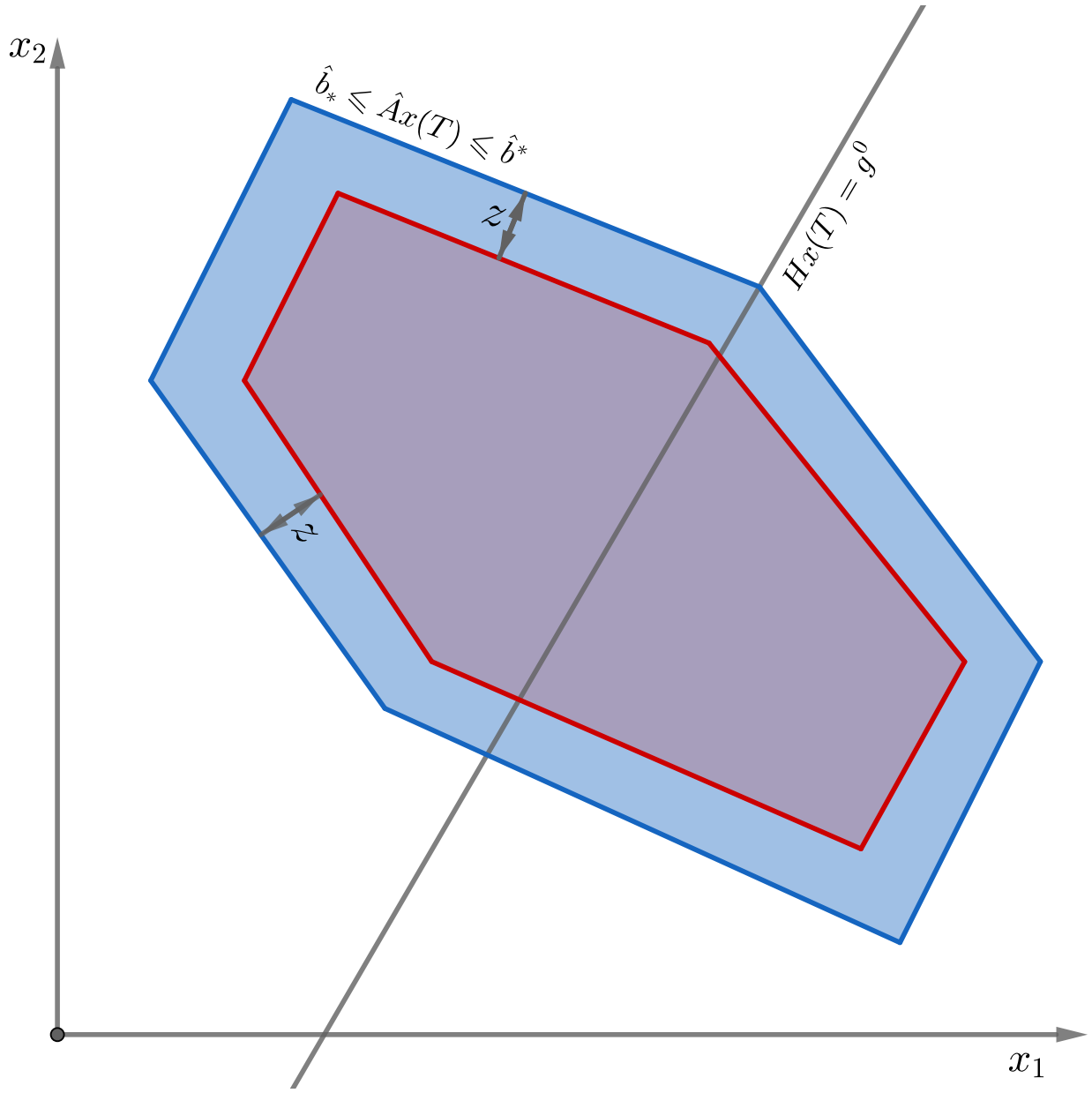

Рисунок 4.1 — Геометрический смысл ограничений (4.13)

границы множества допустимых решений. Это можно сделать, внеся расстояние до границ (4.12) в целевую функцию. Для этого введем в задачу переменную z и наложим на нее ограничения:

$$\begin{aligned} z &\leqslant \boldsymbol{\gamma}^T \left(\hat{A}x^{(s)}(T) - \hat{b}_* \right), \\ z &\leqslant \boldsymbol{\gamma}^T \left(\hat{b}^* - \hat{A}x^{(s)}(T) \right). \end{aligned} \tag{4.13}$$

Здесь $\boldsymbol{\gamma}$ — вектор условных весов неравенств, в простейшем случае, можно задать все веса равными единице. Таким образом, ограничения (4.13) гарантируют, что величина z меньше расстояния от $x^{(s)}(T)$ до границы множества допустимых решений. Геометрический смысл такого подхода

проиллюстрирован на рис. 4.1. Следующий шаг: обновить целевую функцию:

$$J^* = J - \mu z \longrightarrow \min . \tag{4.14}$$

В формуле (4.14) J — одна из целевых функций (1.5)–(1.7), μ — неотрицательный параметр, регулирующий направленность оптимизации. При маленьких значениях μ фокус будет на получении более близких к оптимальному решений, при больших значениях — на более «надежных значениях», когда значение $x(T)$ лежит далеко от границы множества допустимых решений.

4.3.3 Алгоритм построения позиционного режима

Предположим, что движение управляемого объекта нельзя предсказать заранее. Например, это может быть при воздействии случайных внешних возмущений:

$$\dot{x} = f(t, x) + B(t)u + \eta(t), \tag{4.15}$$

$\eta(t)$ — некоторый случайный процесс с нулевым математическим ожиданием.

В таком случае хорошим выходом является управление в режиме реального времени — перестроение управления в ходе процесса движения объекта. Кроме того, такой подход может оказаться полезным в следующих случаях:

– При построении программного управления не удалось удовлетворить терминальные условия и известно, что без пересчета управления не будет выполнено требование $Hx(T) = g^0$.

– Изменилось само терминальное условие. Теперь необходимо выполнить ограничение $\tilde{H}x(T) = \tilde{g}^0$. Например, такое возможно, если ставится цель достичь движущийся объект.

– По каким-либо причинам поменялся критерий качества решения.

Поскольку управление является кусочно-постоянной функцией с периодом дискретизации h, то управление может быть перестроено только в точках t_k.

Шаг 1. Предположим, что было построено некоторое программное управление $\bar{u}^{(0)}(t)$ и соответствующее ему движение $\bar{x}^{(0)}(t)$. В момент t_1 объект оказался в положении $\tilde{x}^{(1)}$. По определенным причинам требуется перестроить управление. В таком случае необходимо решить следующую нелинейную задачу оптимального управления:

$$
\begin{gathered}
c^T x(T) \longrightarrow \min, \\
\dot{x} = f(t,x) + B(t)u + \eta(t), \\
x(t_1) = \tilde{x}^{(1)}, \quad Hx(T) = g^0, \\
l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1,r}, \\
u_i(t) = u_{ik'}, \quad t \in [t_{k'-1},\ t_{k'}), \quad k' = \overline{2,N}, \quad i = \overline{1,r},
\end{gathered}
$$

которую можно решить итеративным алгоритмом, аналогичным, предложенному для программного режима:

$$
\begin{gathered}
c^T \xi^{(s)}(T) \longrightarrow \min, \\
\dot{\xi}^{(s)} = F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)} - x^{(s-1)}(t)\right) + f\left(t, x^{(s-1)}(t)\right) + B(t)u, \\
\xi^{(s)}(t_1) = \tilde{x}^1, \quad H\xi^{(s)}(T) = g^0, \\
l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1,r}, \\
u_i(t) = u_{ik'}, \quad t \in [t_{k'-1},\ t_{k'}), \quad k = \overline{2,N}, \quad i = \overline{1,r}.
\end{gathered}
$$

Начальной траекторией для процесса пересчета управления может послужить траектория в программном режиме $x(t)$.

Замечание 4.2. Количество итераций алгоритма равняется S_1 и определяется условиями выхода из алгоритма построения программного режима.

Замечание 4.3. Поскольку вычисление новой траектории произойдет не мгновенно, очевидно, потребуется перестроить траекторию до наступления времени t_1. Логично выбрать временем начала пересчета момент $t_1 - T_l$, где T_l — лимит времени на построение программного режима. Однако, в таком случае не будет наверняка известно положение объекта $\tilde{x}^1$ в момент t_1. Поэтому правильнее сказать, что $\tilde{x}^1$ — прогноз положения объекта в момент t_1, построенный в точке $t_1 - T_l$.

Обозначим найденное управление символом $\bar{u}^{(1)}(t)$, а фазовую траекторию — $\bar{x}^{(1)}(t)$. Продолжим процесс движения объекта.

Шаг k. Пусть в момент $t_k - T_l$ предполагается, что во время t_k объект окажется в позиции $\tilde{x}^{(k)}$. Построим задачу управления объектом из этой позиции:

$$\begin{gathered} c^T x(T) \longrightarrow \min, \\ \dot{x} = f(t,x) + B(t)u + \eta(t), \\ x(t_k) = \tilde{x}^{(k)}, \quad Hx(T) = g^0, \\ l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1,r}, \\ u_i(t) = u_{ik'}, \quad t \in [t_{k'-1},\ t_{k'}), \quad k' = \overline{k+1,N}, \quad i = \overline{1,r}. \end{gathered} \tag{4.16}$$

В данном случае $k = \overline{1, N-1}$. Нелинейная задача оптимального управления (4.16) может быть решена при помощи итеративного алгоритма:

$$\begin{gathered} c^T \xi^{(s)}(T) \longrightarrow \min, \\ \dot{\xi}^{(s)} = F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)} - x^{(s-1)}(t)\right) + f\left(t, x^{(s-1)}(t)\right) + B(t)u, \\ \xi^{(s)}(t_k) = \tilde{x}^{(k)}, \quad H\xi^{(s)}(T) = g^0, \\ l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1,r}, \\ u_i(t) = u_{ik'}, \quad t \in [t_{k'-1},\ t_{k'}), \quad k' = \overline{k+1,N}, \quad i = \overline{1,r}. \end{gathered} \tag{4.17}$$

Стартовой траекторией для начала итерационного алгоритма может выступить функция $\bar{x}^{(k-1)}(t)$, взятая на отрезке $[t_k, T]$. Найденные управление и траекторию обозначим $\bar{u}^{(k)}(t)$ и $\bar{x}^{(k)}(t)$ соответственно.

Таким образом, построенное управление каждый раз будет использоваться на интервале длиной h, после чего будет перестраиваться в

зависимости от позиции объекта. Запишем итоговую реализацию управления:

$$\bar{u}(t) = \begin{cases} \bar{u}^{(k)}(t), \quad t \in [t_k,\ t_k + 1), \\ k = \overline{0, N-1}, \end{cases} \tag{4.18}$$

Функцию $\bar{u}(t)$ назовем позиционным управлением задачи (4.15), (1.2)–(1.5).

Замечание 4.4. Оптимальность управления $\bar{u}(t)$ зависит от оптимальности программных управлений $\bar{u}^{(k)}(t)$. В случае если показана оптимальность (субоптимальность) решений задач (4.16), назовем функцию $\bar{u}(t)$ оптимальным (субоптимальным) позиционным управлением.

Существование решения

Помимо вопросов о существовании допустимого управления, затронутых в пункте 4.3.2, необходимо задуматься о потере управляемости в моменты пересчета управления: возможна ситуация, при которой множество допустимых решений задачи (4.16) на шаге $k-1$ было не пусто, однако, на шаге k оказалось, что допустимых решений не существует. Такое может случиться в следствие следующих факторов:

1. Ошибка аппроксимации нелинейной функции.
2. Отклонение от программного движения в следствие внешних возмущений.
3. Изменение условий задачи.

В качестве обеспечения более «безопасного» движения (с точки зрения выхода из множества управляемости) предлагается ввести в задачи (4.16) и (4.17) переменную z и ограничения (4.13), а также поменять целевую функцию согласно (4.14).

4.4 Теоретическое обоснование

4.4.1 Сходимость траекторий в программном режиме

Вычтем из нелинейной системы (4.9) вспомогательную линейную систему (4.8):

$$\dot{x}^{(s)} - \dot{\xi}^{(s)} = f\left(t, x^{(s)}\right) + B(t)u^{(s)}(t) -$$
$$-F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)} - x^{(s-1)}(t)\right) - f\left(t, x^{(s-1)}(t)\right) - B(t)u^{(s)}(t) =$$
$$= f\left(t, x^{(s)}\right) - f\left(t, x^{(s-1)}(t)\right) - F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)} - x^{(s-1)}(t)\right).$$

Введем переменную $\Delta^{(s)}(t)$, означающую разность решения исходной задачи и вспомогательной на s-ой итерации алгоритма $x^{(s)}(t) - \xi^{(s)}(t)$. Тогда, можем выписать систему дифференциальных уравнений относительно переменной $\Delta^{(s)}(t)$:

$$\begin{aligned}\dot{\Delta}^{(s)} &= f\left(t, \xi^{(s)}(t) + \Delta^{(s)}\right) - f\left(t, x^{(s-1)}(t)\right) - \\ &- F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)}(t) - x^{(s-1)}(t)\right),\end{aligned} \tag{4.19}$$

при начальном условии $\Delta^{(s)}(0) = x^{(s)}(0) - \xi^{(s)}(0) = x_* - x_* = 0$.

Лемма 4.1. *Пусть существует производная по времени* $\frac{\partial f(t,x)}{\partial t}$ *и вторые производные* $\frac{\partial^2 f(t,x)}{\partial x_j \partial t}$ *и* $\frac{\partial^2 f(t,x)}{\partial x_{j_1} \partial x_{j_2}}$. *Тогда отклонение решения нелинейной замкнутой системы (4.9) от решения задачи оптимального управления (4.8) является бесконечно малой величиной относительно* t^2, *то есть* $\Delta^{(s)}(t) = o\left(t^2\right)$.

Доказательство. Выпишем формулу Тейлора функции $\Delta^{(s)}(t)$ в точке 0:

$$\Delta^{(s)}(t) = \Delta^{(s)}(0) + \dot{\Delta}^{(s)}(0)t + \ddot{\Delta}^{(s)}(0)\frac{t^2}{2} + o\left(t^2\right).$$

Покажем, что первые три слагаемых равны нулю. Для отклонения в стартовой точке очевидно $\Delta^{(s)}(0) = x^{(s)}(0) - \xi^{(s)}(0) = x_* - x_* = 0$. Подставим значение $t = 0$ в формулу (4.19):

$$\dot{\Delta}^{(s)}(0) = f(0, x_*) - f(0, x_*) - F(0, x_*)(x_* - x_*) = 0.$$

Далее найдем вторую производную отклонения, продифференцировав по t формулу (4.19).

$$\begin{gathered}
\ddot{\Delta}^{(s)}(t) = \frac{\partial f\left(t, \xi^{(s)}(t) + \Delta^{(s)}(t)\right)}{\partial t} + \\
+ F\left(t, \xi^{(s)}(t) + \Delta^{(s)}(t)\right)\left(\dot{\xi}^{(s)}(t) + \dot{\Delta}^{(s)}(t)\right) - \\
- \frac{\partial f\left(t, x^{(s-1)}(t)\right)}{\partial t} - F\left(t, x^{(s-1)}(t)\right)\dot{x}^{(s-1)}(t) - \\
- \left(\frac{\partial F\left(t, x^{(s-1)}(t)\right)}{\partial t} + \sum_{j=1}^{n} \left.\frac{\partial F(t,x)}{\partial x_j}\right|_{x=x^{(s-1)}(t)} \dot{x}_j^{(s)}(t)\right)\left(\xi^{(s)}(t) - x^{(s-1)}(t)\right) - \\
- F\left(t, x^{(s-1)}(t)\right)\left(\dot{\xi}^{(s)}(t) - \dot{x}^{(s-1)}(t)\right) = \\
= F\left(t, \xi^{(s)}(t) + \Delta^{(s)}(t)\right)\dot{\Delta}^{(s)}(t) + \frac{\partial f\left(t, \xi^{(s)}(t) + \Delta^{(s)}(t)\right)}{\partial t} - \\
- \frac{\partial f\left(t, x^{(s-1)}(t)\right)}{\partial t} + \left(F\left(t, \xi^{(s)}(t) + \Delta^{(s)}(t)\right) - F\left(t, x^{(s-1)}(t)\right)\right)\dot{\xi}^{(s)}(t) - \\
- \left(\frac{\partial F\left(t, x^{(s-1)}(t)\right)}{\partial t} + \sum_{j=1}^{n} \left.\frac{\partial F(t,x)}{\partial x_j}\right|_{x=x^{(s-1)}(t)} \dot{x}_j^{(s)}(t)\right)\left(\xi^{(s)}(t) - x^{(s-1)}(t)\right).
\end{gathered}$$

Здесь и в дальнейшем $\frac{\partial F(t,x)}{\partial t} = \frac{\partial f(t,x)}{\partial x \partial t}$, а $\frac{\partial F(t,x)}{\partial x_j} = \frac{\partial f(t,x)}{\partial x \partial x_j}$. Теперь получим значение в стартовой точке:

$$\begin{gathered}
\ddot{\Delta}^{(s)}(0) = F(0, x_*)\dot{\Delta}^{(s)}(0) + \left.\frac{\partial f(t, x_*)}{\partial t}\right|_{t=0} - \left.\frac{\partial f(t, x_*)}{\partial t}\right|_{t=0} + \\
+ \left(F(0, x_*) - F(0, x_*)\right)\dot{\xi}^{(s)}(0) - \\
- \left(\left.\frac{\partial F(t, x_*)}{\partial t}\right|_{t=0} + \sum_{j=1}^{n} \left.\frac{\partial F(0,x)}{\partial x_j}\right|_{x=x_*} \dot{x}_j^{(s)}(0)\right)(x_* - x_*) = 0.
\end{gathered}$$

Таким образом показано, что $\Delta^{(s)}(t) = o\left(t^2\right)$. □

Исследуем зависимость сходимости алгоритма от изменения вектора управлений. Введем величину

$$\delta_s = \max_{t\in[0,T]} \left\| B(t) \left(u^{(s)}(t) - u^{(s-1)}(t) \right) \right\|, \tag{4.20}$$

которая представляет собой максимальное на отрезке $[0, T]$ отклонение текущего управления от управления на предыдущем шаге (помноженное на $B(t)$). Проанализируем влияние этого параметра на отклонение решения исходной задачи от решения вспомогательной.

Лемма 4.2. *Отклонение решения вспомогательной линейной задачи от решения нелинейной системы, замкнутой управлением, полученным на предыдущем шаге алгоритма, удовлетворяет неравенству*

$$\left\| \xi^{(s)}(t) - x^{(s-1)}(t) \right\| \leqslant \frac{1}{M} \left(e^{Mt} - 1 \right) \delta_s, \quad \forall s \geqslant 2, \quad t \in [0, T]. \tag{4.21}$$

Доказательство. Выпишем систему дифференциальных уравнений для x^{s-1} и ξ^s:

$$\dot{x}^{(s-1)}(t) = f\left(t, x^{(s-1)}(t)\right) + B(t)u^{(s-1)}(t),$$
$$\dot{\xi}^{(s)}(t) = F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)}(t) - x^{(s-1)}(t)\right) + f\left(t, x^{(s-1)}(t)\right) + B(t)u^{(s)}(t).$$

Вычтем первое уравнение из второго:

$$\begin{aligned} \dot{\xi}^{(s)}(t) - \dot{x}^{(s-1)}(t) = F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)}(t) - x^{(s-1)}(t)\right) + \\ + B(t)\left(u^{(s)}(t) - u^{(s-1)}(t)\right). \end{aligned} \tag{4.22}$$

Равенство (4.22) — система дифференциальных уравнений относительно $\xi^{(s)}(t) - x^{(s-1)}(t)$ при начальном условии $\xi^{(s)}(0) - x^{(s-1)}(0) = x_* - x_* = 0$. Тогда можем найти решение с помощью формулы Коши:

$$\xi^{(s)}(t) - x^{(s-1)}(t) = Y_s(t) \int_0^t Y_s^{-1}(\tau) B(\tau) \left(u^{(s)}(\tau) - u^{(s-1)}(\tau) \right) d\tau, \tag{4.23}$$

где $Y_s(t)$ — матрицант системы (4.22), нормированный в точке 0. Оценим норму разности в левой части (4.23):

$$\left\| \xi^{(s)}(t) - x^{(s-1)}(t) \right\| \leqslant \|Y_s(t)\| \int_0^t \left\| Y_s^{-1}(\tau) \right\| \cdot \left\| B(\tau) \left(u^{(s)}(\tau) - u^{(s-1)}(\tau) \right) \right\| d\tau.$$

Тогда с учетом свойств матрицанта и обозначения (4.20):

$$\left\|\xi^{(s)}(t) - x^{(s-1)}(t)\right\| \leqslant e^{Mt} \int\limits_0^t e^{-M\tau}\delta_s d\tau = \frac{1}{M}\left(e^{Mt} - 1\right)\delta_s.$$

□

Лемма 4.3. *Для разности решений нелинейной системы, замкнутой управлениями, найденными на текущем и предыдущем шагах алгоритма, справедливо ограничение*

$$\left\|x^{(s)}(t) - x^{(s-1)}(t)\right\| \leqslant \frac{1}{M}\left(e^{Mt} - 1\right)\delta_s, \quad \forall s \geqslant 2, \quad t \in [0, T]. \tag{4.24}$$

Доказательство. Рассмотрим разность $x^{(s)}(t)$ и $x^{(s-1)}(t)$:

$$\begin{gathered}\dot{x}^{(s)}(t) - \dot{x}^{(s-1)}(t) = \\ = f\left(t, x^{(s)}(t)\right) - f\left(t, x^{(s-1)}(t)\right) + B(t)\left(u^{(s)}(t) - u^{(s-1)}(t)\right).\end{gathered} \tag{4.25}$$

Проинтегрируем выражение (4.25) на отрезке $[0, t]$ с учетом начального условия $x^{(s)}(0) - x^{(s-1)}(0) = x_* - x_* = 0$:

$$\begin{gathered}x^{(s)}(t) - x^{(s-1)}(t) = \int\limits_0^t \left(f\left(\tau, x^{(s)}(\tau)\right) - f\left(\tau, x^{(s-1)}(\tau)\right)\right) d\tau + \\ + \int\limits_0^t B(\tau)\left(u^{(s)}(\tau) - u^{(s-1)}(\tau)\right) d\tau.\end{gathered}$$

Воспользуемся теоремой Лагранжа о среднем:

$$f\left(t, x^{(s)}(t)\right) - f\left(t, x^{(s-1)}(t)\right) = F\left(t, \mu(t)\right)\left(x^{(s)}(t) - x^{(s-1)}(t)\right),$$

где $\mu(t)$ — функция, принимающая значения между $x^{(s-1)}(t)$ и $x^{(s)}(t)$ для любого t. Далее, оценим $x^{(s)}(t) - x^{(s-1)}(t)$ по норме:

$$\left\|x^{(s)}(t)-x^{(s-1)}(t)\right\| \leqslant \int\limits_0^t \|F\left(\tau,\mu(\tau)\right)\| \cdot \left\|\left(x^{(s)}(\tau)-x^{(s-1)}(\tau)\right)\right\| d\tau +$$

$$+\int\limits_0^t \left\|B(\tau)\left(u^{(s)}(\tau)-u^{(s-1)}(\tau)\right)\right\| d\tau \leqslant$$

$$\leqslant M\int\limits_0^t \left\|\left(x^{(s)}(\tau)-x^{(s-1)}(\tau)\right)\right\| d\tau + \delta_s t.$$

Наконец, воспользовавшись усиленной леммой Гронуолла [90], получим неравенство:

$$\left\|x^{(s)}(t)-x^{(s-1)}(t)\right\| \leqslant \frac{1}{M}\left(e^{Mt}-1\right)\delta_s$$

□

Лемма 4.4. *Норма отклонения решения нелинейной замкнутой системы (4.9) от решения соответствующей вспомогательной линейной задачи оптимального управления (4.8) может быть ограничена сверху:*

$$\left\|x^{(s)}(t)-\xi^{(s)}(t)\right\| \leqslant \frac{2}{M}\left(e^{Mt}-1\right)\delta_s, \quad \forall s \geqslant 2, \quad t\in[0,T]. \tag{4.26}$$

Доказательство. Воспользуемся неравенством треугольника и соотношениями (4.21) и (4.24):

$$\left\|x^{(s)}(t)-\xi^{(s)}(t)\right\| = \left\|x^{(s)}(t)-x^{(s-1)}(t)+x^{(s-1)}(t)-\xi^{(s)}(t)\right\| \leqslant$$

$$\leqslant \left\|x^{(s)}(t)-x^{(s-1)}(t)\right\| + \left\|\xi^{(s)}(t)-x^{(s-1)}(t)\right\| \leqslant$$

$$\leqslant \frac{1}{M}\left(e^{Mt}-1\right)\delta_s + \frac{1}{M}\left(e^{Mt}-1\right)\delta_s = \frac{2}{M}\left(e^{Mt}-1\right)\delta_s.$$

□

Следствие 4.1. *Существует верхняя оценка расстояния финального положения объекта от плоскости, заданной терминальным условием* $Hx(T) = g^0$*:*

$$\left\|Hx^{(s)}(T)-g^0\right\| \leqslant \frac{2\|H\|}{M}\left(e^{MT}-1\right)\delta_s \quad s\geqslant 2. \tag{4.27}$$

Доказательство. Заметим, что согласно условиям задачи (4.8) справедливо $H\xi^{(s)}(T) = g^0$. Тогда, с учетом (4.27):

$$\left\| Hx^{(s)}(T) - g^0 \right\| = \left\| H\left(x^{(s)}(T) - \xi^{(s)}(T)\right) \right\| \leqslant$$
$$\leqslant \|H\| \cdot \left\| x^{(s)}(T) - \xi^{(s)}(T) \right\| \leqslant \|H\| \frac{2}{M} \left(e^{MT} - 1\right) \delta_s.$$

□

Следствие 4.2. *Пусть существует предел последовательности* $\lim\limits_{s\to+\infty} u^{(s)}(t) = u(t), \forall t \in [0,T]$. *Тогда решения вспомогательных линейных задач оптимального управления (4.8) и решения замкнутых нелинейных систем дифференциальных уравнений (4.9) сходятся к единому пределу:*

$$\lim_{s\to+\infty} \xi^{(s)}(t) = \lim_{s\to+\infty} x^{(s)}(t) = x(t), \quad t \in [0,T]. \tag{4.28}$$

При этом функция $x(t)$ является допустимым решением задачи (4.1), (1.2)–(1.5).

Доказательство. Поскольку из сходимости управлений следует, что $\delta_s \to 0$ при $s \to +\infty$, то решение $x^{(s)}(t)$ сходится к $\xi^{(s)}(t)$ согласно (4.26) и к $x^{(s-1)}(t)$ согласно (4.24). Предел $x(t)$ является допустимым решением исходной задачи, т. к. с учетом неравенства (4.27) $Hx(T) = g^0$. □

4.4.2 Сходимость траекторий в позиционном режиме

Предыдущие выкладки были посвящены программному режиму. Коснемся теперь режима управления в реальном времени. Важной характеристикой задачи управления (4.15), (1.2)–(1.5) является количество точек переключения управления N. Для управления в режиме реального времени это же число показывает количество точек пересчета управления. Предположим, что число N является гиперпараметром модели и его можно варьировать. С одной стороны, увеличение N приведет к усложнению задачи. Так, при управлении в режиме реального времени придется чаще

решать задачу управления (4.16), и к тому же сама задача будет иметь большую размерность, чем при малых N. С другой стороны, увеличение этого параметра позволит добиться лучшей сходимости.

Лемма 4.5. *Пусть $\bar{x}(t)$ — решение системы (4.15), замкнутое позиционным управлением (4.18), с начальным условием $\bar{x}(0) = x_*$. Тогда при увеличении N финальное положение $\bar{x}(t)$ стремится к плоскости $Hx = g^0$:*

$$\lim_{N\to+\infty} \left\| H\bar{x}(T) - g^0 \right\| = 0.$$

Доказательство. Решение задачи (4.15), (1.2)–(1.5) состоит в последовательном решении N задач (4.16). На последнем шаге будет решаться задача нахождения управления на полуинтервале $[t_{k-1}, t_k)$ или что то же $[(N-1)h, Nh)$. Не трудно показать, что для такой задачи ограничение (4.27) примет вид

$$\left\| H\bar{x}^{(s)}(T) - g^0 \right\| \leqslant \frac{2\|H\|}{M} \left(e^{Mh} - 1\right) \delta_s.$$

В данном контексте $\bar{x}^{(s)}(t)$ — решение замкнутой системы

$$\dot{\bar{x}}^{(s)} = f(t, \bar{x}^{(s)})(t) + B(t)\bar{u}^{(s)}(t),$$
$$\bar{x}\left((N-1)h\right) = \tilde{x}^{(N-1)},$$

а управление $\bar{u}^{(s)}(t)$ найдено на s-ом шаге решения задачи (4.17) при $k = N-1$. Тогда с учетом того, что M и $\|H\|$ — постоянные величины, а δ_s согласно (4.20) и (1.3) ограничена сверху, то выражение

$$\left\| H\bar{x}^{(s)}(T) - g^0 \right\|$$

стремится к нулю при $h \to 0$. А поскольку $h = T/N$, этот же вывод справедлив и при $N \to +\infty$. □

Замечание 4.5. Доказательство леммы 4.5 приведено для случая $s \geqslant 2$, однако, можно показать, что утверждение справедливо и для $s = 1$.

Замечание 4.6. Лемма 4.5 справедлива только при отсутствии внешних возмущений: $\eta(t) \equiv 0$.

4.4.3 Сходимость управлений в программном режиме

Важнейшим является вопрос сходимости управлений при построении программного управления. Как было показано в следствии 4.2, от этого свойства зависит, существует ли предел последовательности конструируемых решений и сойдется ли алгоритм к допустимому решению исходной задачи. Вернемся к формуле (4.23), описывающей разность между решением вспомогательной линейной задачи и решением нелинейной системы, полученным на предыдущем шаге алгоритма. Подставим $t = T$ и учтем терминальное условие (1.2):

$$H\xi^{(s)}(T) = Hx^{(s-1)}(t) + Y_s(T)\int\limits_0^T Y_s^{-1}(t)B(t)\left(u^{(s)}(t) - u^{(s-1)}(t)\right)dt = g^0.$$

Подобно алгоритму сведения терминальных условий, представленному в главе 1, данное равенство может быть трансформировано в систему линейных уравнений:

$$\sum_{i=1}^{r}\sum_{k=1}^{N} h_{ik}^{(s)}\left(u_{ik}^{(s)} - u_{ik}^{(s-1)}\right) = g^0 - Hx^{(s-1)}(T),$$

$$h_{ik}^{(s)} = \int_{t_{k-1}}^{t_k} HY_s(T)Y_s^{-1}(t)b_i(t)dt, \quad i = \overline{1,r}, \quad k = \overline{1,N}.$$

Здесь $u_{ik}^{(s-1)}$ и $u_{ik}^{(s)}$ — значения i-го элемента вектора управлений на k-ом временном интервале на предыдущем и текущем шаге алгоритма соответственно. Тогда задача (4.8) может быть переписана в виде задачи линейного программирования:

$$\begin{gathered}\sum_{i=1}^{r}\sum_{k=1}^{N} c_{ik}^{(s)}u_{ik}^{(s)} \longrightarrow \min,\\ \sum_{i=1}^{r}\sum_{k=1}^{N} h_{ik}^{(s)}\left(u_{ik}^{(s)} - u_{ik}^{(s-1)}\right) = g^0 - Hx^{(s-1)}(T),\\ l_{*i} \leqslant u_{ik}^{(s)} \leqslant l_i^*, \quad i = \overline{1,r}, \quad k = \overline{1,N}.\end{gathered} \tag{4.29}$$

Числа $c_{ik}^{(s)}$ имеют верхний индекс s, поскольку зависят от матрицанта $Y_s(t)$. Отметим, что в отличие от данной целевой функции, критерии (1.6) и (1.7) не зависят от текущего линейного приближения функции $f(t, x)$.

Проанализируем возможные сценарии взаимосвязи $u_{ik}^{(s-1)}$ и $u_{ik}^{(s)}$.

1. $u_{ik}^{(s-1)}$ не является допустимым решением задачи (4.29). В таком случае должно быть построено оптимальное решение $u_{ik}^{(s)}$, отличающееся от управления на предыдущей итерации. При этом не обязательно произойдет приближение к плоскости $Hx = g^0$ или улучшение целевой функции.
2. $u_{ik}^{(s-1)}$ является допустимым, но не оптимальным решением задачи (4.29). Легко заметить, что управление с предыдущего шага будет допустимым, только в случае $Hx^{(s-1)}(T) = g^0$. Следовательно, управление $u^{(s-1)}(t)$ является и допустимым управлением исходной задачи (4.1), (1.2)–(1.5). Однако, существует такое управление $u_{ik}^{(s)}$, что
$$\sum_{i=1}^{r}\sum_{k=1}^{N} c_{ik}^{(s)}\left(u_{ik}^{(s)} - u_{ik}^{(s-1)}\right) < 0.$$
Тогда новое управление будет связано со старым соотношением $\sum\limits_{i=1}^{r}\sum\limits_{k=1}^{N} h_{ik}^{(s)}\left(u_{ik}^{(s)} - u_{ik}^{(s-1)}\right) = 0$, но тем не менее не гарантировано, что такое решение будет допустимым решением нелинейной задачи. Более, того, поскольку целевая функция зависит от текущей линеаризации, значение критерия качества может оказаться хуже, чем на предыдущей итерации.
3. $u_{ik}^{(s-1)}$ является допустимым и оптимальным решением задачи (4.29). Наиболее благоприятный сценарий. Как и в предыдущем случае, $u^{(s-1)}(t)$ является допустимым управлением исходной задачи (4.15), но при этом $u^{(s)}(t) = u^{(s-1)}(t)$. Это гарантирует, что на следующих итерациях будет построено такое же решение, а согласно критериям останова произойдет выход из цикла. Можно

утверждать, что найдено допустимое субоптимальное решение задачи (4.1), (1.2)–(1.5).

4.5 Примеры работы алгоритма

Испытание на ряде тестовых задач показало способность алгоритма дойти до допустимого субоптимального решения нелинейной задачи. Продемонстрируем свойства алгоритма на нескольких примерах.

Алгоритмы построения управления в программном и позиционном режимах были реализованы на языке Python 3. Для решения задачи Коши использовалась функция *odeint*, а для вычисления определенных интегралов — *quad*, обе функции предоставляются в библиотеке *scipy.integrate*. Поиск оптимальных решений для задач линейного и квадратичного программирования осуществлялся с помощью библиотеки *coptpy*, обращающейся к солверу COPT. Оформление графиков проводилось с помощью пакета *matplotlib.pyplot*. Код программы приведен в приложении Б.

4.5.1 Скалярная нелинейная задача

Рассмотрим следующую задачу оптимального управления:

$$\int_0^5 u^2(t)dt \longrightarrow \min_u,$$
$$\dot{x} = \frac{1}{x^2+1} + u,$$
$$x(0) = 0, \quad x(5) = 8,$$
$$0 \leqslant u(t) \leqslant 2, \quad N = 10.$$

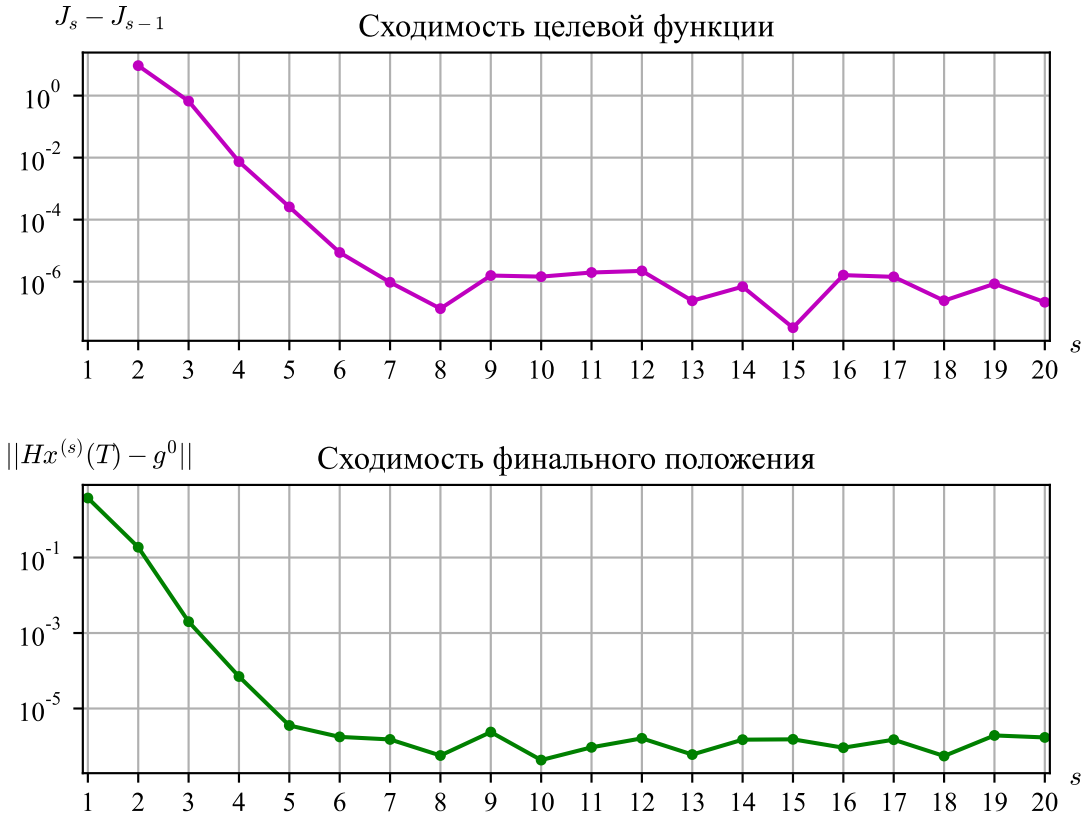

Рисунок 4.2 — Сходимость алгоритма

Нетрудно проверить, что производная нелинейной функции

$$\frac{d}{dx}\left(\frac{1}{x^2+1}\right) = \frac{-2x}{(x^2+1)^2}$$

существует для любого вещественного значения x и ограничена по модулю.

Программное управление для представленной задачи было построено с помощью алгоритма, изложенного в пункте 4.3.2. Для эксперимента было проведено 20 итераций алгоритма. Время расчета составило 2,1 сек. Целевая функция приняла значение 10,3807.

Результаты проиллюстрированы на графиках. На рис. 4.2 показана сходимость алгоритма с точки зрения критериев останова (4.10). рис. 4.3 демонстрирует стремление функции управления на разных итерациях алгоритма к единой функции, а на рис. 4.4 представлена сходимость $x^{(s)}(t)$ к $\xi^{(s)}(t)$ при увеличении s.

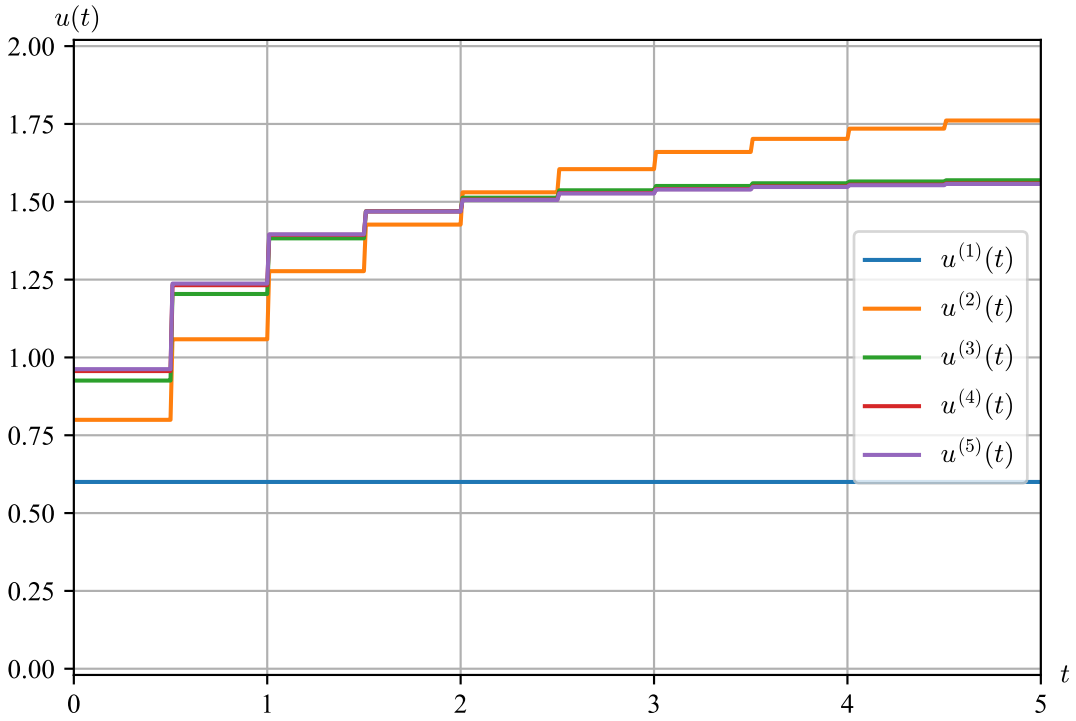

Рисунок 4.3 — Сходимость управления

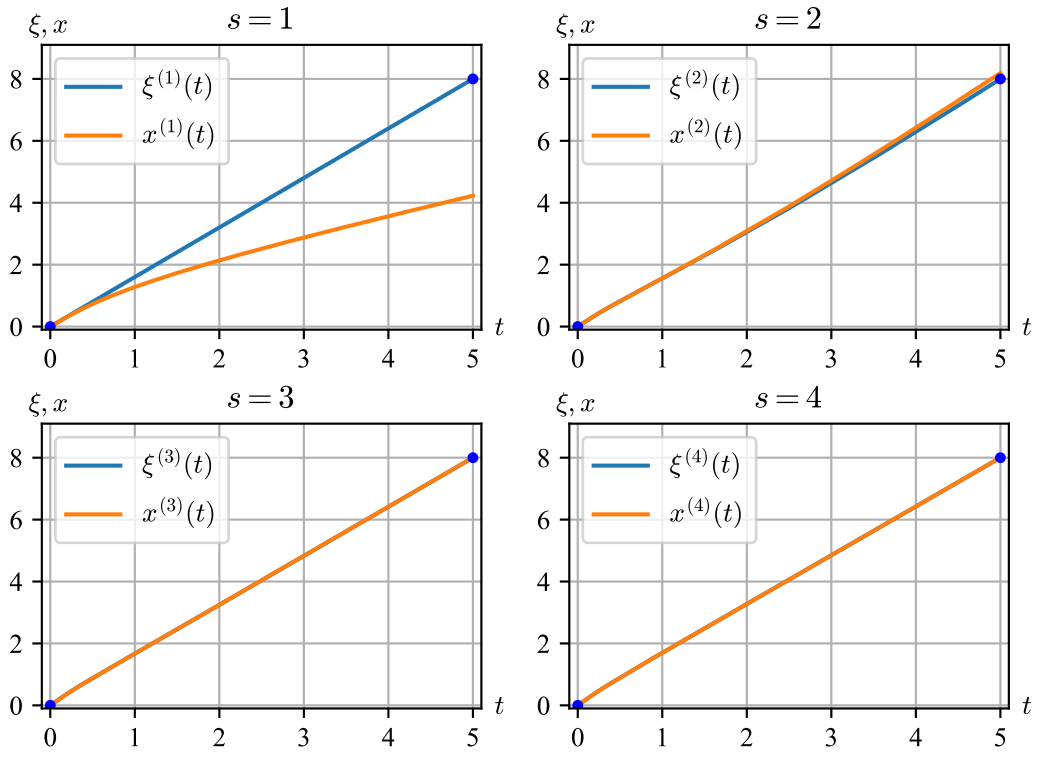

Рисунок 4.4 — Сходимость фазовой переменной

К итогам тестирования алгоритма на выбранной задаче могут быть отнесены следующие выводы:

1. В результате работы алгоритма было построено допустимое субоптимальное решение нелинейной задачи оптимального управления.
2. После шести итераций алгоритма были получены следующие значения критериев останова:

$$\left\| Hx^{(s)}(T) - g^0 \right\| = 1{,}77 \cdot 10^{-6},$$
$$|J_s - J_{s-1}| = 8{,}71 \cdot 10^{-6}.$$

   Далее эти показатели оставались примерно на том же уровне, что свидетельствует о достижении предельных значений. Дальнейший спуск до нуля невозможен по причине наличия вычислительных неточностей.
3. Функция управления практически перестала изменяться после шести итерации. Например, $\delta_7 = 2{,}18 \cdot 10^{-4}$.
4. Несмотря на тот факт, что начальное приближение оказалось не очень удачным и финальное положение объекта отклонилось от требуемого почти в два раза, решение замкнутой нелинейной системы практически сошлось к соответствующему решению вспомогательной линейной системы уже после 4 шагов алгоритма.

4.5.2 Управление маятником

Рассмотрим задачу демпфирования колебаний математического маятника.

$$\int_0^6 |u(t)|\,dt \longrightarrow \min_u,$$

$$\ddot{x} + \sin x = u,$$

$$-4 \leqslant u(t) \leqslant 4,$$

$$x(0) = \pi, \quad \dot{x}(0) = 1, \quad x(6) = \dot{x}(6) = 0.$$

Задача в такой постановке была рассмотрена в работе [8]. Сделаем несколько преобразований и запишем задачу в более удобном виде:

$$\int_0^6 (v_1(t) + v_2(t))\,dt \longrightarrow \min_v,$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\sin x_1 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix},$$

$$0 \leqslant v_1(t) \leqslant 4, \quad 0 \leqslant v_2(t) \leqslant 4,$$

$$x_1(0) = \pi, \quad x_2(0) = 1, \quad x_1(6) = x_2(6) = 0.$$

Вычислим Якобиан функции $f(x_1, x_2) = (x_2, -\sin x_1)^T$:

$$F(x_1, x_2) = \begin{pmatrix} 0 & 1 \\ -\cos x_1 & 0 \end{pmatrix}.$$

Очевидно, что матрица $F(x_1, x_2)$ ограничена для любых x_1 и x_2.

Для начала проанализируем результаты построения управления в программном режиме. Сначала было проведено 10 шагов алгоритма при $N = 4$. Время расчета составило 3,58 секунды. Затем построено решение для $N = 20$. На 10 итераций было затрачено 7,42 секунды. Для обоих вариантов отклонение от терминального условия стабилизировалось на четвертом знаке после запятой после 4–5 шагов алгоритма. Такое отклонение связано с заданной точностью вычисления решений дифференциальных уравнений. Повысив точность, можно добиться лучшей сходимости, но увеличится общее время расчета. Значение целевой функции для $N = 20$ оказалась примерно на 24% лучше. Подробнее с показателями для различных s можно ознакомиться в таб. 4.1.

Таблица 4.1 — Сходимость алгоритма для $N = 4$ и $N = 20$

N	s	J_s	$\left\|Hx^{(s)}(T) - g^0\right\|$	N	s	J_s	$\left\|Hx^{(s)}(T) - g^0\right\|$
4	1	8,40579	4,067779	20	1	7,39461	4,194625
4	2	4,97461	0,774187	20	2	4,20623	0,930559
4	3	4,26182	0,054745	20	3	3,22596	0,005873
4	4	4,22590	0,000955	20	4	3,22113	0,000569
4	5	4,22529	0,000424	20	5	3,22107	0,000614
4	6	4,22524	0,000376	20	5	3,22113	0,000680
4	7	4,22527	0,000393	20	7	3,22108	0,000633
4	8	4,22521	0,000321	20	8	3,22102	0,000562
4	9	4,22528	0,000362	20	9	3,22110	0,000559
4	10	4,22526	0,000387	20	10	3,22103	0,000585

На рис. 4.5 представлены фазовые траектории нелинейных и вспомогательных линейных систем при $N = 4$. Из графиков видно, что при $s = 1$ решение нелинейной системы в финальный момент времени значительно отклонилось от начала координат. Однако, при $s = 4$, траекторию нелинейной системы уже невозможно зрительно отличить от траектории соответствующей линейной системы.

Для тестирования управления в режиме реального времени проведем эксперимент: предположим, что из-за ограничения по времени возможно осуществлять лишь по одной итерации алгоритма в начале движения и в точках переключения управления. Рассчитаем траекторию движения для позиционного управления. На рис. 4.6 представлены фазовые траектории. На графике слева показаны траектории, построенные в различных точках по ходу движения объекта. Так, при $t = 0$ была рассчитана траектория $\bar{x}^{(1)}(t)$. По этой траектории объект двигался до момента $t = 1{,}5$, после чего была сформирована траектория $\bar{x}^{(2)}(t)$. В точке $t = 3$ объект перешел на траекторию $\bar{x}^{(3)}(t)$, а на отрезке $[4{,}5; 6]$ он перемещался по согласно функции $\bar{x}^{(4)}(t)$. Итоговая траектория движения объекта изображена на

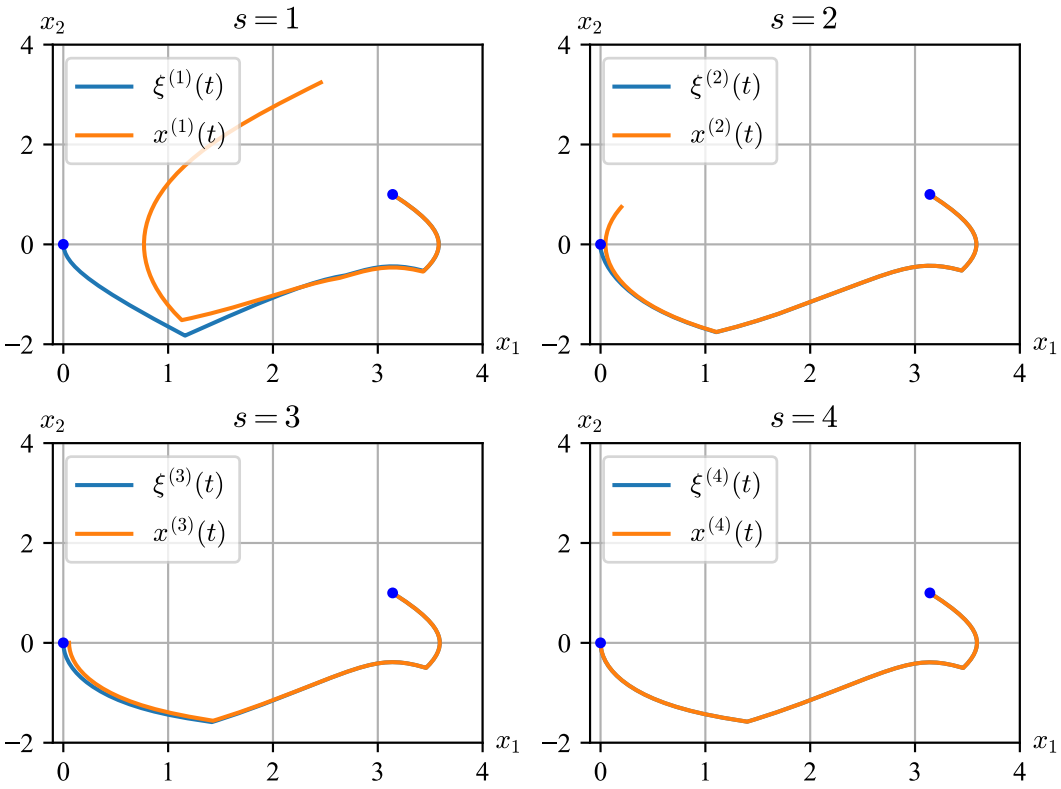

Рисунок 4.5 — Фазовые траектории для различных s при $N = 4$

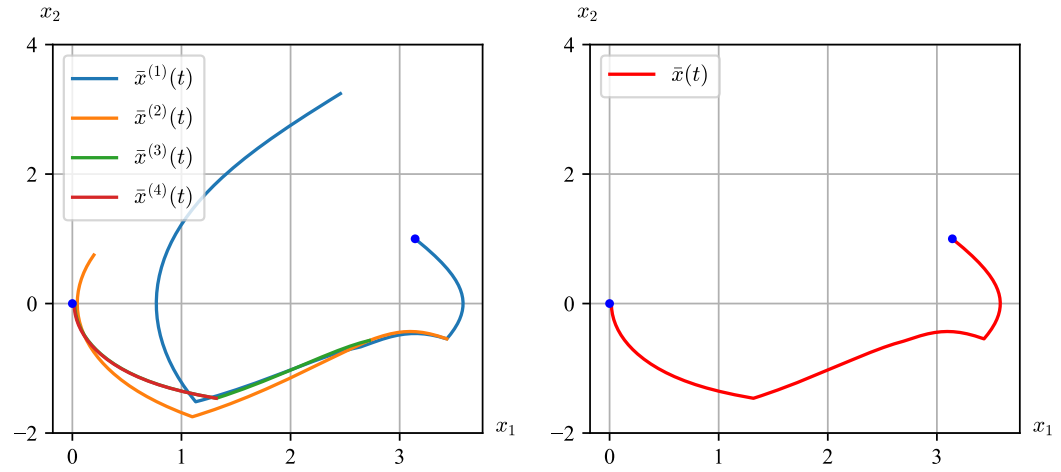

Рисунок 4.6 — Фазовые траектории для режима управления в реальном времени при $N = 4$ и $S_l = 1$

графике справа. В результате, отклонение финального положения от нуля составило 0,0163, а целевая функция приняла значение 4,4667.

По сравнению с программным режимом, отклонение $\left\|H\bar{x}(T) - g^0\right\|$ оказалось лучше, чем на первых трех шагах алгоритма, а значение ЦФ обошло результат первых двух итераций. Следовательно, может быть сделан вывод, что при «ограниченных ресурсах» (небольшие значения гиперпараметров S_l и N), решению в программном режиме удалось несильно отклониться от терминального ограничения, но сформированное позиционное управление оказалось не оптимальным.

Таблица 4.2 — Сходимость в режиме реального времени при $S_l = 1$

N	S_l	J	$\left\|H\bar{x}(T) - g^0\right\|$
4	1	4,4667	$1{,}63 \cdot 10^{-2}$
10	1	3,4952	$1{,}62 \cdot 10^{-5}$
20	1	3,2475	$8{,}07 \cdot 10^{-6}$
40	1	3,2162	$1{,}45 \cdot 10^{-6}$
60	1	3,2130	$4{,}42 \cdot 10^{-7}$
100	1	3,2093	$1{,}87 \cdot 10^{-6}$

Исследуем зависимость сходимости решений от значений гиперпараметра N. В таб. 4.2 представлены результаты работы алгоритма для различных значений N при фиксированном $S_l = 1$. Стоит отметить, что при такой конфигурации финальное положение объекта оказалось ближе к желаемому, чем при управлении в программном режиме, что можно объяснить пересчетом управления на последних участках движения в непосредственной близости от точки $(0, 0)^T$. С другой стороны, значение целевой функции оказалось несколько хуже результатов в программном режиме при больших s. Так, при $N = 4$ решение просело на 9,8%, а при $N = 20$ результат программного режима был превышен всего на 0,8%. Причина данной неоптимальности кроется в неточных аппроксимациях нелинейной функции в начале процесса движения. Тем не менее, исходя

из данного эксперимента можно предположить, что отклонение можно нивелировать увеличением точек пересчета. Наконец, можно заметить, что значение целевой функции улучшается при увеличении N вплоть до 100 точек переключения.

Таблица 4.3 — Сходимость в режиме реального времени при $N = 4$

N	S_l	J	$\left\|H\bar{x}(T) - g^0\right\|$
4	1	4,46672	$1{,}63 \cdot 10^{-2}$
4	2	4.30173	$1.38 \cdot 10^{-5}$
4	3	4.23098	$1.34 \cdot 10^{-5}$
4	4	4.22538	$1.39 \cdot 10^{-5}$
4	5	4.22534	$1.41 \cdot 10^{-5}$

Проварьируем теперь параметр S_l при $N = 4$ и запишем результаты в таб. 4.3. Естественным выводом является тот факт, что пересчет управления в режиме реального времени позволяет улучшить результаты по сравнению с управлением в программном режиме с аналогичными значениями S_l. Для $S_l = 1$ прирост составляет 47%, для $S_l = 2$ — 13,5%, далее продолжает снижаться. Начиная с $S_l = 5$ не происходит улучшения в следствии того, что аппроксимация нелинейной функции в программном режиме точна и не приводит к отклонениям. Кроме того, вычисление управления в режиме реального времени способствует уменьшению отклонения от желаемого положения объекта.

По итогам проведенного эксперимента можно сформулировать выводы для алгоритма управления в режиме реального времени в дополнение к выводам по программному, приведенным в пункте 4.5.1:

1. Управление в режиме реального времени позволяет сгладить ошибку аппроксимации нелинейной функции, возникающую при построении программного управления.
2. Если изначально было построено точное решение (с точки зрения критериев (4.10)), пересчет управления в режиме реального време-

ни не приведет к существенному улучшению. Возможен небольшой выигрыш, но, в целом, будет получено то же самое решение.

3. При наличии возможности, более правильный подход состоит в изначальном вычислении точного решения путем увеличения S_l, нежели построение неточного программного управления с дальнейшим пересчетом.
4. Увеличение точек переключения и пересчета управления N позволяет улучшить значение целевой функции и уменьшить отклонение от терминального условия, но приводит к замедлению расчета.

4.6 Выводы по главе 4

Рассмотрена задача оптимального управления, в которой система обыкновенных дифференциальных уравнений, описывающая динамику движения управляемого объекта, содержит нелинейность общего вида по фазовым переменным. Такая задача трудна для рассмотрения по двум причинам:

- Не существует аналитической формулы решения задачи Коши для системы такого вида. В следствие этого, невозможно свести исходную задачу к некой задаче математического программирования, как это было сделано в главе 1 для линейной системы.
- Отсутствуют эффективные алгоритмы построения множеств достижимости и управляемости для такой задачи. Поэтому не представляется возможным проверить существует ли допустимое решение задачи подобно алгоритму из главы 2 для линейной задачи.

Были предложены приближенные итеративные алгоритмы построения управления в программном и позиционном режимах. Основная идея этих алгоритмов заключается в линеаризации нелинейной функции линей-

ным приближением вдоль траектории движения объекта. Такой подход позволяет избежать ошибки аппроксимации при движении в окрестности заданной траектории. Алгоритм построения программного режима состоит в последовательном пересчете траектории движения объекта на основе информации о траектории, полученной на предыдущем итерации. Перестроение управления в режиме реального времени основано на тех же принципах, но происходит по ходу движения, исходя из текущего положения объекта.

В виде лемм были сформулированы и доказаны некоторые полезные свойства алгоритмов. Так, для программного режима было показано, что при стремлении функций управления, вычисляемых на разных итерациях, к общему пределу, траектория движения объекта сходится к допустимой траектории. Для позиционного режима доказано, что построенное приближенное решение сходится к допустимому решению при увеличении точек пересчета управления.

Алгоритмы были протестированы на двух примерах. Проведенные эксперименты показали хорошую сходимость алгоритмов. Даже при неудачном выборе начальной траектории, формируемые решения быстро стремились к допустимым. Также была показана сходимость целевой функции, что говорит, как минимум, о субоптимальности решения. Дополнительно протестировано поведение алгоритма построения позиционного режима при различном числе итераций алгоритма и различном количестве точек пересчета управления. На тестовых примерах продемонстрирована сходимость траектории, сформированной при управлении в позиционном режиме, к допустимой траектории при увеличении параметра N.

Перспективы дальнейшего изучения задачи могут быть связаны с дальнейшей проработкой теоретической базы алгоритмов, с рассмотрением условий сходимости управлений и критериев оптимальности.

ЗАКЛЮЧЕНИЕ

В работе представлены методы решения задачи оптимального управления в различных постановках. Общей идеей является сведение исходной задачи к некоторой задаче математического программирования. Так, базовая модель, описанная в главе 1, решается в два этапа: сначала осуществляется переход к задаче линейного (квадратичного) программирования, затем выполняется поиск оптимального решения полученной задачи. Такой подход является обоснованным, поскольку статические оптимизационные задачи с ограничениями хорошо изучены: для некоторых классов выведены алгоритмы решения с полиномиальной сложностью, существует множество программных комплексов для решения.

Подводя итоги проведенной работы, выделим основные достижения:

1. Разработаны алгоритмы нахождения программного и позиционного управления для задачи оптимального управления с нелинейной системой и кусочно-постоянным управлением, на которое наложены двусторонние ограничения. Оба алгоритма состоят в последовательном решении линеаризованной задачи. Реализован программный код алгоритмов.
2. Представлены алгоритмы построения оптимального управления в линейной задаче при выборе управления в классе кусочно-линейных и кусочно-квадратичных функций. В первом случае показано сведение к задаче линейного (квадратичного) программирования, во втором — к задаче программирования с квадратичными ограничениями. Реализован программный код решения линейной задаче с кусочно-постоянным, кусочно-линейным или кусочно-квадратичным управлением.
3. Для линейной задачи с кусочно-постоянным управлением при наличии выпуклых и невыпуклых ограничений описан алгоритм

перехода к задаче частично целочисленного линейного (квадратичного) программирования.

4. Предложен метод построения множеств достижимости и управляемости для линейной задачи с кусочно-постоянным управлением и прямыми ограничениями на него. Приведено теоретическое обоснование метода.

Предложенные алгоритмы и их реализация в виде программного кода могут быть применены для различных прикладных задач, а также для создания программного комплекса, решающего общие задачи оптимального управления.

Выделим следующие задачи в качестве направлений дальнейшего развития по данной тематике:

1. Рассмотрение более широких классов нелинейностей в правой части системы дифференциальных уравнений, в том числе: функции с неограниченной матрицей Якоби и функции с нелинейностями по управлению.
2. Доказательство оптимальности вычисляемых управлений в нелинейной задаче.
3. Построение алгоритмов нахождения управления в нелинейной задаче при выборе управления в классе кусочно-линейных и кусочно-квадратичных функций. Логичное направление, поскольку алгоритм решения состоит в последовательных линеаризациях, а для линейных моделей уже выведены методы решения для указанных классов.
4. Исследование множеств достижимости и управляемости нелинейной задачи.

СПИСОК ЛИТЕРАТУРЫ

1. Попков А. С., Баранов О. В. Об оптимальном управлении вращательным движением вала электродвигателя // Процессы управления и устойчивость. — 2014. — Т. 1(17), № 1. — С. 31–36.

2. Popkov A. S., Baranov O. V., Smirnov N. V. Application of adaptive method of linear programming for technical objects control // 2014 International Conference on Computer Technologies in Physical and Engineering Applications (ICCTPEA). — 2014. — Pp. 141–142.

3. Попков А. С. Идентификация динамической модели межотраслевого баланса для экономики России и оптимальное распределение инвестиций на ее основе // Процессы управления и устойчивость. — 2015. — Т. 2(18), № 1. — С. 696–701.

4. Popkov A. S., Smirnov N. V., Baranov O. V. Real-time quadrocopter optimal stabilization // International Conference «Stability and Control Processes» in Memory of V. I. Zubov (SCP). — 2015. — Pp. 123–125.

5. Popkov A. S. Multicriteria Regulation of Investments in the Economy of the Russian Federation // Proceedings of the International Workshop on Applications in Information Technology (IWAIT-2015), The University of Aizu Press. — 2015. — Pp. 68–70.

6. Белоусова М. В., Попков А. С. Построение динамической модели МОБ на основе WIOD // Процессы управления и устойчивость. — 2016. — Т. 3(19), № 1. — С. 601–606.

7. Popkov A. S. Application of the adaptive method for optimal stabilization of a nonlinear object // 2016 International Conference Stability and Oscillations of Nonlinear Control Systems (Pyatnitskiy's Conference). — 2016. — Pp. 1–3.

8. Popkov A. S., Smirnov N. V., Smirnova T. E. On modification of the positional optimization method for a class of nonlinear systems // ACM International Conference Proceeding Series. — 2018. — Pp. 46–51.

9. Попков А. С. Оптимальное управление в невыпуклых множествах // Устойчивость и колебания нелинейных систем управления (конференция Пятницкого) Материалы XV Международной научной конференции. — 2020. — С. 350–353.

10. Попков А. С. Оптимальное программное управление в классе квадратичных сплайнов для линейных систем // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. — 2020. — Т. 16, № 4. — С. 462–470.

11. Попков А. С. Построение множеств достижимости и управляемости в специальной линейной задаче управления // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. — 2021. — Т. 17, № 3. — С. 294–308.

12. Свидетельство о государственной регистрации программы для ЭВМ № 2021662324 Российская Федерация. "Программа для решения задачи управления квадрокоптером с использованием адаптивного метода Габасова"(AdaptCopter) : № 2021615264 : заявл. 12.04.2021 : опубл. 26.07.2021 / О. В. Баранов, А. С. Попков, Н. В. Смирнов ; заявитель федеральное государственное бюджетное образовательное учреждение высшего образования "Санкт-Петербургский государственный университет".

13. Виноградов И. М. Математическая энциклопедия. Том 3: Координаты – Одночлен. — М.: «Советская энциклопедия», 1982. — 592 с.

14. Калман Р. Е. Об общей теории систем управления // Труды I Междунар. конгресса ИФАК. — 1961. — Т. 1, № 1. — С. 521–547.

15. Понтрягин Л. С., Болтянский В. Г., Гамкрелидзе Р. В., Мищенко Е. Ф. Математическая теория оптимальных процессов. — М.: Наука, 1969. — 564 с.

16. Беллман Р., Гликсберг И., Гросс О. Некоторые вопросы математической теории процессов управления. — М.: ИЛ, 1962. — 336 с.

17. Зубов В. И. Математические методы исследования систем автоматического регулирования. — Л.: Машиностроение, 1974. — 334 с.

18. Зубов В. И. Лекции по теории управления. — М.: Наука, 1975. — 600 с.

19. Kwakernaak H., Sivan R. Linear Optimal Control Systems. — Wiley–Interscience, 1972. — 608 pp.

20. Trentelman H., Stoorvogel A., Hautus M. Control Theory for Linear Systems. — University of Groningen, Johann Bernoulli Institute for Mathematics and Computer Science, 2002. — 403 pp.

21. Хлебников М. В., Щербаков П. С., Честнов В. Н. Задача линейно–квадратичного управления: I. Новое решение // Автомат. и телемех. — 2015. — № 12. — С. 65–79.

22. Срочко В. А., Аксенюшкина Е. В., Антоник В. Г. Решение линейно–квадратичной задачи оптимального управления на основе конечномерных моделей // Известия Иркутского государственного университета. Серия Математика. — 2021. — Т. 37. — С. 3–16.

23. Матвеев А. С., Якубович В. А. Абстрактная теория оптимального управления. — СПб.: Издание С.–Петербургского университета, 1994. — 364 с.

24. Матвеев А. С., Якубович В. А. Оптимальные системы управления: обыкновенные дифференциальные уравнения. Специальные задачи: учеб. пособие для вузов. — СПб.: Изд-во СПб. гос. ун-та, 2003. — 537 с.

25. Maciejowski J. M. Predictive control with constraints: Pearson Education Limited. — Harlow, UK: Prentice–Hall, 2002. — 352 pp.

26. Пропой А. И. Применение методов линейного программирования для синтеза импульсных автоматических систем // Автомат. и телемех. — 1963. — Т. 24, № 7. — С. 912–920.

27. Rawlings J. B., Mayne D. Q. Model Predictive Control: Theory and Design. — Madison: Nob Hill Pub, 2009. — 533 pp.

28. Пономарев А. А. Аппроксимация обратной связи в регуляторе «предиктор—корректор» явной функцией // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. — 2017. — Т. 13, № 2. — С. 193–208.

29. Faulwasser T., Findeisen R. Nonlinear Model Predictive Control for Constrained Output Path Following // IEEE Transactions on Automatic Control. — 2016. — Vol. 61, no. 4. — Pp. 1026–1039.

30. Faulwasser T., Weber T., Zometa P., Findeisen R. Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot // IEEE Transactions on Control Systems Technology. — 2017. — Vol. 25. — Pp. 1505–1511.

31. Балашевич Н. В., Габасов Р., Кириллова Ф. М. Численные методы программной и позиционной оптимизации линейных систем управления //

Журнал вычислительной математики и математической физики. — 2000. — Т. 40, № 6. — С. 838–859.

32. Альсевич В. В., Габасов Р., Глушенков В. С. Оптимизация линейных экономических моделей. — Минск: Изд-во БГУ, 2000. — 210 с.

33. Канторович Л. В. Математические методы организации планирования производства. — Л.: ЛГУ, 1939. — 68 с.

34. Dantzig G. B. Linear Programming and Extensions. — Santa Monica, CA: RAND Corporation, 1963. — 641 pp.

35. Дикин И. И. Итеративное решение задач линейного и квадратичного программирования // Докл. АН СССР. — 1967. — Т. 174, № 4. — С. 747–748.

36. Евтушенко Ю. Г., Жадан В. Г. Релаксационный метод решения задач нелинейного программирования // Ж. вычисл. матем. и матем. физ. — 1977. — Т. 17, № 4. — С. 890–904.

37. Karmarkar N. A New Polynomial-Time Algorithm for Linear Programming // Combinatorica. — 1984. — Vol. 4. — Pp. 373–395.

38. Габасов Р., Кириллова Ф. М., Тятюшкин А. И. Конструктивные методы оптимизации. — Минск: Изд-во «Университетское», 1984. — Т. 1. — 216 с.

39. Хачиян Л. Г. Полиномиальный алгоритм в линейном программировании // Докл. АН СССР. — 1979. — Т. 244, № 5. — С. 1093–1096.

40. Forsgren Anders, Gill Philip E., Wong Elizabeth. Active-set methods for convex quadratic programming // arXiv: Optimization and Control. — 2015.

41. Monteiro Renato, Adler Ilan. Interior path following primal-dual algorithms: Part II: Convex quadratic programming // Mathematical Programming. — 1989. — Vol. 44. — Pp. 43–66.

42. Lobo Miguel Sousa, Vandenberghe Lieven, Boyd Stephen, Lebret Hervé. Applications of second-order cone programming // Linear Algebra and its Applications. — 1998. — Vol. 284, no. 1. — Pp. 193–228.

43. Andersen Erling D., Roos Kees, Terlaky Tamás. On implementing a primal-dual interior-point method for conic quadratic optimization // Mathematical Programming. — 2003. — Vol. 95. — Pp. 249–277.

44. Land A. H., Doig A. G. An Automatic Method of Solving Discrete Programming Problems // Econometrica. — 1960. — Vol. 28. — Pp. 497–520.

45. Cornuéjols Gérard. Revival of the Gomory cuts in the 1990's // Annals of Operations Research. — 2007. — Vol. 149, no. 1. — Pp. 63–66.

46. Marchand H., Martin A., Weismantel R., Wolsey L. Cutting planes in integer and mixed integer programming // Discrete Applied Mathematics. — 2002. — Vol. 123, no. 1. — Pp. 397–446.

47. Gilmore P. C., Gomory R. E. A Linear Programming Approach to the Cutting-Stock Problem // Operations Research. — 1961. — Vol. 9, no. 6. — Pp. 849–859. — URL: http://www.jstor.org/stable/167051.

48. Gilmore P. C., Gomory R. E. A Linear Programming Approach to the Cutting-Stock Problem – Part II // Operations Research. — 1963. — Vol. 11, no. 6. — Pp. 863–888. — URL: https://doi.org/10.1287/opre.11.6.863.

49. Desrosiers Jacques, Lübbecke Marco. A Primer in Column Generation // Column Generation. — 2006. — 03. — Pp. 1–32.

50. Dantzig George B., Wolfe Philip. Decomposition Principle for Linear Programs // Operations Research. — 1960. — Vol. 8, no. 1. — Pp. 101–111.

51. Desrosiers Jacques, Lübbecke Marco. Branch-Price-and-Cut Algorithms // Wiley Encyclopedia of Operations Research and Management Science. — 2011. — 01.

52. Rothberg E. An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions // INFORMS Journal on Computing. — 2007. — 11. — Vol. 19. — Pp. 534–541.

53. Fischetti Matteo, Lodi Andrea. Local branching // Mathematical Programming. — 2003. — 09. — Vol. 98. — Pp. 23–47.

54. Danna Emilie, Rothberg Edward, Pape Claude. Exploring relaxation induced neighborhoods to improve MILP solutions // Mathematical Programming. — 2005. — 01. — Vol. 102. — Pp. 71–90.

55. Decision Tree For Optimization Software. — URL: http://plato.asu.edu/bench.html (online; accessed: 30.06.2022).

56. Gurobi — The Fastest Solver — Gurobi. — URL: https://www.gurobi.com (online; accessed: 30.06.2022).

57. CPLEX Optimizer | IBM. — URL: https://www.ibm.com/analytics/cplex-optimizer (online; accessed: 30.06.2022).

58. Cardinal Optimizer (COPT) — Cardinal Operations. — URL: https://www.shanshu.ai/copt (online; accessed: 30.06.2022).

59. SCIP. — URL: https://www.scipopt.org/index.php (online; accessed: 30.06.2022).

60. coin-or/Cbc: COIN–OR Branch–and–Cut solver. — URL: https://github.com/coin-or/Cbc (online; accessed: 30.06.2022).

61. Corporation GAMS Development. Solver Manuals. — URL: https://www.gams.com/latest/docs/S_MAIN.html#SOLVERS_MODEL_TYPES (online; accessed: 10.08.2022).

62. Бабаджанянц Л. К., Потоцкая И. Ю. Управление по критерию расхода в механических системах. — СПб.: Санкт-Петербургский государственный университет, 2003. — 137 с.

63. Формальский А. М. Управляемость и устойчивость систем с ограниченными ресурсами. — М.: Наука, 1973. — 368 с.

64. Черноусько Ф. Л. Оценивание фазовых состояний динамических систем. Метод эллипсоидов. — М.: Наука, 1988. — 319 с.

65. Gayek J. Approximating reachable sets for a class of linear control systems // International Journal of Control. — 1986. — Vol. 43. — Pp. 441–453.

66. Baier R., Büskens C., Chahma I. A., Gerdts M. Approximation of reachable sets by direct solution methods for optimal control problems // Optimization Methods and Software. — 2007. — Vol. 22, no. 3. — Pp. 433–452.

67. Baier R., Matthias G. A computational method for non-convex reachable sets using optimal control // European Control Conference (ECC). — 2009. — Pp. 97–102.

68. Гусев М. И., Осипов И. О. Асимптотика множеств достижимости нелинейных управляемых систем на малых промежутках времени // Динамические системы: устойчивость, управление, оптимизация. Материалы Международной научной конференции памяти профессора Р. Ф. Габасова. — 2021. — С. 85–87.

69. Ekimov A. V., Balykina Yu. E., Svirkin M. V. On the estimation of the attainability set of nonlinear control systems // AIP Conference Proceedengs. — 2015. — Vol. 1648. — P. 450008.

70. Половинкин Е. С., Балашов М. В. Элементы выпуклого и сильно выпуклого анализа. — М.: ФИЗМАТЛИТ, 2004. — 416 с.

71. Dantzig G. B., Eaves B. C. Fourier-Motzkin elimination and its dual // J. Combinatorial Theory Ser. A. — 1973. — Vol. 14. — Pp. 288–297.

72. Park J. J., Kuipers B. A smooth control law for graceful motion of differential wheeled mobile robots in 2D environment // IEEE International Conference on Robotics and Automation. — 2011. — Pp. 4896–4902.

73. Балашевич Н. В., Габасов Р., Кириллова Ф. М. Алгоритмы программной и позиционной оптимизации систем управления с промежуточными фазовыми ограничениями // Журнал вычислительной математики и математической физики. — 2001. — Т. 41, № 10. — С. 1485–1504.

74. Жабко А. П., Чижова О. Н., Котина Е. Д. Дифференциальные уравнения и устойчивость / Под ред. Н. В. Черезова. — СПб.: Лань, 2015. — 320 с.

75. Балашевич Н. В., Габасов Р., Кириллова Ф. М. Вычисление оптимальных программы и управления в линейной задаче с фазовым ограничением // Журнал вычислительной математики и математической физики. — 2005. — Т. 45, № 12. — С. 2112–2130.

76. Sargent R., Sullivan G. R. The Formulation of Optimal Control Problems as Nonlinear Programmes. — 1977.

77. Bock H. G., Plitt K. J. A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems // IFAC Proceedings Volumes. — 1984. — Vol. 17, no. 2. — Pp. 1603–1608.

78. Hargraves C., Paris S. Direct Trajectory Optimization Using Nonlinear Programming and Collocation // AIAA J. Guidance. — 1987. — Vol. 10. — Pp. 338–342.

79. Garg D., Patterson M., Hager W. et al. An overview of three pseudospectral methods for the numerical solution of optimal control problems. — 2017.

80. Габасов Р., Кириллова Ф. М., Ружицкая Е. А. Демпфирование и стабилизация маятника при больших начальных возмущениях // Изв. РАН. Теория и системы управл. — 2001. — № 1. — С. 29–38.

81. Балашевич Н. В., Габасов Р., Калинин А. И., Кириллова Ф. М. Оптимальное управление нелинейными системами // Журнал вычислительной математики и математической физики. — 2002. — Т. 42, № 7. — С. 969–995.

82. Згонников А. В. Синтез оптимальной обратной связи в одной нелинейной механической системе // Процессы управления и устойчивость: Труды 40-й международной научной конференции аспирантов и студентов. — 2009. — С. 21–26.

83. Габасов Р., Кириллова Ф. М., Альсевич В. В. и др. Методы оптимизации: пособие. — Минск: Издательство «Четыре четверти», 2011. — 472 с.

84. Зубов В. И. Синтез многопрограммных устойчивых управлений // Докл. АН СССР. — 1991. — Т. 318, № 2. — С. 274–277.

85. Смирнов Н. В., Соловьева И. В. Применение метода позиционной оптимизации для многопрограммной стабилизации билинейных систем // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. — 2009. — № 3. — С. 251–259.

86. Smirnov N. V. Multiprogram Control for Dynamic Systems: A Point of View // Proceedings of the 2012 Joint International Conference on Human–Centered Computer Environments. — HCCE 12. — New York, NY, USA: Association for Computing Machinery, 2012. — Pp. 106–113. — URL: https://proxy.library.spbu.ru:2060/10.1145/2160749.2160773.

87. Andersson J. A. E., Gillis J., Horn G. et al. CasADi — A software framework for nonlinear optimization and optimal control // Mathematical Programming Computation. — 2019. — Vol. 11, no. 1. — Pp. 1–36.

88. Houska B., Ferreau H. J., Diehl M. ACADO Toolkit — An Open Source Framework for Automatic Control and Dynamic Optimization // Optimal Control Applications and Methods. — 2011. — Vol. 32, no. 3. — Pp. 298–312.

89. DIDO — Optimal control software — Third-Party Products & Services — MATLAB & Simulink. — URL: https://www.mathworks.com/products/connections/product_detail/dido.html (online; accessed: 09.08.2022).

90. Лизоркин П. И. Курс дифференциальных и интегральных уравнений с дополнительными главами анализа. — М.: Наука, 1981. — 384 с.

91. Репозиторий GitHub — piecewise_control: Algorithm of constructing optimal solution for linear optimal control problem in the class of piecewise constant, piecewise linear or piecewise quadratic functions. — URL: https://github.com/alexander-popkov/piecewise_control (дата обращения: 11.08.2022).

92. Репозиторий GitHub — nonlinear_control: Algorithms for construct program and positional solutions in nonlinear optimal control problem with piecewise constant control. — URL: https://github.com/alexander-popkov/nonlinear_control (дата обращения: 11.08.2022).

ПРИЛОЖЕНИЕ А

Программный код решения линейной задачи оптимального управления в различных классах управления

В данном приложении представлено описание структуры программного кода алгоритма решения линейной задачи оптимального управления в классе кусочно-постоянных, кусочно-линейных или кусочно-квадратичных функций. Код выложен в публичный репозиторий [91].

Программный комплекс состоит из следующих скриптов:

- *run.py*: функция запуска алгоритма, в ней осуществляется выбор класса управления.
- *problem/ProblemStatement.py:* постановка задачи оптимального управления с линейной системой, задание параметров модели.
- *control/PiecewiseConstantControl.py:* построение оптимального кусочно-постоянного управления. Исходная задача сводится к задаче линейного или квадратичного программирования. Затем вычисляется решение полученной задачи с помощью солвера COPT (библиотека *coptpy*).
- *control/PiecewiseLinearControl.py:* построение оптимального кусочно-линейного управления. Исходная задача сводится к задаче линейного или квадратичного программирования. Затем вычисляется решение полученной задачи с помощью солвера COPT (библиотека *coptpy*).
- *control/QuadraticSplineControl.py:* построение оптимального кусочно-квадратичного управления. Исходная задача сводится к выпуклой задаче программирования с квадратичными ограничениями. Затем вычисляется решение полученной задачи с помощью солвера COPT (библиотека *coptpy*).

- *movement/PCCObjectMovement.py:* вычисление траектории движения объекта как решения системы дифференциальных уравнений, замкнутой кусочно-постоянным управлением.
- *movement/PLCObjectMovement.py:* вычисление траектории движения объекта как решения системы дифференциальных уравнений, замкнутой кусочно-линейным управлением.
- *movement/QSCObjectMovement.py:* вычисление траектории движения объекта как решения системы дифференциальных уравнений, замкнутой кусочно-квадратичным управлением.
- *numerical/DefiniteIntegral.py:* решение задачи Коши с линейной однородной системой с использованием функции *odeint* из пакета *scipy.integrate*.
- *numerical/DifferentialEquation.py:* вычисление определенного интеграла с помощью функции *quad* из пакета *scipy.integrate*.
- *numerical/Interpolation.py:* интерполяция таблично заданной функции кубическими сплайнами с помощью функции *interp1d* из пакета *scipy.interpolate*.
- *plotting/Plotting.py:* функции построения графиков. Используются средства библиотеки *matplotlib.pyplot*.

ПРИЛОЖЕНИЕ Б

Программный код решения нелинейной задачи оптимального управления в программном и позиционном режимах

В данном приложении представлено описание структуры программного кода алгоритма решения нелинейной задачи оптимального управления в классе кусочно-постоянных функций в программном и позиционном режимах. Код доступен в публичном репозитории [92].

Программный комплекс состоит из следующих скриптов:

- *run_program_control.py:* функция запуска алгоритма расчета программного управления для нелинейной задачи. Здесь задается число итераций алгоритма S_l.
- *run_positional_control.py:* функция запуска алгоритма расчета позиционного управления для нелинейной задачи. Определяется число итераций алгоритма S_l.
- *task/Task.py:* класс, в котором осуществляется задание нелинейной задачи оптимального управления и основных параметров модели.
- *task/nonlinear_scalar_task.py:* реализация класса *Task* в виде нелинейной модели, представленной в пункте 4.5.1.
- *task/pendulum_control.py:* реализация класса *Task* в виде модели маятники, приведенной в пункте 4.5.2.
- *problems/NonLinearProblem.py:* класс для работы с нелинейной моделью. Здесь реализованы функции линеаризации нелинейной правой части и расчета траектории движения объекта как решения задачи Коши с нелинейной системой, замкнутой найденным управлением.
- *problems/LinearProblem.py:* класс для работы с линейной моделью. Здесь осуществляется сведение к задаче линейного или квадратич-

ного программирования. Для этого реализованы функции решения системы дифференциальных уравнений (*scipy.integrate.odeint*), вычисления определенного интеграла (*scipy.integrate.quad*) и построения матрицанта.

- *lin_prog_task/LPTask.py:* класс, в котором хранятся параметры интервальной задачи линейного или квадратичного программирования.
- *lin_prog_model/LPModel.py:* класс, работающий с задачей линейного (квадратичного) программирования. С помощью библиотеки *coptpy* инициализируется модель, задаются переменные, ограничения и целевая функция. Затем средствами солвера COPT осуществляется поиск оптимального решения.
- *common/common.py:* ряд вспомогательных функций.
- *plotting/plotting.py:* построение графиков для демонстрации работы алгоритма с использованием пакета *matplotlib.pyplot*.

Saint-Petersburg State University

Alexandr S. Popkov

Optimal positional control in nonlinear control systems

Scientific specialisation 2.3.1.
System analysis, control and information processing, statistics

Dissertation is submitted for the degree of
candidate of physical and mathematical sciences

Translation from Russian

Scientific supervisor:
Doctor of Physical and Mathematical Sciences, Professor
Nikolay V. Smirnov

Saint-Petersburg — 2022

Table of Contents

INTRODUCTION

Relevance of the topic. In this dissertation we consider the problem of constructing an optimal control, which transfers the object from the initial state to some plane. The search for controls is performed in a class of piecewise constant functions. A system of ordinary differential equations is used to describe the dynamic process. The quality of control is determined by the value of linear or quadratic objective functional.

Before the dynamic process begins, the control function (program control) is determined and the trajectory of the object is calculated. Due to various internal and external factors, the object may deviate from the original trajectory during the motion. Then there is a sense of real-time realignment of control based on the current position. Such control is called positional control.

Such problems are widespread in various fields. For example, the author of this paper tested the presented algorithms on the following models:

– Rotary motion control of a motor shaft.
– Moving an unmanned cart.
– Distribution of investments in sectors of the economy.
– Quadcopter motion stabilization.
– Damping of an artificial satellite.
– Control of a mathematical pendulum.
– Damping of a sleeping Lagrange top.

Many control processes of technical objects are objectively non-linear in nature, therefore, for their accurate modelling and, subsequently, control it is necessary to have a control construction apparatus for non-linear systems.

In the theory of ordinary differential equations the analytical solution of the Cauchy problem for linear systems of any dimensionality is known: the Cauchy formula is derived. This led to the development of methods for solving linear control problems. However, analytical solutions for non-linear systems of

differential equations exist only for very specific kinds of non-linearities, which makes it impossible to invent exact methods of finding control for systems with non-linear right-hand sides. This paper develops a method of numerical construction of control for non-linear systems in terms of phase variables by means of iterative linearization of the right part of the system and finding the optimal control for the linear system.

The relevance of the problem is also demonstrated by the publication activity on this topic in both national and foreign journals in recent decades. In addition, software packages for such tasks are being actively developed and improved.

Aim and objectives. The main aim of the work is to develop a numerical method for finding an approximate solution in a non-linear optimal control problem. A non-linear problem is understood as a control in a system of ordinary differential equations containing non-linearities with respect to phase variables. It is necessary to construct algorithms for the construction of program and positional controls. The program control is calculated before the beginning of motion and then it is recalculated based on the current position taking into account internal and external disturbances. It is assumed that the control is corrected after a given time interval since the last correction, while the motion of the controlled object does not stop, which makes it possible to speak about real-time control.

To achieve the goal it is necessary to solve a number of tasks:

1. Study modern scientific developments in this direction, identify promising directions.
2. Research the reachability set and the controllability set in a linear control problem when choosing a control in a class of piecewise constant functions. Information about these sets is important when solving a non-linear problem, because it is one way or another reduced to a linear one.

3. Develop a method for solving the problem with linear system and choice of control in the class of piecewise linear and piecewise quadratic functions. Consideration of such types of control makes it possible to find solutions with the best value of the objective function and to impose smoothness constraints on the sought control.
4. Develop an approach to solving a linear problem with additional convex and non-convex constraints on controls and phase variables. Setting constraints of this type allows us to consider more specific classes of problems.
5. Develop a numerical algorithm for constructing program and positional controls in a non-linear problem with control selection in a class of piecewise constant functions.
6. Explore the theoretical properties of algorithms.
7. Implement the algorithm for constructing controls in a non-linear problem in the form of a set of functions in Python. Test the algorithm on various examples and scenarios.

Thesis statements to be defended

1. A method for constructing sets of reachability and controllability for a linear problem in the class of piecewise constant controls in the presence of two-sided constraints, a rigorous theoretical justification of the method.
2. Algorithms for reducing the optimal control problem to a linear programming problem for piecewise linear and piecewise quadratic classes of controls.
3. Algorithms for reducing the problem of optimal control with piecewise constant control and linear (quadratic) functional with additional convex and non-convex constraints to the problem of mixed integer linear (quadratic) programming.
4. Algorithms for the construction of program and positional controls for a non-linear optimal control problem with piecewise constant control.

5. Software implementation of the presented algorithms in the form of a set of programs.

Methodology and methods of the research. The main idea of solving optimal control problems in various formulations is to reduce them to mathematical programming problems. After forming the methods of control construction, the theoretical properties of solutions are studied and algorithms are tested on test examples. Methods of linear algebra, mathematical analysis, differential equations, control theory, linear and quadratic programming are used.

Theoretical and practical significance of the work consists in the proposed constructive algorithms for solving the stated problems. Methods of constructing control for a non-linear problem in program and positional modes are presented, which are tested on a set of examples and can be used to control specific devices.

The initial problem can be extended by considering additional classes of control functions and additional constraints on controls and phase variables. Consideration of the problem in such formulations makes practical sense, since often application models contain specific constraints or special requirements to the solution.

Also practical significance represents the algorithm for constructing the sets of reachability and controllability. In various applied problems information about all possible future states of the object is considered important. The proposed algorithm is constructive and can be easily implemented for an applied problem. In addition, the proof that such an algorithm allows to form exact sets has its own significant scientific value.

Finally, the implemented algorithms are accompanied by the program code, which can be used as a basis for creating import-independent software products to solve the problems of optimal control. This task seems important and relevant from a practical point of view.

Scientific novelty

1. An algorithm for control construction in program and positional modes for the optimal control problem with a non-linear system, terminal conditions and control constraints has been developed. The idea of the algorithm is the consecutive linearization of the non-linear function along the motion trajectory and the subsequent calculation of a new trajectory with the control, which is the optimal solution of the linearized problem.
2. For a linear problem in which the control is chosen in the class of piecewise constant functions and has two-sided constraints, a method for constructing the sets of reachability and controllability in the form of a system of linear algebraic inequalities is proposed. It is shown that these sets are polyhedrons in the space of phase variables, the theoretical justification of the algorithm is given.
3. A method for finding the optimal control for a linear problem with piecewise quadratic control, terminal conditions and direct constraints on controls is developed. The key feature of the method consists in reducing the original problem to a convex programming problem with quadratic constraints. Algorithms for solving problems of this class have much less complexity than algorithms for non-convex problems.
4. An algorithm for constructing an optimal control for a linear problem with piecewise constant control in the presence of convex and non-convex constraints on phase variables (at given time moments) and on controls is proposed. Non-convex constraints are defined as a set of groups of inequalities, at each time moment the constraints of at least one group must be satisfied. The ability to reduce the problem in such formulation to a mixed integer linear or quadratic programming problem is shown.

Reliability is ensured by the correctness of problem statements obtained from the literature and during seminars at the Faculty of Applied Mathematics

— Control Processes of St Petersburg State University. The presented results have been tested at many conferences, and the publications that served as the basis for the thesis have been peer-reviewed and published in Russian and international journals. All proposed algorithms have been programmed and tested on various tasks.

Personal contribution. The thesis is an independent work of the author. The main statements, which are to be defended, represent the author's personal contribution. All results presented in the dissertation were achieved by the author of the work, except for those places where it is explicitly stated and reference to the primary source is indicated. Most of the results are based on the author's publications in scientific journals. The program code presented in the appendix was implemented by the author.

Approbation of the work. The results presented in this dissertation have been presented and discussed at conferences:

1. XLV International Scientific Conference on Control Processes and Stability (CPS'14), April 1–4, 2014, St Petersburg, Russia.
2. International Conference on Computer Technology in Physical and Engineering Applications (ICCTPEA–2014), June 30 – July 4, 2014, St Petersburg, Russia.
3. XLVI International Scientific Conference on Control Processes and Stability (CPS'15), April 6–9, 2015, St Petersburg, Russia.
4. III International Conference Dedicated to the Memory of Professor Vladimir Zubov, October 5–9, 2015, St Petersburg, Russia.
5. International Workshop on Applications in Information Technology (IWAIT–2018), October 8–10, 2015, Aizu-Wakamatsu, Japan.
6. XLVII International Scientific Conference on Control Processes and Stability (CPS'16), April 4–7, 2016, St Petersburg, Russia.
7. XIII International Conference «Stability and Oscillations of Nonlinear Control Systems» (Pyatnitskiy's Conference), June 1–3, 2016, Moscow, Russia.

8. 3rd International Conference on Applications in Information Technology (ICAIT–2018), November 1–3, 2018, Aizu-Wakamatsu, Japan.
9. IV International Conference Dedicated to the Memory of Professor Vladimir Zubov, October 5–9, 2020, St Petersburg, Russia.

Publications. The results of the thesis are presented in eleven scientific publications [1–11], of which two articles were published in journals included in the list of editions of the Higher Attestation Commission [10; 11], and seven works — in the journals indexed in the Scopus and Web of Science databases [2; 4; 7; 8; 10; 11]. A computer program «AdaptCopter» [12] has also been registered.

Support. This work was supported by the Russian Foundation for Basic Research (project N 19-31-90033).

Contents and structure of the work. The thesis consists of an introduction, four chapters, conclusion, and two appendices. The total volume of the dissertation is 141 pages, including 11 figures and 4 tables. The reference list contains 92 titles.

Overview of the dissertation. In the **introduction** there is a brief statement of the problem, the relevance is justified, the aim and objectives of the thesis are formed, the main provisions of the thesis are presented, the scientific and practical value and scientific novelty of the work is described. Next, the information about the approbation of the work, the main publications and financial support is given, and a brief summary of the work is presented.

In the **first chapter** the optimal control problem with a linear system is considered. The terminal control problem consists in transferring the controlled object from the initial position to some plane for a given time. The control is chosen from a class of piecewise constant functions with a fixed sampling period. Three objective criteria, which determine the quality of the solution of the problem, are set. The first functional is linear, the second can be reduced

to linear, and the third is quadratic. For each specific problem one of these functionals or their linear combination is chosen.

A method for solving a linear problem is described, which consists in the transition to a linear or quadratic programming problem with subsequent solution of the resulting optimization problem. This chapter also gives an overview of methods for solving mathematical programming problems and a comparative analysis of software for solving such problems.

Although the model in question has been well studied in the literature, Chapter 1 is important in the overall structure of the paper, since the results obtained are the basis on which the following chapters are based.

In the **second chapter** the question of constructing reachability and controllability sets for the linear model from Chapter 1 is researched. It is shown that the problem of constructing these sets is reduced to the problem of linear mapping of a multidimensional cube from one space to another. As a result of research of properties of the sets it is proved that they are convex polyhedrons and can be described by a system of linear inequalities. The main result of the chapter is a constructive algorithm for constructing sets and its rigorous proof. An analysis of the complexity of the algorithm is carried out.

The **third chapter** considers a linear problem with additional conditions and constraints. The first part of the chapter studies the problem of finding the optimal solution when choosing a control in alternative classes of functions. It is shown that a linear problem with piecewise linear control and a linear (quadratic) functional reduces to a linear (quadratic) programming problem. In the case of piecewise quadratic control the transition to a convex programming problem with quadratic constraints is demonstrated.

In the second part of the chapter, the basic linear model is supplemented by convex and non-convex linear constraints on the phase variables in the nodal points and on controls. For convex constraints there is an algorithm for reduction to a linear (quadratic) programming problem, and for a model with

non-convex constraints — to a mixed integer linear (quadratic) programming problem.

The **fourth chapter** presents an algorithm for constructing control in program and positional modes for a non-linear optimal control problem. The algorithm for computing the program control consists of an iterative recalculation of the control based on the trajectory of the object found at the previous step. A linear approximation of the non-linear function along this trajectory is found, then the updated control is determined as the optimal solution of the linearized problem. After that, the solution of the Cauchy problem of the original system closed by the found control is calculated. The obtained solution is the new trajectory of the object.

The following describes the algorithm for constructing the positional control. At the node points, the control is recalculated based on the current position or the predicted position of the object. The program mode algorithm is used for recalculation, but additional attention is paid to the calculation time.

The theoretical properties of the algorithms have been studied, and the conditions of convergence to an admissible solution of the original problem have been derived. The work of the algorithms is demonstrated on two test problems.

In the **conclusion** the achieved results are listed and an assessment of further development prospects is given.

Appendix A describes the structure of the software package for solving a linear optimal control problem for the choice of control in a class of piecewise constant, piecewise linear, or piecewise quadratic functions.

Appendix B describes the structure of the software package for solving a non-linear problem with piecewise constant control in the program and positional modes.

CHAPTER 1. LINEAR OPTIMAL CONTROL PROBLEM

The first chapter is introductory. It considers a linear optimal control problem and gives the necessary terminology. In Paragraph 1.1, the formulation of the problem in a generalized form is given. In Paragraph 1.2, a review of the literature on this topic is conducted, the place of the current work is indicated. Paragraph 1.3 describes the algorithm for reducing the original problem to a linear programming problem. In Paragraph 1.4, considerations about methods for solving linear programming problems and the use of software packages for these purposes are given.

1.1 Problem Formulation

1.1.1 Object Dynamics

Consider a linear control problem. Let's assume that the motion of a controlled object is described by a system of ordinary differential equations:

$$\dot{x} = A(t)x + B(t)u + d(t). \tag{1.1}$$

In this expression

- t is an independent variable (time) that takes values on a segment from 0 to T, where T is a hyperparameter of the model.
- $x = x(t)$ is an n-dimensional vector-function of phase variables. The components $x_j(t)$ are continuous at $t \in [0, T]$.
- $\dot{x}$ is a vector of derivatives of phase variables with respect to time. Here and below the derivative is understood as the left derivative.

- $u = u(t)$ is an r-dimensional vector-function of controls. The components $u_i(t)$ are left continuous.
- $A(t)$ is a matrix function of t of dimension $n \times n$.
- $B(t)$ is a matrix function of t of dimension $n \times r$, composed of columns $b_i(t)$.
- $d(t)$ is an n-dimensional vector-function.

In addition, the functions $A(t)$, $B(t)$, and $d(t)$ are assumed to be continuous on the segment $[0,\ T]$.

By replacing variables, we can achieve the exclusion of the vector $d(t)$ from the system. However, its presence is important for future calculations. Particularly $d(t)$ will be used in the linearization of a non-linear function.

1.1.2 Terminal Conditions

The goal of control is to transfer the object from a given initial position to some linear manifold in the predetermined time T:

$$x(0) = x_*, \quad Hx(T) = g^0, \tag{1.2}$$

where H is a matrix of full rank and has dimension $m \times n$ $(m \leqslant n)$, g^0 is an m-dimensional vector.

An important special case is when $m = n$. Then the problem of transferring the object to a given final position $x(T) = H^{-1}g^0$ is considered. Another special case is when $m = 0$. Under this condition, the trajectory of the controlled object has a free right end.

1.1.3 Control Constraints

Direct constraints are imposed on the components of the control vector:

$$l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1,r}. \tag{1.3}$$

Consequently, in the basic formulation of the problem controls are selected from the r-dimensional rectangle.

We will choose the controls $u_i(t)$ in the class of piecewise constant functions with a given sampling period $h = T/N$:

$$u_i(t) = \begin{cases} u_{ik}, \quad t \in [t_{k-1},\ t_k), \\ k = \overline{1,N}, \end{cases} \qquad i = \overline{1,r}. \tag{1.4}$$

where $t_k = kh$.

1.1.4 Objective Function

This paper considers three objective functions: two linear and one quadratic.

Final System Position

$$J_1 = c^T x(T) \longrightarrow \min. \tag{1.5}$$

The first objective function assumes minimization of the linear combination of the phase variables at a finite point in time. Thus, it is a condition on the final position of the system. Here c is some n-dimensional vector. Problem (1.1)–(1.5) is the Mayer problem [13]. Note that it makes sense to consider such an objective function only when the right end is not fixed, that is, when $m < n$, and accordingly the matrix H in condition (1.2) is rectangular.

The simplest example of such a functional is the criterion $x_j(T) \to \max$, which requires maximizing the final position of one of the phase vector components.

Resource Consumption

$$J_2 = \int_0^T \|u(t)\|_1 \, dt = \int_0^T \sum_{i=1}^{r} |u(t)| \, dt \longrightarrow \min . \tag{1.6}$$

The second objective function is an integral of the norm of the control vector. In this case we will consider «Manhattan distance» as a norm, because it is the most convenient for reduction to linear optimization problems.

Quadratic Functional

$$J_3 = \int_0^T u(t)^T Q u(t) \, dt \longrightarrow \min . \tag{1.7}$$

The third objective function is an integral of the quadratic form from the vector of controls. Here Q is a symmetric matrix of dimension $r \times r$, its elements will be denoted by the symbol $q_{i_1 i_2}$. We will also assume that Q is a non-negative definite matrix. Problem (1.1)–(1.4), (1.7) is the Lagrange problem [13].

An example of use of such objective function is minimization of integral from the square of control norm: $\int_0^T \|u(t)\|_2^2 \, dt \to \min$. The physical meaning of such criterion is minimization of use of control actions (minimization of energy consumption) similar to the second objective function, but in another normalized space.

1.2 Literature Review

One of the founders of the mathematical theory of control is considered to be Rudolf Kalman, who made a significant contribution to the development of this field of knowledge [14].

The classical method for solving the problem of optimal control is the Pontryagin's maximum principle [15]. It was formulated in 1958 by a team of scientists under the guidance of L. C. Pontryagin in the form of necessary conditions for optimal solution. The method is effective for many particular cases, but it is not universal due to the necessity of solving non-linear algebraic equations.

An alternative to the maximum principle is the method of dynamic programming. It is based on the Bellman principle of optimality [16], which guarantees that sufficient conditions of optimality are met.

A significant contribution to the development of the theory of motion stability, the theory of automatic control, and the theory of optimal processes was made by V. I. Zubov [17]. He proposed a method of successive approximations for finding of optimal program motions and for solving the problem of optimal control synthesis [18]. Separate results were obtained for the problem with a linear system and a quadratic functional, an optimal stabilizing control is found as a solution of the matrix Riccati equation by the method of successive approximations.

Also the problem of constructing a stabilizing control for a linear-quadratic problem is covered in many foreign sources [19; 20]. Among modern works, we can highlight the approach using matrix inequalities to solve the convex linear-quadratic problem [21] and the algorithm for constructing piecewise linear control for the non-convex problem [22].

The abstract theory of optimal control, the founder of which is V. A. Yakubovich [23; 24], can be singled out as a separate direction in the research of the optimal control problem. He managed to formulate optimality conditions for various classes of problems.

A relevant vector of development of approximate methods for the problem of optimal control is the model predictive control (MPC) [25], the idea of which is to construct a control for some horizon of time ahead. In the 1960s and 70s, the algorithm for linear systems [26] was developed, which consists in reducing

it to a linear programming problem. Modern works are devoted to non-linear problems [27; 28] and industrial applications [29; 30].

The basis for this study was the work of a team of scientists under the leadership of R. F. Gabasov. They proposed a two-stage algorithm for finding the optimal control for linear systems: reduction of the control problem to an interval linear programming (ILP) problem [31], with a subsequent solution of this problem by a specially developed adaptive method [32]. A method for solving the non-linear optimal control problem was presented by the team and will be described in Paragraph 4.2.

Gabasov's method for the linear case was tested by the author of this paper for the problem of controlling the rotational motion of the electric motor shaft [1], controlling the motion of a four-wheeled cart [2], and allocating investments in industries of a multi-product economy [3]. An optimal control was constructed for the linearized quadcopter model in [4].

1.3 Reduction of Optimal Control Problem to Interval Linear Programming Problem

1.3.1 Terminal Conditions Reduction

The constraint on the left end of the trajectory will be automatically fulfilled as the initial condition of the Cauchy problem. In order to express the final state of the object as a function of the controls, let's write out the Cauchy formula for the solution of system (1.1) with the initial condition $x(0) = x_*$ at the point T:

$$x(T) = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)\left(B(t)u(t) + d(t)\right)dt.$$

Hereinafter $Y(t)$ is a state-transition matrix of the homogeneous system of differential equations with the matrix $A(t)$ normalized at the point 0:

$$\dot{Y}(t) = A(t)Y(t), \quad Y(0) = E.$$

Let's replace the control by its representation as a piecewise constant function (1.4):

$$x(T) = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)d(t)dt + \sum_{i=1}^{r}\sum_{k=1}^{N}\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)u_{ik}dt.$$

We can take the control values u_{ik} on the time segments beyond the integral, since these values do not depend on time:

$$x(T) = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)d(t)dt + \sum_{i=1}^{r}\sum_{k=1}^{N}\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dtu_{ik}.$$

Section 3.1.3 will address the issue of calculating integrals of the form

$$\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt,$$

but in the meantime, note that this integral does not contain the variables we are looking for, so its value is a constant, which we denote by $b^{i,k}$:

$$b^{i,k} = \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i dt, \quad i = \overline{1,r}, \quad k = \overline{1,N}.$$

The remaining items in the right-hand side of the Cauchy formula also do not depend on u_{ik}. Denote their sum by the symbol d^0:

$$d^0 = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)d(t)dt,$$

and the question of calculating d^0 will be covered later.

Thus, we can write out a representation of the final state of the object $x(T)$ as a linear function of the variables u_{ik}:

$$x(T) = d^0 + \sum_{i=1}^{r}\sum_{k=1}^{N} b^{i,k}u_{ik}. \tag{1.8}$$

Then boundary condition (1.2) appears as a system of linear constraints:

$$\sum_{i=1}^{r}\sum_{k=1}^{N} Hb^{i,k}u_{ik} = g^0 - Hd^0.$$

After introducing the new notations we get:

$$\sum_{i=1}^{r}\sum_{k=1}^{N} h_l^{i,k}u_{ik} = g_l, \quad l = \overline{1,m}. \tag{1.9}$$

Here $h^{i,k} = Hb^{i,k}$, and $g = g^0 - Hd^0$. As a result, terminal conditions (1.2) have turned into m linear equations with rN variables (1.9).

1.3.2 Reduction of Direct Constraints on Controls

For this class of controls, the reduction is trivial. Direct constraints (1.3) will evidently turn into two-sided constraints on the values of the controls on the time segments:

$$l_{*i} \leqslant u_{ik} \leqslant l_i^*, \quad i = \overline{1,r}, \quad k = \overline{1,N}. \tag{1.10}$$

1.3.3 Objective Functions Reduction

To reduce the **first objective function** (1.5), we use found representation of the final state of the object (1.8). Then we can write criterion (1.5) as a linear function of the controls:

$$J_1 = c^T d^0 + \sum_{i=1}^{r}\sum_{k=1}^{N} c^T b^{i,k}u_{ik} \longrightarrow \min\,.$$

Since the first item is a constant, we can remove it from the objective function, keeping in mind that this will change the value of the functional:

$$J_1 = \sum_{i=1}^{r}\sum_{k=1}^{N} c_{ik}u_{ik} \longrightarrow \min, \tag{1.11}$$

where $c_{ik} = c^T b^{i,k}$. Thus, (1.11) is a linear functional with respect to the variables u_{ik}.

In the case of the **second objective function** (1.6) we introduce two types of additional variables into the model to convert the sum of modules into a linear function:

$$\begin{array}{l} u_{ik} = u_{ik}^{+} - u_{ik}^{-}, \\ u_{ik}^{+} \geqslant 0, u_{ik}^{-} \geqslant 0, \end{array} \quad k = \overline{1,N}, \quad i = \overline{1,r}. \tag{1.12}$$

Then functional (1.6) will change to the sum of the new variables:

$$J_2 = h \sum_{i=1}^{r} \sum_{k=1}^{N} \left(u_{ik}^{+} + u_{ik}^{-} \right) \longrightarrow \min . \tag{1.13}$$

Based on the type of objective function (1.13), there is the following relationship between the variables:

$$\begin{cases} \text{if } u_{ik} \geqslant 0, \ \text{ then } u_{ik} = u_{ik}^{+}, \\ \text{if } u_{ik} \leqslant 0, \ \text{ then } u_{ik} = -u_{ik}^{-}. \end{cases}$$

Thus, we can say that u_{ik}^{+} denotes the positive part of the number u_{ik}, and u_{ik}^{-} denotes the negative part.

Next, we proceed to the **third criterion**. Let's replace in (1.7) the control by its representation as piecewise constant function (1.4):

$$J_2 = \int_0^T u(t)^T Q u(t) dt = \sum_{k=1}^{N} \int_{t_{k-1}}^{t_k} \left(\sum_{i_1=1}^{r} \sum_{i_2=1}^{r} u_{i_1 k} q_{i_1 i_2} u_{i_2 k} \right) dt.$$

Then, given that $t_k - t_{k-1} = h$, we can write out the final form of the expression:

$$J_2 = h \sum_{i_1=1}^{r} \sum_{i_2=1}^{r} q_{i_1 i_2} \sum_{k=1}^{N} u_{i_1 k} u_{i_2 k} \longrightarrow \min . \tag{1.14}$$

We got functional (1.14) as a quadratic form with respect to the variables u_{ik}. In Section 3.1.2 we will show that this is a non-negative definite quadratic form.

1.3.4 Reduction Results

As a result, linear optimal control problem (1.1)–(1.5) was reduced to problem (1.9)–(1.11):

$$\sum_{i=1}^{r}\sum_{k=1}^{N} c_{ik}u_{ik} \longrightarrow \min,$$

$$\sum_{i=1}^{r}\sum_{k=1}^{N} h_l^{i,k}u_{ik} = g_l, \quad l = \overline{1,m},$$

$$l_{*i} \leqslant u_{ik} \leqslant l_i^*, \quad i = \overline{1,r}, \quad k = \overline{1,N},$$

with the following characteristics:

– rN variables,

– m linear equations,

– $2rN$ linear inequalities,

– a linear objective function.

Since two-sided constraints are given for all variables, this problem can be classified as an interval linear programming problem, which, in addition to classical methods, can also be effectively solved using the adaptive method [32].

When the functional is changed to an integral of the sum of control component models, problem (1.1)–(1.4), (1.6) is reduced to (1.9), (1.10), (1.12), (1.13):

$$h\sum_{i=1}^{r}\sum_{k=1}^{N} \left(u_{ik}^{+} + u_{ik}^{-}\right) \longrightarrow \min,$$

$$\sum_{i=1}^{r}\sum_{k=1}^{N} h_l^{i,k}u_{ik} = g_l, \quad l = \overline{1,m},$$

$$l_{*i} \leqslant u_{ik} \leqslant l_i^*, \quad i = \overline{1,r}, \quad k = \overline{1,N},$$

$$u_{ik} = u_{ik}^{+} - u_{ik}^{-}, \quad i = \overline{1,r}, \quad k = \overline{1,N},$$

$$u_{ik}^{+} \geqslant 0, u_{ik}^{-} \geqslant 0, \quad i = \overline{1,r}, \quad k = \overline{1,N}.$$

This is a linear programming problem with the following dimensions:

– $3rN$ variables,

– $rN + m$ linear equations,

– $4rN$ linear inequalities,

– a linear objective function.

Finally, using the Lagrange functional, control problem (1.1)–(1.4), (1.7) will become quadratic programming problem (1.9), (1.10), (1.14):

$$h \sum_{i_1=1}^{r} \sum_{i_2=1}^{r} q_{i_1 i_2} \sum_{k=1}^{N} u_{i_1 k} u_{i_2 k} \longrightarrow \min,$$

$$\sum_{i=1}^{r} \sum_{k=1}^{N} h_l^{i,k} u_{ik} = g_l, \quad l = \overline{1,m},$$

$$l_{*i} \leqslant u_{ik} \leqslant l_i^*, \quad i = \overline{1,r}, \quad k = \overline{1,N}.$$

The characteristics of this problem are written out below:

– rN variables,

– m linear equations,

– $2rN$ linear inequalities,

– a convex quadratic objective function.

1.4 Linear and Quadratic Programming Problem Solution

When solving a variety of optimal control problems, it becomes necessary to find optimal values for various mathematical programming problems. In this dissertation we will encounter several classes of optimization problems:

1. Linear Programming (LP),
2. Quadratic Programming (QP),
3. Quadratically Constrained Programming (QCP),
4. Mixed Integer Linear Programming (MILP),
5. Mixed Integer Quadratic Programming (MIQP),
6. Mixed Integer Quadratically Constrained Programming (MIQCP).

Despite the fact that some problems are special cases of others (see Fig. 1.1), it turns out to be more efficient to consider solution methods for each particular class. In this paragraph, we will describe methods for solving each of the presented classes and give examples of specialized programs that can be used to find optimal solutions of these problems.

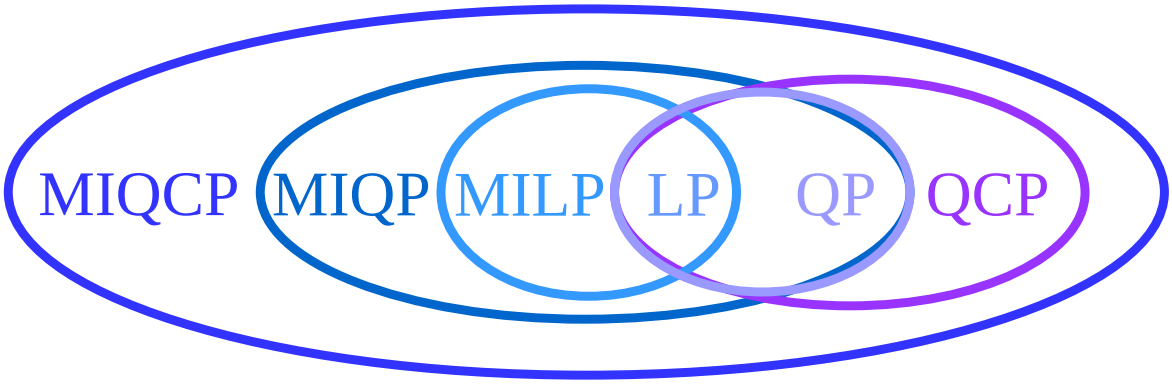

Figure 1.1 — Relation between Types of Optimization Tasks

1.4.1 Linear Programming

The LP problem is a classical optimization problem: the goal is to find an extremum of a multidimensional linear function under linear constraints on the variables. Geometrically, the linear constraints generate a convex polyhedron in multidimensional space, and the objective function defines a direction vector. The optimal solution is some vertex of the polyhedron, and in some cases optimal solutions can be all points belonging to a certain edge or facet.

The founder of linear programming as a separate class of optimization problems is considered to be L. V. Kantorovich, who also proposed a method for solving the problem [33].

Simplex Method. In 1947, while working on the problem of improving the planning process for the Air Force, G. Dantzig formulated the simplex

method [34], which is an effective method of finding an optimal solution and is considered one of the most popular approaches to this day.

Interior Point Method. The algorithm was developed in the 60s and 70s [35; 36] for convex optimization problems with constraints. In 1984, the interior point method (Karmarkar's algorithm [37]) was adapted specifically for the linear programming problem, solving the problem in polynomial time.

Adaptive Method. In the early 1970s, R. F. Gabasov together with F. M. Kirillova proposed and substantiated a new approach to solving linear programming problems. This initiated a major trend known as constructive optimization methods [38]. The results of this direction were widely used in the 70–80s in solving applied problems.

One of the advantages of the adaptive method is that its use does not lead to an increase in the dimensionality of the problem, which is inevitable when reducing the original problem to the canonical form of the linear programming problem, suitable for the solution of the simplex method [32].

Ellipsoid Method. Among other methods, the ellipsoid method [39] can be additionally mentioned, which was the first polynomial method for solving the linear programming problem, but in practice has not proved to be effective enough.

1.4.2 Quadratic Programming

In the literature it is common to consider separately two classes of quadratic problems:

1. QP, which is an optimization problem with linear constraints and a quadratic objective function,
2. QCP, which is an optimization problem with linear and quadratic constraints and a linear or quadratic objective function.

An important characteristic of quadratic problems is the convexity condition of the constraints and the objective function. In the convex case, the set of admissible solutions contains a single optimum, which is global. The complexity of solving a problem depends significantly on the fulfilment of this condition. In this paper we consider optimization problems only with convex constraints and objective functions.

To solve the QP problem, the active-set [40] and interior point [41] methods are used. The problem with quadratic constraints is more complicated. It is usually considered as a special case of the more general Second-Order Cone Programming (SOCP) problem [42]. Using the interior point method, the optimal solution of this problem can be found in polynomial time [43].

1.4.3 Mixed Integer Programming

A more general case of linear and quadratic problems are partially integer problems — when some variables are binary or integer. Let's describe ways to solve the MILP problem, assuming that similar approaches are valid for MIQP problems.

Brute-Force Search. The simplest approach to solving the problem is a brute-force search of the values of binary and integer variables. The values of discrete variables are fixed, thus the problem is reduced to a linear programming problem, which is solved by standard methods. Among all sets of values for which there is an admissible solution, we choose the one at which the optimum of the objective function is reached.

Branch and Bound. The algorithm was proposed in the 1960s [44] and is, in a sense, a more efficient brute-force search of discrete variables. In the first step, the relaxed problem is solved — all binary and integer variables are

treated as real variables. After solving, in case the optimal solution exists, two situations are possible:

1. The values of all discrete variables are integers. Then the optimal solution of the relaxed problem is an optimal solution of the original problem.
2. If there is a discrete variable that took a non-integer value $x_i = x_i^*$, then the original problem must be split into two. In the first subtask we impose an additional constraint $x_i \leqslant \lfloor x_i^* \rfloor$, in the second — an additional constraint $x_i \geqslant \lceil x_i^* \rceil$. Both subtasks are solved by the same algorithm, so the algorithm is recursive. If there are solutions for both subtasks, the solution of the original problem will be the solution of the subtask with the best value of the objective function. Despite the theoretical advantages of the method, guaranteed convergence to the optimal solution in practice is often too slow.

Branch and Cut. One of the improvements of the branch and bound method is the branch and cut method proposed in the mid-1990s [45; 46]. In addition to branching, for each subproblem the algorithm involves cutting off solutions that are known to be non-numerical by introducing additional constraints. There are many cutoff algorithms, one of the most popular being the Gomory algorithm [47; 48].

Branch and Price. A modification of the branch and bound method based on the column generation method [49]. At the first step an incomplete problem which does not contain all variables and constraints of the original problem is generated. This is done on the assumption that some of the variables will take on a null value, hence they can be excluded from the problem. Usually, the Dantzig—Wolfe decomposition [50] is used to reformulate the problem. Then, during the branching algorithm, certain variables can be returned to the model. This approach reduces computation time and the use of RAM. There is also a branch-price-and-cut algorithm [51], where an additional step of cutting off non-integer solutions is included.

Heuristics. To accelerate optimization, it may be justified to use heuristic algorithms in parallel with the branch and cut method. The main advantage of such algorithms is the ability to quickly find quality solutions, however, the drawback is the lack of guarantee of finding a solution and impossibility to evaluate their optimality. Examples of heuristic algorithms are:

1. Genetic Algorithm [52],
2. Local Branching [53],
3. RINS (Relaxation Induced Neighborhood Search) [54].

1.4.4 Use of Specialized Software

There are various software packages for solving mathematical programming problems in which the above solution algorithms are implemented. Such complexes are usually presented in the form of software libraries, they are commonly referred to as solvers.

To select a solver, you need to consider the specifics of the task to be solved and the overall architecture of the project. As a rule, you can be guided by the following criteria:

1. Types of problems to be solved,
2. Manufacturer, license type, license cost,
3. Performance on benchmarks, i. e. on a set of standard tasks [55],
4. Availability of a user-friendly manual, manufacturer supportб and user community,
5. Application Programming Interface (API),
6. Possibility to specify a starting solution — the use of the initial solution allows you to accelerate the search for an optimum,
7. Possibility to manage solver settings,

8. Analysis of incompatible models — if there are no admissible solutions, the algorithm finds a set of conflicting constraints,
9. Possibility to use custom methods — finding a solution using algorithms that take into account the specifics of the problem in parallel with the basic solver methods.

Table 1.1 gives a small comparison of popular solvers. We consider the best known commercial solvers Gurobi [56] and CPLEX [57], released in 2019 COPT [58], and their publicly available counterparts SCIP [59] and CBC [60]. You can also find a list of solvers with classification by type of problem at the GAMS website for mathematical modelling [61].

Table 1.1 — Solver Comparison

Metrics	**Gurobi**	**CPLEX**	**COPT**	**SCIP**	**CBC**
Developer	Gurobi Optimization, USA	IBM, USA	Cardinal Operation, China	Zuse Institute Berlin, Germany	COIN-OR Foundation
License Type	Commercial and free academic license	Commercial and free academic license	Commercial and free academic license	Free use for research purposes by members of nonprofit organizations	Free distribution
Problem Types	(MI)LP, (MI)QP, (MI)QCP	(MI)LP, (MI)QP, (MI)QCP	(MI)LP, (MI)QP, (MI)QCP	(MI)LP, (MI)QP, (MI)QCP	(MI)LP
API	C/C++, Java, .NET, Python, MATLAB, R	C/C++, Java, .NET, Python, MATLAB	C/C++/C#, Java, Python	C/C++, Python, Julia, MATLAB, Java	C++, Mathematica, MATLAB, R, Python, Julia, Rust
LP Benchmarks (Simplex Method)	1.43	—	1	44.9	9.52
LP Benchmarks (Interior Point Method)	1.33	—	1	—	77.8
MILP Benchmarks	1	—	2.34	8.39	10.2
QCP Benchmarks	2.03	—	1	—	—

MIQCP Benchmarks	1	—	—	13.1	—

For benchmarks the average relative time to solve problems is given, normalized by the result of the leader. Gaps in the data indicate a lack of performance information on the specified benchmarks.

In the current work we chose COPT to solve optimization problems. The solver was accessed through the Python programming language using the *coptpy* library provided by the solver manufacturers.

1.5 Example. Damping of Sleeping Lagrange Top

As an illustration of how the algorithm works, consider the problem of the damping of sleeping Lagrange top [62].

The motion of the top is described by a system of ordinary differential equations with control:

$$\ddot{\xi} + B\dot{\eta} - A\xi = u_1,$$
$$\ddot{\eta} - B\dot{\xi} - A\eta = u_2,$$

where $\xi(t)$ and $\eta(t)$ denote the coordinates of the top of the wave; $u_1(t)$, $u_2(t)$ are the generalized forces; A and B are the model parameters characterizing the physical properties of the object. In this example we assume $A = 1$, $B = 3$.

The terminal problem is to transfer the object from the position

$$(\xi = -10,\ \dot{\xi} = 5,\ \eta = 6,\ \dot{\eta} = 8)$$

to the zero position. Thus, at the initial moment the object is at $(-10,\ 6)$ and has a non-zero velocity. It is necessary to place the object at rest.

We set the parameter T to 20, that is, we will consider the process on the interval [0, 20]. Let's divide this time interval into 50 equal sections. Set the

lower bounds equal to -2, and set the upper bounds equal to 2. We will search for the control in the class of piecewise constant functions.

As the objective function we will consider the integral of the square of the norm of the control vector.

Then after changing the variables

$$x_1 = \xi, \quad x_2 = \dot{\xi}, \quad x_3 = \eta, \quad x_4 = \dot{\eta},$$

we obtain the problem of optimal control with a linear system and a quadratic functional:

$$\int_0^{20} \left(u_1^2(t) + u_2^2(t)\right) dt \longrightarrow \min,$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -3 \\ 0 & 0 & 0 & 1 \\ 0 & 3 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix},$$

$$\begin{pmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \\ x_4(0) \end{pmatrix} = \begin{pmatrix} -10 \\ 5 \\ 6 \\ 8 \end{pmatrix}, \quad \begin{pmatrix} x_1(20) \\ x_2(20) \\ x_3(20) \\ x_4(20) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$\begin{pmatrix} -2 \\ -2 \end{pmatrix} \leqslant \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} \leqslant \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \quad t \in [0,\, 20].$$

The reduction procedure was implemented in Python 3, and the quadratic programming problem was solved using the COPT solver (see Appendix A).

The total running time of the program was 0.32 seconds. The value of the objective function is 92.80. The results are shown in Fig. 1.2.

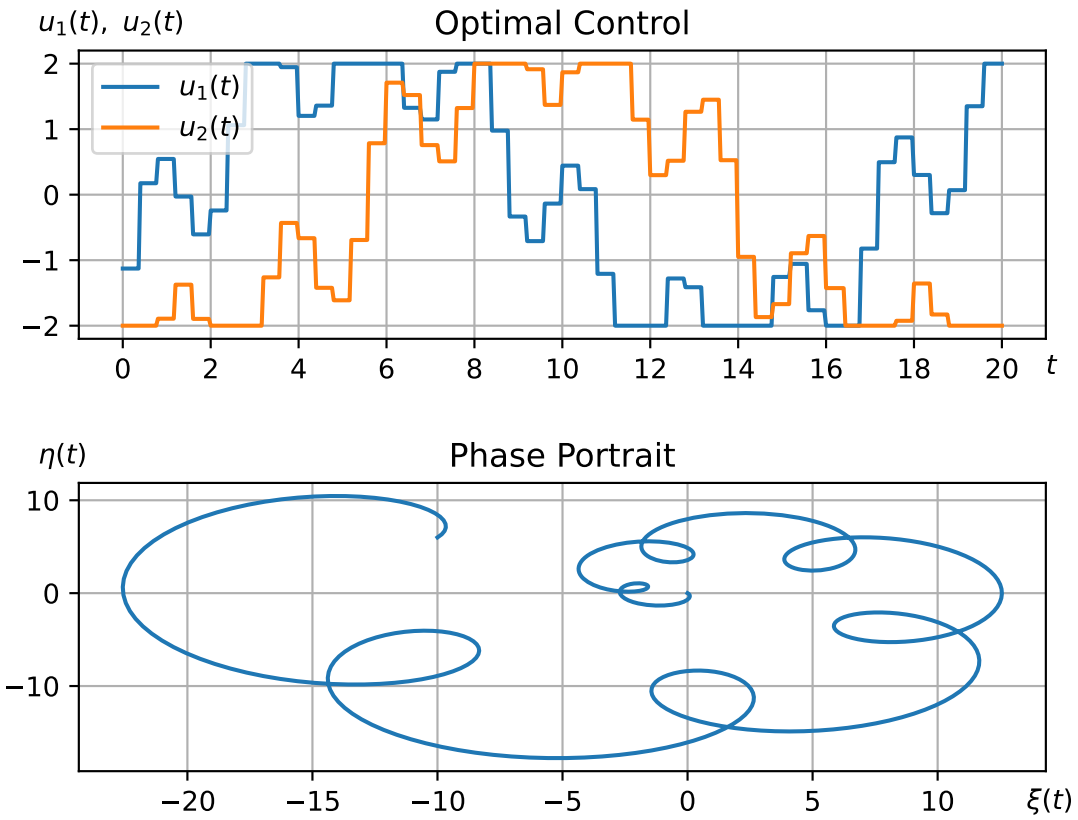

Figure 1.2 — Dumping of Sleeping Lagrange Top in Case of Control from Class of Piecewise Constant Functions

1.6 Chapter 1 Conclusion

In Chapter 1, we described an algorithm for solving the linear optimal control problem, which consists of two main steps:

1. **Reduction to a mathematical programming problem**. This transition is performed using the Cauchy formula and is possible due to the choice of controls from the class of piecewise constant functions. The reduction step is described in detail in Paragraph 1.3. Depending on the type of the objective function, a linear or convex quadratic programming problem is formed.
2. **Solving the mathematical programming problem**. Since a linear programming problem and a convex quadratic programming

problem are well studied, we can use one of the specialized software products (solvers) to solve them. Paragraph 1.4 describes ideas for basic algorithms to solve such problems, gives an overview of existing solvers, and compares their speed for different classes of problems.

The results presented in Chapter 1 are based on the theoretical basis formed by the scientific school of R. F. Gabasov [31; 32; 38]. In terms of the overall structure of the thesis work, the current chapter is the basis on which we will rely in the following sections. Several directions of development have been chosen, which are relevant, but contain unresolved issues, in consequence of which they are of research interest:

1. Construction of reachability and controllability sets for basic problem (1.1)–(1.4).
2. Choice of control from alternative classes of functions: piecewise linear and piecewise quadratic.
3. Addition of more convex and non-convex constraints on control and phase variables to the basic model.
4. Replacement of a linear system of ordinary differential equations with a non-linear one.
5. Real-time control restructuring when the object deviates from the program mode trajectory.

CHAPTER 2. CONSTRUCTION OF REACHABILITY AND CONTROLLABILITY SETS

Practical implementation of optimal control problems with the transition to linear programming has shown the theoretical importance of justifying the feasibility of solving the problem, that is, the existence of the optimal solution. This led to the problem of analyzing the sets of reachability and controllability of the original problem. Chapter 2 considers the construction of these sets. Paragraph 2.1 contains definitions of the sets of reachability and controllability. Paragraph 2.3 shows that the problems of constructing these sets are equivalent and can be reduced to the linear mapping problem for a multidimensional cube. Then, in Paragraph 2.5, we analyze the existing approaches to the solution of this problem. Since they are excessively labor-intensive, the question of creating a more efficient algorithm arises. In 2.6 we propose an algorithm for constructing sought sets as a system of linear inequalities. The correctness of the algorithm is proved as a theorem. Further the complexity of the presented approach is estimated.

2.1 Problem Formulation

The purpose of this dissertation is to solve the optimal control problem for various classes of systems. However, it may not be known in advance whether the solution of the problem exists. To answer this question it will be useful to construct two sets: the **reachability set** is all possible states of the object that can be reached in a certain time under the action of an admissible control, and the **controllability set** is all initial positions of the system from which it is possible to achieve the final position in a given time by an admissible control.

Let's consider control problem (1.1)–(1.4). It can be supplemented with any of objective functions (1.5)–(1.7), however, they will have no effect on the existence of admissible controls. Therefore, the current chapter will deal with the terminal control problem without any criterion.

Now let's formalize the notions of reachability set and controllability set.

Definition 1. Let the controlled object be in the state $\hat{x}$ at the moment $\hat{t}$ $(0 \leqslant \hat{t} < T)$. We call the *reachability set* $X_1(t, \hat{t}, \hat{x})$ of problem (1.1)–(1.4) as a set of all possible states $x^{(1)}(t)$, where $t \in (\hat{t},\ T]$, to which system (1.1) can move in time $t - \hat{t}$ under the initial condition $x(\hat{t}) = \hat{x}$ and the selection of a control of form (1.3) satisfying the constraint:

$$X_1(t, \hat{t}, \hat{x}) = \{x^{(1)}(t) = x(t) \mid \dot{x}(\tau) \equiv A(\tau)x(\tau) + B(\tau)u(\tau) + d(\tau),$$
$$x(\hat{t}) = \hat{x},\ \tau \in [\hat{t},\ t],\ u(\tau) \text{ satisfies (1.3), (1.4)}\}.$$

Definition 2. Let at time $\hat{t}$ $(0 < \hat{t} \leqslant T)$ the controlled object be in the state $\hat{x}$. We call the *controllability set* $X_2(t, \hat{t}, \hat{x})$ of problem (1.1)–(1.4) as a set of all possible states $x^{(2)}(t)$, where $t \in [0,\ \hat{t})$, from which system (1.1) can move in time $\hat{t} - t$ to the state $x(\hat{t}) = \hat{x}$ by selecting a control of form (1.3) satisfying constraint (1.4):

$$X_2(t, \hat{t}, \hat{x}) = \{x^{(2)}(t) = x(t) \mid \dot{x}(\tau) \equiv A(\tau)x(\tau) + B(\tau)u(\tau) + d(\tau),$$
$$x(\hat{t}) = \hat{x},\ \tau \in [t,\ \hat{t}],\ u(\tau) \text{ satisfies (1.3), (1.4)}\}.$$

The purpose of the current chapter is to describe these sets in a convenient form for analysis.

Although a linear controllable system is considered, methods for solving non-linear problems often consist of using linearized systems. Therefore, it is assumed that the results obtained here will be applied to algorithms for finding optimal control in non-linear systems in Chapter 4.

2.2 Literature Review

The problem of constructing sets of reachability and controllability has been repeatedly considered in classical sources [15; 63; 64], in particular, they present constructive methods for constructing sets of reachability in the form of ellipsoids.

In the modern literature one can find works devoted to the construction of sets for the linear problem [65;66]. There are also approaches to the construction of reachability sets for non-linear systems [67–69], however, the described approaches represent labor-intensive algorithms consisting, for example, in solving the optimal control problem.

The difference of this work is the emphasis on a special type of control: it is selected from a class of piecewise constant functions, due to which, as will be shown below, the sets of reachability and controllability are convex polyhedrons and are described by systems of linear inequalities. The goal is to implement a constructive algorithm for constructing these sets, which can be used to quickly check whether a point belongs to a reachability or controllability set.

The results presented in the current chapter were published in [11].

2.3 Transition to Linear Mapping Problem

Let's show that the problems of constructing the sets $X_1(t, \hat{t}, \hat{x})$ and $X_2(t, \hat{t}, \hat{x})$ are equivalent and can be reduced to the problem of linear representation of a convex polyhedron.

2.3.1 Reduction for Reachability Set

For problem (1.1) with initial condition $x(\hat{t}) = \hat{x}$ we write out the Cauchy formula, denoting $\hat{Y}(t)$ as a state-transition matrix of the system normalized at $\hat{t}$:

$$x(t) = \hat{Y}(t) \left(\hat{x} + \int_{\hat{t}}^{t} \hat{Y}(t)\hat{Y}^{-1}(\tau) \left(Bu(\tau) + d(\tau) \right) d\tau \right) \tag{2.1}$$

and use the representation of the control in form (1.3).

Among the numbers $\overline{1, N}$ we find k_1 such that $(k_1 - 1)h \leqslant \hat{t} < k_1 h$. Similarly, we obtain a number k_2 such that either $t \in [0, T)$, then among the numbers $\overline{1, N}$ we can find k_2 for which $(k_2 - 1)h \leqslant t < k_2 h$, or $t = T$, then we assume $k_2 = N$.

Let's set a sequence

$$s_k = \begin{cases} \hat{t} & \text{if } k = k_1 - 1, \\ kh & \text{if } k = \overline{k_1, k_2 - 1}, \\ t & \text{if } k = k_2. \end{cases} \tag{2.2}$$

Now rewrite (2.1) by expanding the brackets and using the representation of the control as (1.3), taking into account (2.2):

$$\begin{aligned} x(t) = \hat{Y}(t)\hat{x} + \sum_{i=1}^{r} \sum_{k=k_1}^{k_2} \left(\int_{s_{k-1}}^{s_k} \hat{Y}(t)\hat{Y}^{-1}(\tau) b_i(\tau) d\tau u_{i,k} \right) + \\ + \int_{\hat{t}}^{t} \hat{Y}(t)\hat{Y}^{-1}(\tau) d(\tau) d\tau. \end{aligned} \tag{2.3}$$

Here b_i denotes the i-th column of the matrix B.

Let's make substitutions:

$$\begin{gathered} U_1 = \left(u_{1,k_1}, u_{2,k_1}, \ldots, u_{r,k_1}, u_{1,k_1+1}, u_{2,k_1+1}, \ldots, u_{r,k_2}\right)^T, \\ b_{i,k}^{(1)} = \int\limits_{s_{k-1}}^{s_k} \hat{Y}(t)\hat{Y}^{-1}(\tau)b_i d\tau, \\ B_1 = \left(b_{1,k_1}^{(1)}, b_{2,k_1}^{(1)}, \ldots, b_{r,k_1}^{(1)}, b_{1,k_1+1}^{(1)}, b_{2,k_1+1}^{(1)}, \ldots, b_{r,k_2}^{(1)}\right), \\ q_1 = \hat{Y}(t)\hat{x} + \int\limits_{\hat{t}}^{t} \hat{Y}(t)\hat{Y}^{-1}(\tau)d(\tau)d\tau. \end{gathered} \tag{2.4}$$

Then we rewrite equation (2.3) with (2.4) as follows:

$$x = B_1 U_1 + q_1. \tag{2.5}$$

Remark 2.1. Generally speaking, the matrix B_1 and the vector q_1, and hence the vector x depend on the parameters t, $\hat{t}$, and $\hat{x}$. Moreover, the choice of t and $\hat{t}$ affects the size of the vector U_1 and the number of columns in B_1. We assume that they are fixed, and for simplicity we omit the notation of dependencies.

Thus, we obtained a system of linear algebraic equations in which the matrix B_1 has dimension $n \times R$, where $R = r(k_2 - k_1 + 1)$. The final value of R is determined by the choice of points t and $\hat{t}$, for example, for $t = 0$ and $\hat{t} = T$ we have $R = rN$.

Let's introduce $L_1^{(1)}$ and $L_2^{(1)}$ as R-dimensional vectors consisting of $k_2 - k_1 + 1$ repeats of vectors $(l_{*1}, \ldots, l_{*r})^T$ and $(l_1^*, \ldots, l_r^*)^T$ respectively.

Then the direct constraints on the components of vector U_1 can be rewritten as

$$L_1^{(1)} \leqslant U_1 \leqslant L_2^{(1)}. \tag{2.6}$$

As we see, the problem of constructing a reachability set is to find all vectors $x(t)$ such that there exists the vector U_1 whose components satisfy two-sided constraints (2.6), and linear equation (2.5) is satisfied.

Let's simplify the form of problem (2.5), (2.6). We replace the vector of controls and the vector of phase variables:

$$\begin{gathered} v_i = \frac{2U_{1,j} - \left(L_{1,j}^{(1)} + L_{2,j}^{(1)}\right)}{L_{1,j}^{(2)} - L_{1,j}^{(1)}}, \quad j = \overline{1,R}, \\ y = x - \frac{1}{2}B_1\left(L_1^{(1)} + L_2^{(1)}\right) - q_1. \end{gathered} \tag{2.7}$$

We also introduce a matrix P of size $n \times R$, consisting of columns

$$p_j = \frac{1}{2}B_{1,j}\left(L_{2,j}^{(1)} - L_{1,j}^{(1)}\right),$$

where $B_{1,j}$ is the j-th column of the matrix D_1.

In new variables (2.7), problem (2.5), (2.6) will look like

$$\begin{gathered} y = Pv, \\ -1 \leqslant v_j \leqslant 1,\ j = \overline{1,R}. \end{gathered} \tag{2.8}$$

Thus, to construct the reachability set it is necessary to perform a linear mapping of an R-dimensional cube into an n-dimensional space. The matrix of the linear operator is the matrix P of dimension $n \times R$.

2.3.2 Reduction for Controllability Set

Similarly, let's reduce the problem of finding the controllability set. For problem (1.1) with finite condition $x(\hat{t}) = \hat{x}$ we consider the Cauchy formula by expressing $\hat{x}$:

$$\hat{x} = \hat{Y}^{-1}(t)x(t) + \int_t^{\hat{t}} \hat{Y}^{-1}(\tau)\left(B(\tau)u(\tau) + d(\tau)\right)d\tau. \tag{2.9}$$

Among the numbers $\overline{1,N}$, find k_1 and k_2 such that $(k_1-1)h \leqslant t < k_1 h$, and either $\hat{t} \in [0,\ T)$, then there exists k_2 such that $(k_2-1)h \leqslant \hat{t} < k_2 h$, or $\hat{t} = T$, then $k_2 = N$.

Let's set a sequence

$$s_k = \begin{cases} t & \text{if } k = k_1 - 1, \\ kh & \text{if } k = \overline{k_1, k_2 - 1}, \\ \hat{t} & \text{if } k = k_2. \end{cases} \tag{2.10}$$

We rewrite (2.9) by expanding the brackets and using the representation of the control in form (1.3), taking into account (2.10):

$$\begin{aligned} x(t) = \hat{Y}(t)\hat{x} - \sum_{i=1}^{r} \sum_{k=k_1}^{k_2} \left(\int_{s_{k-1}}^{s_k} \hat{Y}(t)\hat{Y}^{-1}(\tau) b_i(\tau) d\tau u_{ik} \right) - \\ - \int_{t}^{\hat{t}} \hat{Y}(t)\hat{Y}^{-1}(\tau) d(\tau) d\tau. \end{aligned} \tag{2.11}$$

Let's introduce a notation:

$$\begin{gathered} U_2 = \left(u_{1k_1}, u_{2k_1}, \ldots, u_{rk_1}, u_{1k_1+1}, u_{2k_1+1}, \ldots, u_{rk_2}\right)^T, \\ b_{ik}^{(2)} = - \int_{s_{k-1}}^{s_k} \hat{Y}(t)\hat{Y}^{-1}(\tau) b_i d\tau, \\ B_2 = \left(b_{1k_1}^{(2)}, b_{2k_1}^{(2)}, \ldots, b_{rk_1}^{(2)}, b_{1k_1+1}^{(2)}, b_{2k_1+1}^{(2)}, \ldots, b_{rk_2}^{(2)}\right), \\ q_2 = \hat{Y}(t)\hat{x} - \int_{t}^{\hat{t}} \hat{Y}(t)\hat{Y}^{-1}(\tau) d(\tau) d\tau. \end{gathered} \tag{2.12}$$

Then we rewrite equation (2.11) with (2.12) as

$$x = B_2 U_2 + q_2. \tag{2.13}$$

Inequalities (2.5), (2.13) have the same form. Then, taking into account the direct constraints, the problems of constructing the reachability set and the manageability set can be reduced to problem (2.8). In this sense they are equivalent.

2.4 Properties of Sets

Here and below we will consider the properties of the reachability set, implying that they are also true for the controllability set.

Taking into account the results gained in Paragraph 2.3, let's introduce two sets.

Definition 3. We call the set $\mathbb{V} = [-1;\ 1]^R$ as a *set of admissible controls*, and the set $\mathbb{Y} = \{y \in \mathbb{R}^n |\ \ \exists v \in \mathbb{V} : y = Pv\}$ as a *set of admissible states*.

Property 1. $\mathbb{Y}$ is a nonempty, bounded, closed, convex set.

It follows from the fact that the set $\mathbb{V}$ is nonempty, bounded, closed, and convex, and the linear mapping retains the above properties.

Property 2. $\mathbb{Y}$ is a convex hull of points $\{p^1, \ldots, p^{2^R}\}$. Here $p^k = Pv^k$, and v^k are vertices of $\mathbb{V}$.

The set $\mathbb{V}$ is evidently a convex hull of its vertices, i. e. $\forall v \in V\ \exists \lambda_k$, $k = \overline{1,2^R}$: $v = \sum\limits_{k=1}^{2^R} \lambda_k v^k$, where $\sum\limits_{k=1}^{2^R} \lambda_k = 1$, $\lambda_k \geqslant 0$.

Then $\forall y \in Y\ \exists \lambda_k$ such that

$$y = \sum_{k=1}^{2^R} \lambda_k P v^k = \sum_{k=1}^{2^R} \lambda_k p^k, \quad \sum_{k=1}^{2^R} \lambda_k = 1, \quad \lambda_k \geqslant 0, \quad k = \overline{1,2^R}.$$

Hence, the set $\mathbb{Y}$ is a convex hull of vectors $p^k = Pv^k$.

Property 3. $\mathbb{Y}$ is a convex polyhedron whose vertices are a subset of the set of vectors $\{p^1, \ldots, p^{2^R}\}$, where $p^k = Pv^k$, v^k are vertices of $\mathbb{V}$.

It follows from Property 2 and the properties of convex hulls.

Property 4. If $y \in \mathbb{Y}$, then also $-y \in \mathbb{Y}$.

If $y \in \mathbb{Y}$, then $\exists v \in \mathbb{V} :\ \ y = Pv$. But also $-v \in \mathbb{V}$, and therefore $P(-v) = -y \in \mathbb{Y}$.

Property 5. The set $\mathbb{Y}$ is described by a system of inequalities like $-b \leqslant Ay \leqslant b$.

This set is a convex polyhedron and can be described by a system of linear inequalities. According to Property 4, if $a_i y \leqslant b_i$ holds, then this inequality is also true for $-y : a_i(-y) \leqslant b_i$. Then $-b_i \leqslant a_i y$.

2.5 Algorithms Overview

There are several approaches to solving the problem (2.8). We briefly describe them below.

1) Construction of Convex Hull. As shown in Property 3, the polyhedron $\mathbb{Y}$ is a convex hull of points Pv^k, where v^k are vertices of the cube $\mathbb{V}$. However, even with a sufficiently small dimension R, this approach looks unrealizable, given the computational complexity of algorithms for constructing convex hulls. Thus, if $R = 100$, we obtain $2^{100} \approx 10^{30}$ vertices of the cube.

2) Minkowski Addition. In the space $\mathbb{R}^n$, we define R segments: from the point $-p^1$ to the point p^1, from the point $-p^2$ to the point p^2, etc. Then find the set $\mathbb{Y}_1$ as the geometric place of points belonging to the first segment: $\mathbb{Y}_1 = \{y \in \mathbb{R}^n |\ y = \gamma p^1, -1 \leqslant \gamma \leqslant 1\}$. Define the set $\mathbb{Y}_2$ as the Minkowski sum of the set [70] and the second segment:

$$\mathbb{Y}_2 = \{y_1 + y_2 \in \mathbb{R}^n |\ y_1 \in \mathbb{Y}_1,\ y_2 = \gamma p^2, -1 \leqslant \gamma \leqslant 1\}.$$

Continue constructing the sums for all the segments, and at the end we get the set $\mathbb{Y}$:

$$\mathbb{Y} = \left\{y_1 + y_2 \in \mathbb{R}^n |\ y_1 \in \mathbb{Y}_{R-1},\ y_2 = \gamma p^R, -1 \leqslant \gamma \leqslant 1\right\}$$

Unfortunately, this approach also has exponential complexity. The first step will construct a segment connecting two points, the second step will construct a polygon with four vertices, and the last step will construct a polyhedron that can contain 2^R vertices.

3) Fourier — Motzkin Elimination. Algorithms that work not with vertices of the polyhedron, but with its faces, look more promising. They include the Fourier — Motzkin elimination [71]. Its main idea is to exclude variables from the system of inequalities. The method iteratively removes one variable each from the inequalities. Then, if we consider system (2.8) as a system of inequalities with respect to variables y and v, the problem is reduced to the exclusion of R variables v_i from this system.

Nevertheless, this approach also has a disadvantage — when excluding variables, too many redundant constraints are generated, and if you do not eliminate them at each step, the end of the algorithm will result in a system of $n \cdot 2^R$ inequalities.

2.6 Set Construction Method. Theorem

We set the goal of building a more efficient algorithm. The first step is to remove unnecessary (linearly dependent) state variables. If the rank of matrix P is less than the number of rows, we select $n - \operatorname{rank} P$ components of vector y and express them through the rest. Thus, we will obtain $n - \operatorname{rank} P$ constraints of the equation type. Further on, we will assume that this step is passed and there is a matrix P of full rank: $\operatorname{rank} P = n \leqslant R$. Besides, we will assume that $n \geqslant 2$ (at $n = 1$ the algorithm of construction of the set is rather obvious).

Now we will go through the columns of the matrix P. We select $n - 1$ linearly independent columns and find a fundamental system of solutions with respect to the vector-row $\tilde{a}$ for the equation

$$\tilde{a}\left(p^{j_1}, \ldots, p^{j_{n-1}}\right) = 0. \tag{2.14}$$

This is a one-parameter family of solutions. Let's choose some value from it. For example, we can add the constraint $\|\tilde{a}\| = 1$ for uniqueness of the solution.

In fact, vector $\tilde{a}$ is a normal vector of the plane in the space $\mathbb{R}^n$, constructed by vectors $p^{j_1}, \ldots, p^{j_{n-1}}$.

Now we multiply $\tilde{a}$ by the matrix P and find the maximum product of the resulting expression by the vector $v \in \mathbb{V}$:

$$\tilde{\beta} = \max_{v \in \mathbb{V}} \tilde{a} P v = \sum_{j=1}^{R} |\tilde{a} p^j|, \tag{2.15}$$

where $\tilde{a}P$ is an R-dimensional vector-row containing the values of projections of the vector $\tilde{a}$ to the columns of the matrix P. By construction, the projections to vectors $p^{j_1}, \ldots, p^{j_{n-1}}$ are equal to zero. Since the components of the vector v vary from -1 to 1 and do not depend on each other, the maximum value of the item $\tilde{a}p^j v_j$ is reached when $v = \operatorname{sgn}(ap^j)$ and is equal to $|ap^j|$.

Let's introduce a matrix A consisting of vector-rows $\tilde{a}$ constructed for all sets of linearly independent columns $p^{j_1}, \ldots, p^{j_{n-1}}$ of the matrix P, and a vector b consisting of numbers $\tilde{\beta}$ corresponding to rows $\tilde{a}$. We also introduce the set

$$\bar{\mathbb{Y}} = \{y \in \mathbb{R}^n | \ -b \leqslant Ay \leqslant b\}.$$

Theorem 2.1. *$y \in \mathbb{Y}$ if and only if $y \in \bar{\mathbb{Y}}$.*

P r o o f . *Necessity.* Let the vector $y \in \mathbb{Y}$. Then there exists a vector v such that $y = Pv$ and $v \in \mathbb{V}$. Let's show that $-b \leqslant Ay \leqslant b$.

Substituting the expression Pv instead of y in this inequality, for the row l we obtain

$$-\beta_l \leqslant a_l P v \leqslant \beta_l,$$

or, the same,

$$-\sum_{j=1}^{R} |a_l p^j| \leqslant \sum_{j=1}^{R} a_l p^j v_j \leqslant \sum_{j=1}^{R} |a_l p^j|.$$

Since $v_j \in [-1; 1]$, $j = \overline{1, R}$, the inequality is obviously satisfied for any $l = \overline{1,k}$, so $y \in \bar{\mathbb{Y}}$.

Sufficiency.

I. Consider a matrix A whose dimension is $k \times n$, where k is a number of different sets of $n-1$ linearly independent columns of P. First, let's show that $k \geqslant n$. Since $\operatorname{rank} P = n$, we can find n linearly independent columns in the matrix P. Then we can make n sets of $n-1$ linearly independent columns in each.

Now let's show that $\operatorname{rank} A = n$. We construct a matrix $\bar{P}$ of n linearly independent columns of matrix P: $\bar{P} = (p^{j_1}, \ldots, p^{j_n})$. By construction, the matrix A contains n rows $a_{l_1}, \ldots, a_{l_n}$ such that

$$a_{l_i} p^{j_1} = \cdots = a_{l_i} p^{j_{i-1}} = a_{l_i} p^{j_{i+1}} = \cdots = a_{l_i} p^{j_n} = 0, \quad i = \overline{1,n}.$$

We make a matrix $\bar{A}$ from these rows. Then

$$\bar{A}\bar{P} = \begin{pmatrix} a_{l_1} p^{j_1} & 0 & \cdots & 0 \\ 0 & a_{l_2} p^{j_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{l_n} p^{j_n} \end{pmatrix}.$$

Next, we assume that $a_{l_i} p^{j_i} = 0$. This means that $a_{l_i} \bar{P} = (0\ 0\ \ldots\ 0)$, which implies linear dependence of the columns of the matrix $\bar{P}$, which contradicts the conditions. Therefore $a_{l_i} p^{j_i} \neq 0\ \forall i = \overline{1,n}$. Then $\det(\bar{A}\bar{P}) \neq 0$, hence $\det(\bar{A}) \neq 0$, or the same, $\operatorname{rank} \bar{A} = n$.

Since the matrix $\bar{A}$ is composed of the rows of the matrix A, then $\operatorname{rank} A \geqslant n$. But since A contains n columns, then $\operatorname{rank} A = n$.

II. To prove sufficiency it is necessary to show: for any point $y \in \bar{\mathbb{Y}}$ there is a point $v \in \mathbb{V}$ such that $y = Pv$ is true. Let's start by proving this for vertices of the set $\bar{\mathbb{Y}}$.

Choose a point $\hat{y}$ from the space $\mathbb{R}^n$ such that $-b \leqslant A\hat{y} \leqslant b$ is satisfied. Among these inequalities there exist n such constraints that

1. the rows $a_{l_1}, \ldots, a_{l_n}$ generating these inequalities are linearly independent,

2. at the point $\hat{y}$ these inequalities become equal by one of the two-sided constraints. Without limiting the generality of the foregoing, we assume that the upper constraints transform to equations: $a_{l_i}\hat{y} = \beta_{l_i}, i = \overline{1,n}$.

Under these conditions, the point $\hat{y}$ will be the vertex of the polyhedron $\bar{\mathbb{Y}}$.

Since $\operatorname{rank} P = \operatorname{rank}\{P, \hat{y}\} = n$, the system of equations $Pv = \hat{y}$ will have at least one solution. It is necessary to show the existence of a solution at $v \in \mathbb{V}$.

Suppose the opposite: for any vector v that is a solution of the system $Pv = \hat{y}$, there is a number j^* that $|v_{j^*}| > 1$.

III. We define vector $\hat{v} : P\hat{v} = \hat{y}$ so that it has a number of convenient properties. For this purpose, we choose some equation $a_{l_i}\hat{y} = \beta_{l_i}$. For convenience we denote $\bar{a} = a_{l_i}$, $\bar{\beta} = \beta_{l_i}$.

The columns of matrix P can be divided into two classes: orthogonal and non-orthogonal with row $\bar{a}$. Let the number R_1 denote the quantity of orthogonal columns. It follows from the properties of the row $\bar{a}$ and the matrix P that $R_1 \geqslant n-1$, but there is at least one non-orthogonal column. For simplicity, let's rearrange the columns of P so that the orthogonal ones go first:

$$\begin{cases} \bar{a}p^j = 0, \ j = \overline{1, R_1}, \\ \bar{a}p^j \neq 0, \ j = \overline{R_1 + 1, R}. \end{cases}$$

Now suppose that $\hat{v}_j = \operatorname{sgn}(\bar{a}p^j), j = \overline{R_1 + 1, R}$. In this case, the equation $\bar{a}\hat{y} = \bar{\beta}$ will be fulfilled regardless of the values of the other $\hat{v}_j$:

$$\bar{a}\hat{y} = \bar{a}P\hat{v} = \sum_{j=1}^{R} \bar{a}p^j\hat{v}_j = \sum_{j=R_1+1}^{R} \bar{a}p^j \operatorname{sgn}(\bar{a}p^j) = \sum_{j=R_1+1}^{R} |\bar{a}p^j| = \bar{\beta}.$$

Further, we wonder about finding $\hat{v}_1, \ldots, \hat{v}_{R_1}$. By the construction of $\bar{a}$ we know that there exists a set of $n-1$ linearly independent columns of matrix P orthogonal to the row $\bar{a}$. Hence, $\operatorname{rank}\{p^1, \ldots, p^{R_1}\} = n-1$. Then we can select $n-1$ columns as the basis of this subspace, and express the other columns through them. Let the basis columns be the columns $p^1, \ldots, p^{n-1}$:

$$c_{1,j}p^1 + \cdots + c_{n-1,j}p^{n-1} + p^j = 0, \quad j = \overline{n, R_1}. \tag{2.16}$$

Next, we introduce a notation:

$$\hat{A} = \begin{pmatrix} a_{l_1} \\ \vdots \\ a_{l_{i-1}} \\ a_{l_{i+1}} \\ \vdots \\ a_{l_n} \end{pmatrix}, \quad \hat{b} = \begin{pmatrix} \beta_{l_1} - \sum_{j=R_1+1}^{R} a_{l_1} p^j \hat{v}_j \\ \vdots \\ \beta_{l_{i-1}} - \sum_{j=R_1+1}^{R} a_{l_{i-1}} p^j \hat{v}_j \\ \beta_{l_{i+1}} - \sum_{j=R_1+1}^{R} a_{l_{i+1}} p^j \hat{v}_j \\ \vdots \\ \beta_{l_n} - \sum_{j=R_1+1}^{R} a_{l_n} p^j \hat{v}_j \end{pmatrix}.$$

Then there is a system of $n-1$ equations with R_1 unknowns:

$$\hat{A}(p^1\hat{v}_1 + \cdots + p^{R_1}\hat{v}_{R_1}) = \hat{b}. \tag{2.17}$$

We will call those $\hat{v}_j$ which are at the columns included in the subspace basis as *basis variables*, and a solution of system (2.17) as a *basis* if the values of the non-basic variables are zero. In addition, we denote the set of indices of basis variables by the symbol B, and of non-basic variables by NB.

Lemma 2.1. *If for any solution of system* (2.17) *there is such a number* j^* *that* $|\hat{v}_{j^*}| > 1$, *then there exist a basis solution* $\tilde{v}$ *and number* j' *such that* $|\tilde{v}_{j'}| > 1 + \sum_{j \in NB} |c_{j',j}|$.

P r o o f . For simplicity, let the set of indices of basis variables equals $B = \{1, \ldots, n-1\}$, then $NB = \{n, n+1, \ldots R_1\}$. Let's introduce a matrix P_1 consisting of columns $(p^1, \ldots, p^{n-1})$.

Since the matrix $\hat{A}P_1$ has size $(n-1) \times (n-1)$ and its rank is $n-1$, system (2.17) has a single solution for the given basis. Let's find it and denote by $\tilde{v}_1, \ldots, \tilde{v}_{n-1}$.

We will prove the lemma by induction, depending on the number of non-basic columns.

1. Let $R_1 = n-1$. By the condition of the lemma, there exists j^* such that $|\tilde{v}_{j^*}| > 1$.
2. Let $R_1 = s$, $s > n-1$ and there exist a basis $\{1, \ldots, n-1\}$ and number j_1 such that $|\tilde{v}_{j_1}| > 1 + \sum_{j=n}^{s-1} |c_{j_1,j}|$. It is necessary to show that there is a basis solution $\tilde{v}$ and number j' such that $|\tilde{v}_{j'}| > 1 + \sum_{j \in NB} |c_{j',j}|$.

We rewrite (2.17) by replacing the basis solution:

$$\hat{A}\left(p^1\tilde{v}_1 + \cdots + p^{n-1}\tilde{v}_{n-1}\right) = \hat{b}.$$

On the left side of the equation, we subtract and add the expression $\hat{A}p^s\lambda_s$, where λ_s is some parameter:

$$\hat{A}\left(p^1\tilde{v}_1 + \cdots + p^{n-1}\tilde{v}_{n-1} - p^s\lambda_s + p^s\lambda_s\right) = \hat{b}.$$

Now for the subtractable part, let's represent p^s as a linear combination of columns $p^1, \ldots, p^{n-1}$ according to (2.16):

$$\hat{A}\left(p^1(\tilde{v}_1 + c_{1,s}\lambda_s) + \cdots + p^{n-1}(\tilde{v}_{n-1} + c_{n-1,s}\lambda_s) + p^s\lambda_s\right) = \hat{b}.$$

Thus, we obtain a new solution of system (2.17), depending on the parameter λ_s:

$$\hat{v}_j = \begin{cases} \tilde{v}_j + c_{j,s}\lambda_s, & j = \overline{1,n-1}, \\ 0, & j = \overline{n,s-1}, \\ \lambda_s, & j = s. \end{cases}$$

We will consider λ_s in the range from -1 to 1.

By the problem condition, there exists a number j_1 that $|\tilde{v}_{j_1}| > \mu$, where $\mu = 1 + \sum_{j=n}^{s-1} |c_{j_2,j}|$. Let's consider what happens to $\hat{v}_{j_1}$ when λ_s is varied. Without limiting the generality of the foregoing, we assume that $|\hat{v}_{j_1}|$ increases as λ_s increases, i. e. $|\tilde{v}_{j_1} + c_{j_1,s}| = |\tilde{v}_{j_1}| + |c_{j_1,s}|$. Then two variants are possible:

a) When λ_s is reduced to -1, the absolute value of $\hat{v}_j$ remains greater than μ. Hence, $|\tilde{v}_{j_1} - c_{j_1,s}| = |\tilde{v}_{j_1}| - |c_{j_1,s}| > 1 + \sum_{j=n}^{s-1} |c_{j_1,j}|$. In that case the lemma is proved and $j' = j_1$;

b) There is a number λ_s^1 within $(-1,\, 0)$ that $|\tilde{v}_{j_1} + \lambda_s^1 c_{j_1,s}| = \mu$. In this case, by the lemma condition when $\lambda_s = \lambda_s^1$ there exists a number j_2 for which it is true that $|\tilde{v}_{j_2} + \lambda_s^1 c_{j_2,s}| > \mu$. Then we look in which direction $|\hat{v}_{j_2}|$ changes as λ_s approaches -1. If it decreases, then by analogy with j_1 either $j' = j_2$ or there is a number λ_s^2 $(-1 < \lambda_s^2 < \lambda_s^1)$ for which the equation $|\tilde{v}_{j_2} + \lambda_s^2 c_{j_2,s}| = \mu$

holds. There is a number j_3 such that $|\tilde{v}_{j_3} + \lambda_s^2 c_{j_3,s}| > \mu$. Continuing by the same logic, for some number j_k it turns out that $|\tilde{v}_{j_k} - c_{j_k,s}| > \mu$, hence $j' = j_k$.

If $|\hat{v}_{j_2}|$ increases as λ_s decreases, i. e. $|\tilde{v}_{j_2} - c_{j_2,s}| = |\tilde{v}_{j_2}| + |c_{j_2,s}|$, then given the continuous dependence of $\hat{v}_{j_1}$ and $\hat{v}_{j_2}$ on λ_s we can determine a small positive number ε, for which the next system is valid

$$\begin{cases} \left|\tilde{v}_{j_1} + (\lambda_s^1 + \varepsilon)c_{j_1,s}\right| = |\tilde{v}_{j_1}| + (\lambda_s^1 + \varepsilon)|c_{j_1,s}| > \mu, \\ \left|\tilde{v}_{j_2} + (\lambda_s^1 + \varepsilon)c_{j_2,s}\right| = |\tilde{v}_{j_2}| - (\lambda_s^1 + \varepsilon)|c_{j_2,s}| > \mu. \end{cases}$$

Then we multiply the second inequality by $\left|\frac{c_{j_1,s}}{c_{j_2,s}}\right|$ and add to the first one:

$$|\tilde{v}_{j_1}| + \left|\frac{c_{j_1,s}}{c_{j_2,s}}\right| |\tilde{v}_{j_2}| > \mu \left(1 + \left|\frac{c_{j_1,s}}{c_{j_2,s}}\right|\right). \tag{2.18}$$

Now we need to rebuild the basis, i. e., add the column p^s instead of p^{j_2}:

$$B_s = B\backslash\{j_2\} \cup \{s\}, \quad NB_s = NB \cup \{j_2\}\backslash\{s\}.$$

To do this, we need to find λ_s such that $\hat{v}_{j_2}$ is zero. The new basis solution can be expressed through the old one as follows:

$$\tilde{v}^{(s)} = \begin{cases} \tilde{v}_j - \frac{c_{j,s}}{c_{j_2,s}}\tilde{v}_{j_2}, & j \in B_s\backslash\{s\}, \\ -\frac{1}{c_{j_2,j}}\tilde{v}_{j_2}, & j = s, \\ 0, & j = j \in NB_s. \end{cases}$$

The representation of the non-basic columns through the basis columns will also change:

$$\sum_{k \in B_1} c_{k,j}^{(s)} p^k + p^j = 0 \quad \forall j \in NB_1,$$

where

$$c_{k,j}^{(s)} = \begin{cases} c_{k,j} - c_{j_2,j}\frac{c_{k,s}}{c_{j_2,s}}, & \forall k \in B_s\backslash\{s\} & \forall j \in NB_s\backslash\{j_2\}, \\ -c_{j_2,j}\frac{1}{c_{j_2,s}}, & k = s, & \forall j \in NB_s\backslash\{j_2\}, \\ \frac{c_{k,s}}{c_{j_2,s}}, & \forall k \in B_s\backslash\{s\}, & j = j_2, \\ \frac{1}{c_{j_2,s}}, & k = s, & j = j_2. \end{cases}$$

Consequently, it must be shown: in the basis solution $\hat{v}_j$ there exists a number j' such that $|\tilde{v}_{j'}^{(s)}| > 1 + \sum_{j \in NB_s} \left| c_{j',j}^{(s)} \right|$.

From the earlier assumptions we know that $\tilde{v}_{j_1} c_{j_1,s} > 0$ and $\tilde{v}_{j_2} c_{j_2,s} < 0$. From this we can conclude that $\tilde{v}_{j_1} \left(\frac{c_{j_1,s}}{c_{j_2,s}} \tilde{v}_{j_2} \right) < 0$, and therefore $|\tilde{v}_{j_1}^{(s)}| = \left| \tilde{v}_{j_1} - \frac{c_{j_1,s}}{c_{j_2,s}} \tilde{v}_{j_2} \right| = |\tilde{v}_{j_1}| + \left| \frac{c_{j_1,s}}{c_{j_2,s}} \tilde{v}_{j_2} \right|$. Using (2.18), we obtain the inequality

$$|\tilde{v}_{j_1}^{(s)}| > \left(1 + \sum_{j=n}^{s-1} |c_{j_2,j}| \right) \left(1 + \left| \frac{c_{j_1,s}}{c_{j_2,s}} \right| \right) > 1 + \left| \frac{c_{j_1,s}}{c_{j_2,s}} \right| + \sum_{j=n}^{s-1} \left| c_{j_2,j} \frac{c_{j_1,s}}{c_{j_2,s}} \right|,$$

or in other notations:

$$|\tilde{v}_{j_1}^{(s)}| > 1 + \left| c_{j_1,j_2}^{(s)} \right| + \sum_{j=n}^{s-1} \left| c_{j_1,j}^{(s)} \right| = \sum_{j \in NB_s} \left| c_{j_1,j}^{(s)} \right|.$$

In this case $j' = j_1$. Thus, the lemma is proved, and we can return to the proof of the main theorem. □

For simplicity, we define $\tilde{v}_1, \ldots, \tilde{v}_{n-1}$ as the basis variables. Also we assume $j' = 1$, and re-define $c_{j',j}$ as c_j. Then we can refine the vector $\hat{v}$:

$$\begin{cases} \hat{v}_1 > 1 + \sum_{j=n}^{R_1} |c_j|, \\ \hat{v}_j = 0, \ j = \overline{n, R_1}. \end{cases}$$

IV. Let's show that among the rows of the matrix A there is a row a^* such that

1. $a^* p^2 = a^* p^3 = \cdots = a^* p^{n-1} = 0$.
2. $\operatorname{sgn}(a^* p^1) = \operatorname{sgn}(\hat{v}_1)$.
3. $(a^* p^j)(\bar{a} p^j) \geqslant 0 \quad \forall j = \overline{1,R}$.

It is known that $\operatorname{rank}\{p^1, p^2, \ldots p^{n-1}\} = n - 1$. Since $\operatorname{rank} P = n$, there is a column p^s in this matrix such that $\operatorname{rank}\{p^1, p^2, \ldots p^{n-1}, p^s\} = n$, and then $\operatorname{rank}\{p^2, \ldots p^{n-1}, p^s\} = n - 1$.

By construction, there exists a row a in the matrix A such that $ap^2 = ap^3 = \cdots = ap^{n-1} = ap^s = 0$, but $ap^1 \neq 0$.

Due to the fact that the inequalities $-\beta_l \leqslant a_l y \leqslant \beta_l$ are invariant with respect to zero, one can replace the row a_l with the row $-a_l$ in the matrix A. Consequently, it is possible to achieve $ap^1 \hat{v}_1 > 0$.

Thus we have shown the existence of the vector a satisfying requirements 1 and 2. The fulfillment of requirement 3 will be shown by induction.

Base case. For $j = \overline{1, n-1}$ it is obvious that $\bar{a}p^j = 0$.

Induction step. Let $(ap^j)(\bar{a}p^j) \geqslant 0$ for $j = \overline{1,k-1}$, but $(ap^k)(\bar{a}p^k) < 0$. Then there exist two non-negative numbers γ_1 and γ_2 for which $\gamma_1 ap^k + \gamma_2 \bar{a}p^k = 0$ holds. Let's introduce the row $a^* = \gamma_1 a + \gamma_2 \bar{a}$, in which case $a^*p^k = 0$. Since $\bar{a}$ is orthogonal with vectors $p^1, p^2, \ldots, p^{n-1}$ and, in turn, a is orthogonal with vectors $p^2, \ldots, p^{n-1}, p^s$, it is true that

$$a^*p^j = (a + \bar{a})p^j = 0 \quad \forall j = \overline{2, n-1}.$$

Thus $a^*p^1\hat{v}_1 = ap^1\hat{v}_1 > 0$. This means that the row a^* satisfies requirements 1 and 2.

In addition, in the case of $\forall j = \overline{1,k}$

$$(a^*p^j)(\bar{a}p^j) = (\gamma_1 ap^j + \gamma_2 \bar{a}p^j)(\bar{a}p^j) = \gamma_1 (ap^j)(\bar{a}p^j) + \gamma_2 (\bar{a}p^j)(\bar{a}p^j) \geqslant 0.$$

It remains to show that a^* enters the matrix A. In the matrix A there exists a row a_l, constructed as a solution of the system of $n-1$ equations:

$$a_l p^2 = \cdots = a_l p^{n-1} = a_l p^k = 0.$$

Since the solution of this system is a one-parameter family, and a^* is its solution, then a_l and a^* coincide up to the multiplier. Since the multiplier does not affect the properties (and the row can always be multiplied by -1), we can say that a^* is a row of the matrix A.

Thus, by induction it is shown that there is a row a^* that satisfies the three requirements.

V. Let's find what $\beta^* = \max\limits_{v \in \mathbb{V}} (a^* P v)$ equals. Earlier we defined how the non-basic columns are expressed through basis columns (2.16). Let's take advantage of the fact that $a^*p^j = 0$ for $j = \overline{2, n-1}$. By multiplying equations (2.16) by a^* on the left side, we obtain the equations

$$a^*p^j = -c_j a^*p^1, \ \forall j = \overline{n, R_1}.$$

Then

$$\beta^* = a^* p^1 \operatorname{sgn}\left(a^* p^1\right) + \sum_{j=n}^{R_1} \left(-c_j a^* p^1\right) \operatorname{sgn}\left(-c_j a^* p^1\right) + \sum_{j=R_1+1}^{R} \left|a^* p^j\right| =$$

$$= \left|a^* p^1\right| \left(1 + \sum_{j=n}^{R_1} |c_j|\right) + \sum_{j=R_1+1}^{R} \left|a^* p^j\right|.$$

Now let's find the value of $a^* P\tilde{v}$. Since $a^* p^j = 0,\ j = \overline{2,n-1}$, $\hat{v}_j = 0$, $j = \overline{n,R_1}$, and $\hat{v}_j = \operatorname{sgn}(\bar{a} p^j),\ j = \overline{R_1+1,R}$, we get the expression

$$a^* P\hat{v} = a^* p^1 \hat{v}_1 + \sum_{j=R_1+1}^{R} a^* p^j \operatorname{sgn}(\bar{a} p^j).$$

Since $(a^* p^j)(\bar{a} p^j) > 0$, then $\operatorname{sgn}(a^* p^j) = \operatorname{sgn}(\bar{a} p^j)$. Let's rewrite the previous equation taking into account that $a^* P v_1 > 0$:

$$a^* P\hat{v} = \left|a^* p^1\right| |\hat{v}_1| + \sum_{j=R_1+1}^{R} \left|a^* p^j\right|.$$

Finally, let's estimate the expression $a^* P\hat{v} - \beta^*$:

$$a^* P\hat{v} - \beta^* = \left|a^* p^1\right| \left(|\hat{v}_1| - 1 - \sum_{j=n}^{R_1} |c_j|\right) > 0.$$

Hence, $a^* P\hat{v} > \beta^*$, and since $\hat{y} = P\hat{v}$, $a^*\hat{y} > \beta^*$. Thus, a contradiction is shown: $\hat{y} \notin \bar{\mathbb{Y}}$. So there exists $\hat{v}$ such that $\hat{y} = P\hat{v}$ and $\hat{v} \in \mathbb{V}$.

VI. It has been shown that the vertices of the polyhedron $\bar{\mathbb{Y}}$ belong to the set $\mathbb{Y}$, denote the vertices as $y^1, \ldots, y^h$.

It is known that a polyhedron can be described as a convex hull of its vertices, i. e. $y \in \bar{\mathbb{Y}}$ if and only if $y = \sum_{i=1}^{h} \lambda_i y^i$, where $\sum_{i=1}^{h} \lambda_i = 1, \lambda_i \geqslant 0$. Since $y^i \in \mathbb{Y}$, there exists a vector $v^i \in \mathbb{V}$ for which $y^i = P v^i$. So, $y = \sum_{i=1}^{h} \lambda_i y^i = \sum_{i=1}^{h} \lambda_i P v^i = P v$.

Since $\mathbb{V}$ is a convex set, the convex hull of points belonging to it is part of it. Hence, $v = \sum_{i=1}^{h} v^i \in \mathbb{V}$, so $y \in \mathbb{Y}$. As a result, it is shown that if $y \in \bar{\mathbb{Y}}$, then $y \in \mathbb{Y}$. $\square$

2.7 Algorithm Analysis

As a result, the algorithm for constructing the reachability (controllability) set of problem (2.8) consists of the following steps:

1. Transition from the construction problem of the set $X_1(t, \hat{t}, \hat{x})$ or $X_2(t, \hat{t}, \hat{x})$ to linear mapping problem of R-dimensional cube to n-dimensional space (2.8).
2. Excluding linearly dependent equations from the system $y = Pv$. To do this, we can use the Gauss method. If $\operatorname{rank} P = n_1$, the result will be $n - n_1$ linear equations with respect to y.
3. Iterating over all sets of $n_1 - 1$ linearly independent columns of the matrix P. For each such set of columns, a vector-row $\tilde{a}$ is calculated as a non-zero solution of a homogeneous system of linear algebraic equations (SLAE) with a transpose matrix consisting of columns (2.14). To find $\tilde{a}$, we can use the Gauss method or any other method for solving SLAEs.
4. Calculating $\tilde{\beta}$ for rows $\tilde{a}$ (2.15). Then matrix A, consisting of all found $\tilde{a}$, and corresponding vector b of numbers $\tilde{\beta}$ are formed. The system of inequalities $-b \leqslant A \leqslant b$ is the result of mapping (2.8).
5. Return from variables y to x. As a result, reachability set $X_1(t, \hat{t}, \hat{x})$ or controllability set $X_2(t, \hat{t}, \hat{x})$ are described as a system of n_1 linear equations and series of two-sided linear inequalities.

The maximal possible number of rows in matrix A is the number of combinations $C_R^{n_1}$. One can find the rows $\tilde{a}$ as a solution of system (2.14) or as a generalization of the vector product on the n_1-dimensional space. For example, for three-dimensional space, the row $\tilde{a}$ for columns p^{j_1} and p^{j_2} can be found

as the determinant:

$$\tilde{a} = \begin{vmatrix} \vec{i} & p_1^{j_1} & p_1^{j_2} \\ \vec{j} & p_2^{j_1} & p_2^{j_2} \\ \vec{k} & p_3^{j_1} & p_3^{j_2} \end{vmatrix}.$$

Thus, the problem can be reduced to the calculation of $C_R^{n_1}$ n_1-dimensional determinants.

2.8 Chapter 2 Conclusion

An algorithm for constructing sets of reachability and controllability for a linear control problem with bilateral constraints is proposed. The algorithm consists in the transition from problem (1.1)–(1.4) to system (2.8), the construction of a system of linear inequalities with respect to y, and the transition back to the variables $x(t)$. As a result, the reachability (controllability) set given by the system of linear constraints is found.

Besides the direct use of these sets, there are two other useful applications:

1. In a control problem with an external perturbation that can make major corrections to the object's motion, and the control is therefore periodically real-time rebuilt [7], it may be useful to assess at each realignment whether the desired position of the system is reachable.
2. Non-linear optimal control problems can be solved by system linearizations. Thus, in Chapter 4, we will describe a control construction algorithm for non-linear systems, which consists of a consecutive system linearization and construction of control for the linear system. However, due to the inaccuracy of linearizations, it may turn out that the desired final state is not reachable. The possibility of this problem can be minimized by assessing, at each step, how far the final state is from the boundary of the reachability set.

CHAPTER 3. CONSTRUCTION OF OPTIMAL CONTROL SUBJECT TO CONSTRAINTS

In this chapter we set a problem to study the possibility of taking into account additional constraints on control and phase variables for the linear terminal problem (1.1)–(1.3) with the linear objective function (1.5) or with the quadratic functional (1.7).

In Paragraph 3.1, the question of choosing a class of admissible controls will be considered in detail. In addition to the class of piecewise constant functions (1.4), which was chosen as the main one for this dissertation work, controls in the form of piecewise linear and piecewise quadratic functions are also of interest. The use of more general classes leads to a complication of the problem solution, but for all that, additional conditions can be imposed on continuity and smoothness. For example, for the problem of controlling wheeled mobile robots, which are used to transport people with disabilities, the smoothness of control actions is the most important criterion [72].

In [10] a control method in the class of quadratic splines was proposed. The results presented in Paragraph 3.1 are a follow-up of the outlined approach.

In Paragraph 3.2 the control problem with additional constraints on controls and phase variables will be studied. First, linear constraints on controls will be considered, then the question of constructing controls for non-convex sets of admissible controls will be considered. Further the problem with constraints on phase variables will be covered. Note that the problem of control construction under convex linear constraints on phase variables at given moments of time was studied in [73].

3.1 Constraints on Class of Controlling Functions

In this paragraph, we consider the problem of controlling a linear system under various constraints on the control class. Three classes are of the greatest interest:

1. Control as a piecewise constant function.
2. Control as a piecewise linear function.
3. Control as a piecewise quadratic function.

The choice of such classes is caused by the convenience of solving the problem while preserving the efficiency. At the same time, consideration of control in one of the given classes makes it possible to achieve piecewise continuity, continuity or smoothness of controls, which is often an important requirement resulting from the physical properties of the processes under study.

Thus, the selection of controls in the first class allows to achieve piecewise continuity, in the second class — piecewise continuity or continuity, in the third class — piecewise continuity, continuity or smoothness.

The class of piecewise constant functions (1.4) was covered in detail in Chapter 1. Now we will focus on the alternative choices of the control functions.

3.1.1 Linear Problem with Piecewise Linear Control

Let's consider the case in which the components of the control vector are piecewise linear functions:

$$u_i(t) = \begin{cases} p_{ik}^{(1)} t + p_{ik}^{(0)}, & t \in [t_{k-1},\ t_k), \\ k = \overline{1, N}, \end{cases} \qquad i = \overline{1,r}. \tag{3.1}$$

where $t_k = kh$.

We will show an algorithm for reducing the linear optimal control problem (1.1)–(1.3), (3.1) with the objective function (1.5) or (1.7) to a mathematical programming problem.

Continuity Conditions Reduction

Since the controls are represented as piecewise continuous functions, discontinuities are possible only at knots. Therefore it is necessary to add continuity conditions at the junctions of time segments:

$$p_{ik}^{(1)} t_k + p_{ik}^{(0)} = p_{ik+1}^{(1)} t_k + p_{ik+1}^{(0)}, \quad k = \overline{1,N-1}, \quad i = \overline{1,r}. \tag{3.2}$$

Thus, the continuity conditions have been reduced to $r(N-1)$ linear equations with respect to the variables $p_{ik}^{(0)}$ and $p_{ik}^{(1)}$.

If the problem does not require continuity in the knots, it is worth disabling the constraints (3.2).

Terminal Conditions Reduction

The conditions for the initial state of the object are always satisfied since they are part of the Cauchy problem. Let's define the conditions for the final state.

We write out the Cauchy formula for the system (1.1) at the final moment of time:

$$x(T) = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)\left(B(t)u(t) + d(t)\right)dt.$$

Let's divide the time segment $[0,\ T]$ into N segments of equal length and apply the representation of the control in the form of piecewise linear functions (3.1):

$$x(T) = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)d(t)dt +$$

$$+\sum_{i=1}^{r}\sum_{k=1}^{N}\left[\left(\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)tdt\right)p_{ik}^{(1)} + \left(\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i dt\right)p_{ik}^{(0)}\right].$$

Section 3.1.3 will address the issue of calculating integrals of the form

$$\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i dt, \quad \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i t dt,$$

but in the meantime let's re-define them, having in mind that they are constants:

$$\begin{aligned} b^{i,k,0} &:= \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt, \\ b^{i,k,1} &:= \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)tdt, \end{aligned} \qquad k = \overline{1,N}, \quad i = \overline{1,r},$$

In addition, we introduce a notation:

$$d^0 = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)d(t)dt.$$

Then we can write out the representation of $x(T)$ as a linear function of the variables $p_{ik}^{(0)}$ and $p_{ik}^{(1)}$:

$$x(T) = d^0 + \sum_{i=1}^{r}\sum_{k=1}^{N}\left(b^{i,k,0}p_{ik}^{(0)} + b^{i,k,1}p_{ik}^{(1)}\right). \tag{3.3}$$

Let's substitute the expression (3.3) into the boundary condition (1.2):

$$Hd^0 + \sum_{i=1}^{r}\sum_{k=1}^{N}\left(Hb^{i,k,0}p_{ik}^{(0)} + Hb^{i,k,1}p_{ik}^{(1)}\right) = g^0,$$

and introduce a series of new notations:

$$\begin{aligned} h^{i,k,0} &= Hb^{i,k,0}, \\ h^{i,k,1} &= Hb^{i,k,1}, \end{aligned} \qquad k = \overline{1,N}, \quad i = \overline{1,r},$$

$$g = g^0 - Hd^0.$$

Now we write out the resulting conditions:

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(h_l^{i,k,0}p_{ik}^{(0)}+h_l^{i,k,1}p_{ik}^{(1)}\right)=g_l,\quad l=\overline{1,m}. \tag{3.4}$$

Thus, the terminal conditions (1.2) have turned into m linear equations (3.4) with $2rN$ variables.

Reduction of Direct Constraints on Controls

According to the direct constraints (1.3) and the representation of the control functions in the form (3.1), the following conditions must be met:

$$l_i^{(1)}\leqslant p_{ik}^{(1)}t+p_{ik}^{(0)}\leqslant l_i^{(2)}\quad\forall t\in[t_{k-1},\ t_k),\quad k=\overline{1,N},\quad i=\overline{1,r}.$$

Since linear functions are monotonic, it is sufficient to consider these inequalities at the ends of the considered time intervals:

$$\begin{array}{l} l_i^{(1)}\leqslant p_{ik}^{(1)}t_{k-1}+p_{ik}^{(0)}\leqslant l_i^{(2)},\\ l_i^{(1)}\leqslant p_{ik}^{(1)}t_k+p_{ik}^{(0)}\leqslant l_i^{(2)},\end{array}\quad k=\overline{1,N},\quad i=\overline{1,r}. \tag{3.5}$$

Hence, the direct constraints (1.3) are reduced to $4rN$ linear inequalities (3.5) with respect to $2rN$ variables $p_{ik}^{(0)}$ and $p_{ik}^{(1)}$.

Objective Functions Reduction

In order to reduce the first objective function, we use the representation of the final state of the object (3.3). Then we can write the criterion (1.5) as a linear function of the variables $p_{ik}^{(0)}$ and $p_{ik}^{(1)}$:

$$J_1=c^Td^0+\sum_{i=1}^{r}\sum_{k=1}^{N}\left(c^Tb^{i,k,0}p_{ik}^{(0)}+c^Tb^{i,k,1}p_{ik}^{(1)}\right).$$

Since the first item is a constant, we can remove it from the objective function, keeping in mind that this may change the value of the functional:

$$J_1 = \sum_{i=1}^{r} \sum_{k=1}^{N} \left(c^{i,k,0} p_{ik}^{(0)} + c^{i,k,1} p_{ik}^{(1)} \right) \longrightarrow \min . \tag{3.6}$$

In (3.6) we introduced notations

$$\begin{aligned} c^{i,k,0} &= c^T b^{i,k,0}, \\ c^{i,k,1} &= c^T b^{i,k,1}, \end{aligned} \qquad k = \overline{1,N}, \quad i = \overline{1,r}.$$

Thus, (3.6) is a linear functional with respect to the variables $p_{ik}^{(0)}$ and $p_{ik}^{(1)}$.

The second criterion (1.6) will not be considered for this type of control because finding the modulus from the piecewise linear function requires additional calculations that are beyond the scope of this paper.

Now let's proceed to the third criterion. We replace the control $u(t)$ by a piecewise linear function (3.1) in the objective function (1.7):

$$J_2 = \sum_{k=1}^{N} \sum_{i_1=1}^{r} \sum_{i_2=1}^{r} \int_{t_{k-1}}^{t_k} q_{i_1 i_2} \left(p_{i_1 k}^{(1)} t + p_{i_1 k}^{(0)} \right) \left(p_{i_2 k}^{(1)} t + p_{i_2 k}^{(0)} \right) dt.$$

After calculating integrals and introducing new notations, we obtain a quadratic form with respect to the variables $p_{ik}^{(0)}$ and $p_{ik}^{(1)}$:

$$\begin{aligned} J_2 = \sum_{k=1}^{N} \sum_{i_1=1}^{r} \sum_{i_2=1}^{r} \Big(& h_{i_1 i_2 k}^{(1)} p_{i_1 k}^{(0)} p_{i_2 k}^{(0)} + \\ & + h_{i_1 i_2 k}^{(2)} \left(p_{i_1 k}^{(1)} p_{i_2 k}^{(0)} + p_{i_1 k}^{(0)} p_{i_2 k}^{(1)} \right) + h_{i_1 i_2 k}^{(3)} p_{i_1 k}^{(1)} p_{i_2 k}^{(1)} \Big), \end{aligned} \tag{3.7}$$

where

$$\begin{gathered} h_{i_1 i_2 k}^{(\alpha)} := q_{i_1 i_2} \frac{1}{\alpha} \frac{T^\alpha}{N^\alpha} \left(k^\alpha - (k-1)^\alpha \right), \\ k = \overline{1,N}, \quad i_1 = \overline{1,r}, \quad i_2 = \overline{1,r}, \quad \alpha = \overline{1,3}. \end{gathered}$$

In Section 3.1.2 we will show that the constructed quadratic form is non-negative definite.

Results

It was shown how by choosing the control as a piecewise linear function, the linear optimal control problem (1.1)–(1.3), (3.1), (1.5) can be reduced to a linear programming problem (3.2), (3.4)–(3.6):

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(c^{i,k,0}p_{ik}^{(0)}+c^{i,k,1}p_{ik}^{(1)}\right)\longrightarrow\min$$

$$p_{ik}^{(1)}t_k+p_{ik}^{(0)}=p_{ik+1}^{(1)}t_k+p_{ik+1}^{(0)},\quad k=\overline{1,N-1},\quad i=\overline{1,r}$$

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(h_l^{i,k,0}p_{ik}^{(0)}+h_l^{i,k,1}p_{ik}^{(1)}\right)=g_l,\quad l=\overline{1,m}$$

$$\begin{matrix}l_i^{(1)}\leqslant p_{ik}^{(1)}t_{k-1}+p_{ik}^{(0)}\leqslant l_i^{(2)},\\ l_i^{(1)}\leqslant p_{ik}^{(1)}t_k+p_{ik}^{(0)}\leqslant l_i^{(2)},\end{matrix}\quad k=\overline{1,N},\quad i=\overline{1,r}.$$

This problem has the following characteristics:

– $2rN$ variables,

– $r(N-1)+m$ linear equations,

– $4rN$ linear inequalities,

– linear objective function.

It was also shown how the optimal control problem with a linear system and a quadratic functional (1.1)–(1.3), (3.1), (1.7) can be reduced to a convex quadratic programming problem (3.2), (3.4), (3.5), (3.7):

$$\sum_{k=1}^{N}\sum_{i_1=1}^{r}\sum_{i_2=1}^{r}\left(h_{i_1i_2k}^{(1)}p_{i_1k}^{(0)}p_{i_2k}^{(0)}+h_{i_1i_2k}^{(2)}\left(p_{i_1k}^{(1)}p_{i_2k}^{(0)}+p_{i_1k}^{(0)}p_{i_2k}^{(1)}\right)+h_{i_1i_2k}^{(3)}p_{i_1k}^{(1)}p_{i_2k}^{(1)}\right)\to\min$$

$$p_{ik}^{(1)}t_k+p_{ik}^{(0)}=p_{ik+1}^{(1)}t_k+p_{ik+1}^{(0)},\quad k=\overline{1,N-1},\quad i=\overline{1,r}$$

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(h_l^{i,k,0}p_{ik}^{(0)}+h_l^{i,k,1}p_{ik}^{(1)}\right)=g_l,\quad l=\overline{1,m}$$

$$\begin{matrix}l_i^{(1)}\leqslant p_{ik}^{(1)}t_{k-1}+p_{ik}^{(0)}\leqslant l_i^{(2)},\\ l_i^{(1)}\leqslant p_{ik}^{(1)}t_k+p_{ik}^{(0)}\leqslant l_i^{(2)},\end{matrix}\quad k=\overline{1,N},\quad i=\overline{1,r}.$$

This problem has the following characteristics:

– $2rN$ variables,

– $r(N-1)+m$ linear equations,

– $4rN$ linear inequalities,

– convex quadratic objective function.

Example. Damping of Sleeping Lagrange Top

Let's solve the damping of the sleeping Lagrange top, described in Paragraph 1.5, by constructing a control in the class of piecewise linear functions.

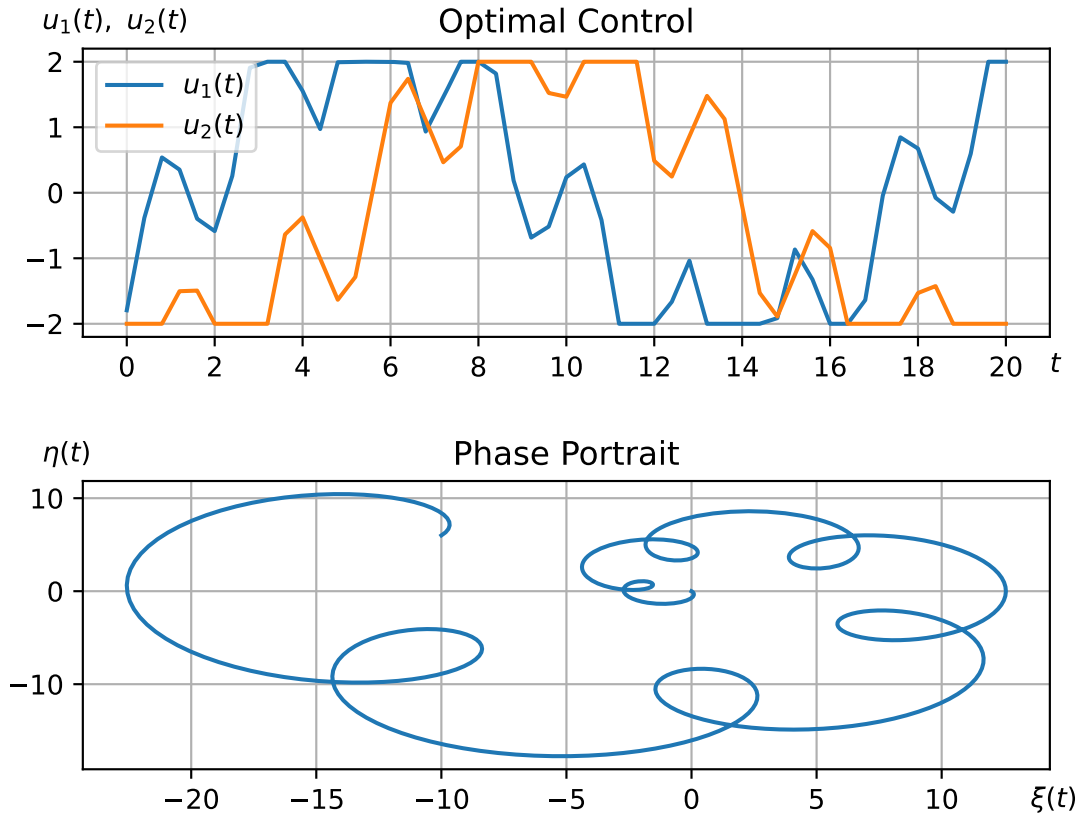

Figure 3.1 — Dumping of Sleeping Lagrange Top in Case of Control from Class of Piecewise Linear Functions

The reduction procedure was implemented in Python 3, and the quadratic programming problem was solved using the COPT optimization package (see Appendix A).

The total running time of the program was 0.57 seconds. The value of the objective function is equal to 91.47. The results are shown in Fig. 3.1.

3.1.2 Linear Problem with Piecewise Square Control

Now suppose that the components of the control vector are piecewise quadratic functions or, what is the same, quadratic splines:

$$u_i(t) = \begin{cases} p_{ik}^{(2)} t^2 + p_{ik}^{(1)} t + p_{ik}^{(0)}, \quad t \in [t_{k-1},\ t_k), \\ k = \overline{1, N}, \end{cases} \qquad i = \overline{1,r}. \tag{3.8}$$

where $t_k = kh$.

Let's show an algorithm for reducing the optimal control problem (1.1)–(1.3), (3.8) with the objective function (1.5) or (1.7) to a mathematical programming problem.

Smoothness Conditions Reduction

Since the controls are represented as quadratic splines, smoothness between knots is guaranteed. However, it is necessary to additionally ensure continuity and smoothness at the knots (the transition points from one quadratic function to another).

Continuity in knots for the control functions:

$$\begin{gathered} p_{ik}^{(2)} t_k^2 + p_{ik}^{(1)} t_k + p_{ik}^{(0)} = p_{ik+1}^{(2)} t_k^2 + p_{ik+1}^{(1)} t_k + p_{ik+1}^{(0)}, \\ k = \overline{1,N-1}, \quad i = \overline{1,r}. \end{gathered} \tag{3.9}$$

Continuity in knots for the derivatives of the control functions:

$$2p_{ik}^{(2)}t_k + p_{ik}^{(1)} = 2p_{ik+1}^{(2)}t_k + p_{ik+1}^{(1)}, \quad k = \overline{1,N-1}, \quad i = \overline{1,r}. \tag{3.10}$$

Thus, the smoothness conditions have been reduced to $2r(N-1)$ linear equations with respect to the variables $p^{(0)}$, $p^{(1)}$, $p^{(2)}$.

Note that if the problem does not require smoothness at the knots, we can disable the constraints (3.9). If continuity is also not required, we should also disable the constraints (3.10).

Terminal Conditions Reduction

The conditions for the initial state of the object are always satisfied since they are part of the Cauchy problem. Let's define the conditions on the final state.

We write out the Cauchy formula for the system (1.1) at the final moment of time:

$$x(T) = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)\left(B(t)u(t) + d(t)\right)dt.$$

Let's divide the time segment $[0,\ T]$ into N segments of equal length and apply the control representation in the form of quadratic splines (3.8):

$$x(T) = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)d(t)dt +$$

$$+\sum_{i=1}^{r}\sum_{k=1}^{N}\left[\left(\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)t^2dt\right)p_{ik}^{(2)} + \right.$$

$$\left. + \left(\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)tdt\right)p_{ik}^{(1)} + \left(\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt\right)p_{ik}^{(0)}\right].$$

Section 3.1.3 will address the issue of calculating integrals of the form

$$\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt, \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)tdt, \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)t^2dt,$$

but in the meantime let's re-define them, given that they are constants:

$$\begin{aligned} b^{i,k,0} &:= \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt, \\ b^{i,k,1} &:= \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)tdt, \qquad k = \overline{1,N}, \quad i = \overline{1,r}, \\ b^{i,k,2} &:= \int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)t^2dt, \end{aligned}$$

Similar to 3.1.1, we introduce the parameter d^0:

$$d^0 = Y(T)x_* + \int_0^T Y(T)Y^{-1}(t)d(t)dt.$$

After that we can write out the representation of $x(T)$ as a linear function of the variables $p^{(0)},\ p^{(1)},\ p^{(2)}$:

$$x(T) = d^0 + \sum_{i=1}^{r}\sum_{k=1}^{N}\left(b^{i,k,0}p_{ik}^{(0)} + b^{i,k,1}p_{ik}^{(1)} + b^{i,k,2}p_{ik}^{(2)}\right). \tag{3.11}$$

Now substitute the expression (3.11) into the boundary condition (1.2):

$$Hd^0 + \sum_{i=1}^{r}\sum_{k=1}^{N}\left(Hb^{i,k,0}p_{ik}^{(0)} + Hb^{i,k,1}p_{ik}^{(1)} + Hb^{i,k,2}p_{ik}^{(2)}\right) = g^0,$$

and introduce a series of new notations:

$$\begin{aligned} h^{i,k,0} &= Hb^{i,k,0}, \\ h^{i,k,1} &= Hb^{i,k,1}, \quad k = \overline{1,N}, \quad i = \overline{1,r}, \\ h^{i,k,2} &= Hb^{i,k,2}, \\ g &= g^0 - Hd^0. \end{aligned}$$

Let's write out the obtained conditions:

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(h_l^{i,k,0}p_{ik}^{(0)} + h_l^{i,k,1}p_{ik}^{(1)} + h_l^{i,k,2}p_{ik}^{(2)}\right) = g_l, \quad l = \overline{1,m}. \tag{3.12}$$

Thus, the terminal conditions (1.2) have turned into m linear equations with $3rN$ variables (3.12).

Reduction of Direct Constraints on Controls

According to the direct constraints (1.3) and the representation of the control functions in the form (3.9), the following constraints must be satisfied:

$$l_i^{(1)} \leqslant p_{ik}^{(2)} t^2 + p_{ik}^{(1)} t + p_{ik}^{(0)} \leqslant l_i^{(2)} \quad \forall t \in [t_{k-1},\ t_k), \quad k = \overline{1,N}, \quad i = \overline{1,r}.$$

They can be replaced by extreme conditions

$$\begin{aligned} l_i^{(1)} &\leqslant \inf_{t\in[t_{k-1},\ t_k)} \left(p_{ik}^{(2)} t^2 + p_{ik}^{(1)} t + p_{ik}^{(0)} \right), \\ l_i^{(2)} &\geqslant \sup_{t\in[t_{k-1},\ t_k)} \left(p_{ik}^{(2)} t^2 + p_{ik}^{(1)} t + p_{ik}^{(0)} \right), \end{aligned} \qquad k = \overline{1,N}, \quad i = \overline{1,r}.$$

In that case, on each of the intervals $[t_{k-1},\ t_k)$ it is necessary to consider only three points suspicious of an extremum: t_{k-1}, t_k and $-p_{ik}^{(1)}/2p_{ik}^{(2)}$.

This leads to a constraint on the knot points:

$$\begin{aligned} l_i^{(1)} &\leqslant p_{ik}^{(2)} t_{k-1}^2 + p_{ik}^{(1)} t_{k-1} + p_{ik}^{(0)} \leqslant l_i^{(2)}, \\ l_i^{(1)} &\leqslant p_{ik}^{(2)} t_k^2 + p_{ik}^{(1)} t_k + p_{ik}^{(0)} \leqslant l_i^{(2)}, \end{aligned} \qquad k = \overline{1,N}, \quad i = \overline{1,r}. \tag{3.13}$$

The situation with the vertex of the parabola $-p_{ik}^{(1)}/2p_{ik}^{(2)}$ is much more complicated. It would be wrong to say that the value of an extremum must satisfy two-sided constraints, since it is permitted to violate the bounds if the extremum argument is to the right or to the left of the segment $[t_{k-1},\ t_k)$.

For simplicity, let's re-define some variables:

$$p := p_{ik}^{(2)}, \quad \tau_1 := t_{k-1}, \quad \tau_2 := t_k, \quad l_1 := l_i^{(1)}, \quad l_2 := l_i^{(2)}.$$

Let's introduce a notation for the values of the quadratic polynomial at the knots:

$$y_1 := p_{ik}^{(2)} t_{k-1}^2 + p_{ik}^{(1)} t_{k-1} + p_{ik}^{(0)}, \quad y_2 := p_{ik}^{(2)} t_k^2 + p_{ik}^{(1)} t_k + p_{ik}^{(0)}.$$

Now we can express the variables $p_{ik}^{(1)}$ and $p_{ik}^{(0)}$ using p, y_1, and y_2:

$$p_{ik}^{(1)} = \frac{y_2 - y_1}{\tau_2 - \tau_1} - p(\tau_1 + \tau_2), \quad p_{ik}^{(0)} = p\tau_1\tau_2 - \frac{y_2 - y_1}{\tau_2 - \tau_1}\tau_1 + y_1.$$

Then we can find the conditions under which the top of the polynomial equals the upper boundary l_2:

$$\left(\frac{y_2 - y_1}{\tau_2 - \tau_1} - p(\tau_1 + \tau_2)\right)^2 - 4p\left(p\tau_1\tau_2 - \frac{y_2 - y_1}{\tau_2 - \tau_1}\tau_1 + y_1 - l_2\right) = 0.$$

After a series of transformations we obtain a quadratic equation with respect to the variable p:

$$(\tau_2 - \tau_1)^2 p^2 + (-2(y_1 + y_2) + 4l_2)\, p + \left(\frac{y_2 - y_1}{\tau_2 - \tau_1}\right)^2 = 0,$$

It has two solutions:

$$p = -\left(\frac{\sqrt{l_2 - y_1} \pm \sqrt{l_2 - y_2}}{\tau_2 - \tau_1}\right)^2.$$

Thus, the following problem was solved: to find the parameters of the quadratic polynomial depending on its values at two given points (y_1 at τ_1 and y_2 at τ_2) and provided that the extremum value is fixed and equal to some predetermined number (l_2). Analyzing the obtained results, we can understand that the solution with the largest absolute value corresponds to a parabola, whose vertex argument is between two given points (within the interval $[\tau_1,\ \tau_2]$). The remaining solution corresponds to a parabola whose vertex argument is to the left or to the right of the given points.

As mentioned earlier, the constraint on the extremum of a quadratic polynomial occurs only if it is located inside the segment in question. Therefore in this case we are interested only in the solution with «plus» sign in the numerator:

$$p = -\left(\frac{\sqrt{l_2 - y_1} + \sqrt{l_2 - y_2}}{\tau_2 - \tau_1}\right)^2.$$

Doing the same for the lower boundary, we find two-sided constraints on p:

$$-\left(\frac{\sqrt{l_2 - y_1} + \sqrt{l_2 - y_2}}{\tau_2 - \tau_1}\right)^2 \leqslant p \leqslant \left(\frac{\sqrt{y_1 - l_1} + \sqrt{y_2 - l_1}}{\tau_2 - \tau_1}\right)^2. \qquad (3.14)$$

Now let's return to the original notations, introduce new variables $s_{ik}^{(1)}$, $s_{ik}^{(2)}$, $s_{ik}^{(3)}$, $s_{ik}^{(4)}$ and add constraints on them:

$$\begin{aligned}
&s_{ik}^{(1)} = p_{ik}^{(2)} t_{k-1}^2 + p_{ik}^{(1)} t_{k-1} + p_{ik}^{(0)} - l_i^{(1)}, \\
&s_{ik}^{(2)} = p_{ik}^{(2)} t_k^2 + p_{ik}^{(1)} t_k + p_{ik}^{(0)} - l_i^{(1)}, \\
&s_{ik}^{(3)} = l_i^{(2)} - p_{ik}^{(2)} t_{k-1}^2 - p_{ik}^{(1)} t_{k-1} - p_{ik}^{(0)}, \qquad k = \overline{1,N}, \quad i = \overline{1,r}. \\
&s_{ik}^{(4)} = l_i^{(2)} - p_{ik}^{(2)} t_k^2 - p_{ik}^{(1)} t_k - p_{ik}^{(0)}, \\
&s_{ik}^{(1)} \geqslant 0, \quad s_{ik}^{(2)} \geqslant 0, \quad s_{ik}^{(3)} \geqslant 0, \quad s_{ik}^{(4)} \geqslant 0,
\end{aligned} \tag{3.15}$$

After that let's rewrite the inequalities (3.14):

$$-\left(\frac{\sqrt{s^{(3)}} + \sqrt{s^{(4)}}}{T/N}\right)^2 \leqslant p_{ik}^{(2)} \leqslant \left(\frac{\sqrt{s^{(1)}} + \sqrt{s^{(2)}}}{T/N}\right)^2, \quad k = \overline{1,N}, \quad i = \overline{1,r}.$$

Then, expanding the brackets, we obtain the following non-linear constraints:

$$\begin{aligned}
&T^2 p_{ik}^{(2)} \leqslant N^2 \left(s_{ik}^{(1)} + s_{ik}^{(2)} + 2\sqrt{s_{ik}^{(1)} s_{ik}^{(2)}}\right), \\
&T^2 p_{ik}^{(2)} \geqslant -N^2 \left(s_{ik}^{(3)} + s_{ik}^{(4)} + 2\sqrt{s_{ik}^{(3)} s_{ik}^{(4)}}\right),
\end{aligned} \qquad k = \overline{1,N}, \quad i = \overline{1,r}. \tag{3.16}$$

Technically, we could have stopped at constraints (3.16), but such non-linear inequalities containing square roots are extremely inconvenient for finding the optimal solution, since they introduce non-convexity into the calculated model and are difficult for solvers to perceive. Therefore we will try to reduce them to convex constraints.

Let's introduce non-negative variables $r_{ik}^{(1)}$ and $r_{ik}^{(2)}$ and express the non-linear part (3.16) through them:

$$\begin{aligned}
&\left(r_{ik}^{(1)}\right)^2 \leqslant 4 s_{ik}^{(1)} s_{ik}^{(2)}, \\
&\left(r_{ik}^{(2)}\right)^2 \leqslant 4 s_{ik}^{(3)} s_{ik}^{(4)}, \qquad k = \overline{1,N}, \quad i = \overline{1,r}. \\
&r_{ik}^{(1)} \geqslant 0, \quad r_{ik}^{(2)} \geqslant 0,
\end{aligned} \tag{3.17}$$

Note that the constraints (3.17) belong to the class of conic equations of second order, we can prove that such constraints have convex set of solutions. This fact greatly simplifies the optimization problem. This result is the main achievement of the current paragraph.

Finally, let's write out the final representation of constraints (3.16):

$$\begin{aligned} T^2 p_{ik}^{(2)} &\leqslant N^2 \left(s_{ik}^{(1)} + s_{ik}^{(2)} + r_{ik}^{(1)} \right), \\ T^2 p_{ik}^{(2)} &\geqslant -N^2 \left(s_{ik}^{(3)} + s_{ik}^{(4)} + r_{ik}^{(2)} \right), \end{aligned} \qquad k = \overline{1,N}, \quad i = \overline{1,r}. \tag{3.18}$$

As a result, it is shown how direct two-sided constraints (1.3) were transformed into linear constraints (3.13), (3.15), (3.18) and quadratic inequalities (3.17).

Objective Functions Reduction

In order to reduce the first objective function, we use the representation of the final state of the object (3.11). Then we can write the criterion (1.5) as a linear function of the variables $p^{(0)},\ p^{(1)},\ p^{(2)}$:

$$J_1 = c^T d^0 + \sum_{i=1}^{r} \sum_{k=1}^{N} \left(c^T b^{i,k,0} p_{ik}^{(0)} + c^T b^{i,k,1} p_{ik}^{(1)} + c^T b^{i,k,2} p_{ik}^{(2)} \right).$$

Since the first item is a constant, we can remove it from the objective function, keeping in mind that this may change the value of the functional:

$$J_1 = \sum_{i=1}^{r} \sum_{k=1}^{N} \left(c^{i,k,0} p_{ik}^{(0)} + c^{i,k,1} p_{ik}^{(1)} + c^{i,k,2} p_{ik}^{(2)} \right) \longrightarrow \min, \tag{3.19}$$

where

$$\begin{aligned} c^{i,k,0} &= c^T b^{i,k,0}, \\ c^{i,k,1} &= c^T b^{i,k,1}, \quad k = \overline{1,N}, \quad i = \overline{1,r}. \\ c^{i,k,2} &= c^T b^{i,k,2}, \end{aligned}$$

Thus, (3.19) is a linear functional with respect to the variables $p^{(0)},\ p^{(1)},\ p^{(2)}$.

The second criterion (1.6) will not be considered for this type of control because finding the modulus from the piecewise quadratic function requires additional calculations that are beyond the scope of this paper.

Let's proceed to the third criterion. We replace the control $u(t)$ by quadratic splines (3.8) in the objective function (1.7):

$$J_2 = \sum_{k=1}^{N} \sum_{i_1=1}^{r} \sum_{i_2=1}^{r} \int_{t_{k-1}}^{t_k} q_{i_1 i_2} \left(p_{i_1 k}^{(2)} t^2 + p_{i_1 k}^{(1)} t + p_{i_1 k}^{(0)} \right) \left(p_{i_2 k}^{(2)} t^2 + p_{i_2 k}^{(1)} t + p_{i_2 k}^{(0)} \right) dt.$$

After calculating integrals and introducing new notations, we obtain a quadratic form with respect to the variables $p_{ik}^{(0)}$, $p_{ik}^{(1)}$ and $p_{ik}^{(2)}$:

$$J_2 = \sum_{k=1}^{N} \sum_{i_1=1}^{r} \sum_{i_2=1}^{r} \sum_{\alpha_1=0}^{2} \sum_{\alpha_2=0}^{2} \left(h_{i_1 i_2 k}^{\alpha_1, \alpha_2} p_{i_1 k}^{(\alpha_1)} p_{i_2 k}^{(\alpha_2)} \right), \tag{3.20}$$

where

$$h_{i_1 i_2 k}^{\alpha_1, \alpha_2} := q_{i_1 i_2} \frac{1}{\beta} \frac{T^\beta}{N^\beta} \left(k^\beta - (k-1)^\beta \right), \quad \beta = \alpha_1 + \alpha_2 + 1,$$
$$k = \overline{1,N}, \quad i_1 = \overline{1,r}, \quad i_2 = \overline{1,r}, \quad \alpha_1 = \overline{0,2}, \quad \alpha_2 = \overline{0,2}.$$

Since Q is a non-negative definite matrix, it can be proved in three steps that the quadratic form in formula (3.20) is also non-negative definite:

1. The final matrix is a block-diagonal matrix consisting of N blocks. Therefore it is necessary to show that each block is a non-negative definite matrix.
2. If we fix t for each k, then it turns out that $u_i(t)$ is a linear function with respect to $p_{ik}^{(0)}$, $p_{ik}^{(1)}$ and $p_{ik}^{(2)}$. Then we can show that the superposition $G(F(\xi))$ is a non-negative definite quadratic form if $G(\eta)$ is a non-negative definite quadratic form and $F(\xi)$ is a linear function. Hence we obtain that under the integral there is a non-negative definite quadratic form with respect to the variables $p_{ik}^{(0)}$, $p_{ik}^{(1)}$ and $p_{ik}^{(2)}$ for each moment t.
3. Next, we use the fact that the result of integrating a non-negative definite quadratic form over the parameter t is a non-negative definite quadratic form.

Remark 3.1. Since piecewise constant and piecewise linear functions are special cases of piecewise quadratic functions, it follows that in (1.14) and (3.7) also non-negative definite quadratic forms were obtained.

Results

It was shown how by choosing the control as a piecewise quadratic function, the linear optimal control problem (1.1)–(1.3), (3.8), (1.5) can be reduced to a programming problem with quadratic constraints (3.9), (3.10), (3.12), (3.13), (3.15), (3.17)–(3.19):

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(c^{i,k,0}p_{ik}^{(0)}+c^{i,k,1}p_{ik}^{(1)}+c^{i,k,2}p_{ik}^{(2)}\right)\longrightarrow\min$$

$$p_{ik}^{(2)}t_k^2+p_{ik}^{(1)}t_k+p_{ik}^{(0)}=p_{ik+1}^{(2)}t_k^2+p_{ik+1}^{(1)}t_k+p_{ik+1}^{(0)},\quad k=\overline{1,N-1},\quad i=\overline{1,r}.$$

$$2p_{ik}^{(2)}t_k+p_{ik}^{(1)}=2p_{ik+1}^{(2)}t_k+p_{ik+1}^{(1)},\quad k=\overline{1,N-1},\quad i=\overline{1,r}.$$

$$\sum_{i=1}^{r}\sum_{k=1}^{N}\left(h_l^{i,k,0}p_{ik}^{(0)}+h_l^{i,k,1}p_{ik}^{(1)}+h_l^{i,k,2}p_{ik}^{(2)}\right)=g_l,\quad l=\overline{1,m}.$$

$$\left.\begin{array}{l} l_i^{(1)}\leqslant p_{ik}^{(2)}t_{k-1}^2+p_{ik}^{(1)}t_{k-1}+p_{ik}^{(0)}\leqslant l_i^{(2)},\\ l_i^{(1)}\leqslant p_{ik}^{(2)}t_k^2+p_{ik}^{(1)}t_k+p_{ik}^{(0)}\leqslant l_i^{(2)},\end{array}\right\}\quad k=\overline{1,N},\quad i=\overline{1,r}.$$

$$\left.\begin{array}{l} s_{ik}^{(1)}=p_{ik}^{(2)}t_{k-1}^2+p_{ik}^{(1)}t_{k-1}+p_{ik}^{(0)}-l_i^{(1)},\\ s_{ik}^{(2)}=p_{ik}^{(2)}t_k^2+p_{ik}^{(1)}t_k+p_{ik}^{(0)}-l_i^{(1)},\\ s_{ik}^{(3)}=l_i^{(2)}-p_{ik}^{(2)}t_{k-1}^2-p_{ik}^{(1)}t_{k-1}-p_{ik}^{(0)},\\ s_{ik}^{(4)}=l_i^{(2)}-p_{ik}^{(2)}t_k^2-p_{ik}^{(1)}t_k-p_{ik}^{(0)},\end{array}\right.\quad k=\overline{1,N},\quad i=\overline{1,r}.$$

$$\left.\begin{array}{l} s_{ik}^{(1)}\geqslant 0,\quad s_{ik}^{(2)}\geqslant 0,\quad s_{ik}^{(3)}\geqslant 0,\quad s_{ik}^{(4)}\geqslant 0,\\ \left(r_{ik}^{(1)}\right)^2\leqslant 4s_{ik}^{(1)}s_{ik}^{(2)},\\ \left(r_{ik}^{(2)}\right)^2\leqslant 4s_{ik}^{(3)}s_{ik}^{(4)},\\ r_{ik}^{(1)}\geqslant 0,\quad r_{ik}^{(2)}\geqslant 0,\end{array}\right.\quad k=\overline{1,N},\quad i=\overline{1,r}.$$

$$T^2 p_{ik}^{(2)} \leqslant N^2 \left(s_{ik}^{(1)} + s_{ik}^{(2)} + r_{ik}^{(1)} \right), \qquad k = \overline{1,N}, \quad i = \overline{1,r}.$$
$$T^2 p_{ik}^{(2)} \geqslant -N^2 \left(s_{ik}^{(3)} + s_{ik}^{(4)} + r_{ik}^{(2)} \right),$$

This problem has the following characteristics:

Variables

1. $3rN$ basic real variables: $p_{ik}^{(0)}$, $p_{ik}^{(1)}$, $p_{ik}^{(2)}$, $k = \overline{1,N}$, $i = \overline{1,r}$.
2. $6rN$ real non-negative variables: $s_{ik}^{(1)}$, $s_{ik}^{(2)}$, $s_{ik}^{(3)}$, $s_{ik}^{(4)}$, $r_{ik}^{(1)}$, $r_{ik}^{(2)}$, $k = \overline{1,N}$, $i = \overline{1,r}$.

Constraints

1. $6rN - 2r + m$ linear constraints: (3.9), (3.10), (3.12), (3.15). Thus, the number of variables can be reduced.
2. $6rN$ linear inequalities: (3.13), (3.18).
3. $2rN$ quadratic inequalities: (3.17). Although the quadratic constraints are not represented by non-negative definite quadratic forms, they are conic equations of the second degree. And then it can be proved that if the variables $s_{ik}^{(1)}$, $s_{ik}^{(2)}$, $s_{ik}^{(3)}$, $s_{ik}^{(4)}$ are non-negative, the set of solutions for these constraints is the convex set.

Thus, for this optimization problem there is a convex set of admissible solutions.

Objective Function

The objective function (3.19) is a linear function of the variables $p_{ik}^{(0)}$, $p_{ik}^{(1)}$, $p_{ik}^{(2)}$, $k = \overline{1,N}$, $i = \overline{1,r}$.

The optimal control problem (1.1)–(1.3), (3.8), (1.7) with a linear system and a quadratic criterion can be reduced to a programming problem with quadratic constraints (3.9), (3.10), (3.12), (3.13), (3.15), (3.17), (3.18), (3.20).

This optimization problem has the same characteristics as the previous version except that the objective function is a convex quadratic function (3.20) of the variables $p_{ik}^{(0)}$, $p_{ik}^{(1)}$, $p_{ik}^{(2)}$, $k = \overline{1,N}$, $i = \overline{1,r}$.

Example. Damping of Sleeping Lagrange Top

Let's return to the damping of the sleeping Lagrange top problem, described in Paragraph 1.5, by constructing the control in the class of piecewise quadratic functions.

The reduction procedure was implemented in Python 3, the quadratic programming problem was solved with the COPT solver (see Appendix A).

The total running time of the program was 1.11 seconds. The value of the objective function is 91.56. The results are shown in Fig. 3.2.

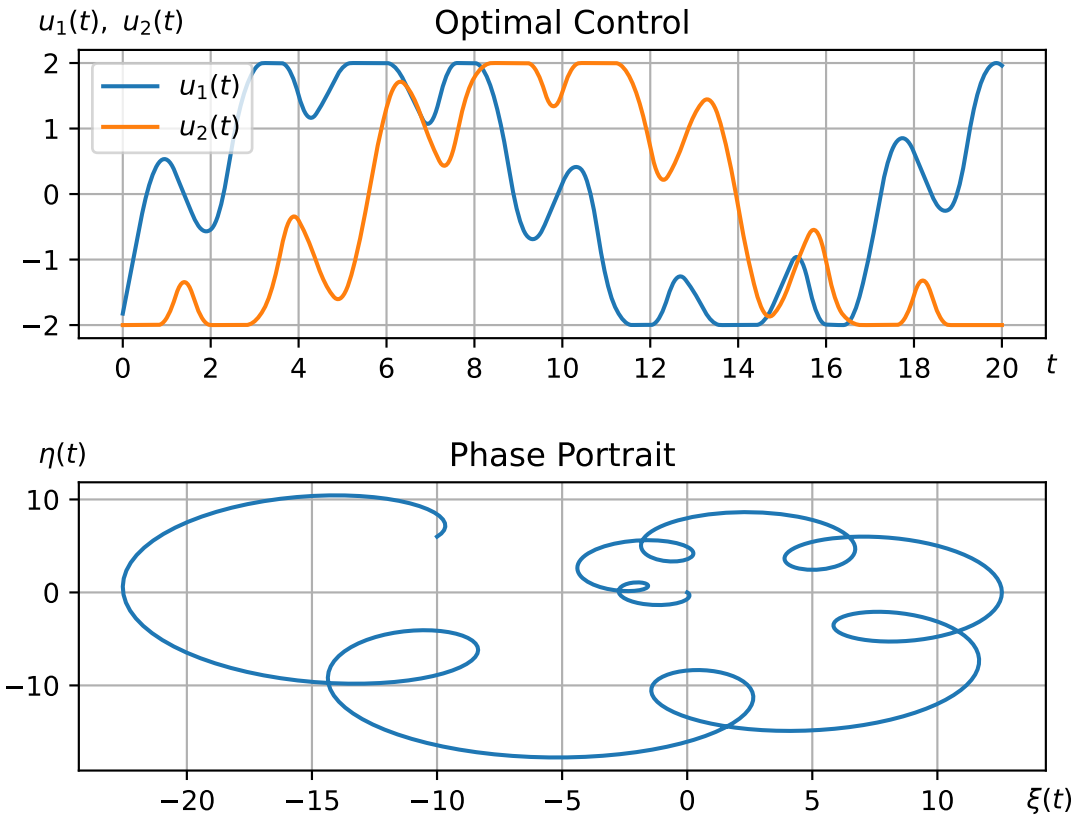

Figure 3.2 — Dumping of Sleeping Lagrange Top in Case of Control from Class of Piecewise Quadratic Functions

3.1.3 Numerical Implementation

Integrating ODEs and Calculating Definite Integrals

In the course of the reduction phase, there was a question of calculating integrals of the type

$$\int_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i(t)dt.$$

First, let's define the way to calculate the matrix $Y(T)Y^{-1}(t)$. Denote by the symbol $Y_{t_1}(t)$ the state-transition matrix of a homogeneous system with matrix $A(t)$ normalized at the point t_1:

$$\dot{Y}_{t_1}(t) = A(t)Y_{t_1}(t), \quad Y_{t_1}(t_1) = E.$$

Now transform the product $Y_0(T)Y_0^{-1}(t)$ using the properties of the state-transition matrix [74]:

$$Y_0(T)Y_0^{-1}(t) = Y_0(T)Y_t(0) = Y_t(T) = Y_T^{-1}(t).$$

Consider a conjugate system

$$\dot{z} = -A^T(t)z. \tag{3.21}$$

Let $Z_{t_1}(t)$ be the state-transition matrix of this system normalized in t_1. Its main property is:

$$Z_{t_1}^T(t) = Y_{t_1}^{-1}(t)$$

Then by substituting $t_1 = T$ we obtain the equation $Z_T^T(t) = Y_T^{-1}(t)$. Hence, the matrix $Y(T)Y^{-1}(t)$ is the transpose state-transition matrix of the conjugate system (3.21) normalized at T. To compute it, it is necessary to solve n Cauchy

problems with the system (3.21) and different initial conditions $z(T) = \vec{e}_j$, $j = \overline{1,n}$. It is assumed that the solutions will be found numerically. For example, in this paper, the function *odeint* from the package *scipy.integrate* was used in Python for this purpose.

After calculating the functions $Z_T(t)$ we are left to calculate the definite integrals $\int\limits_{t_{k-1}}^{t_k} Z_T^T(t)b_i(t)dt$. They can be calculated in quadrature using the function *quad* from the same package *scipy.integrate*.

Integrals like $\int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i t dt$ and $\int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i t^2 dt$ can be calculated by analogy or using the found values of $\int\limits_{t_{k-1}}^{t_k} Y(T)Y^{-1}(t)b_i dt$ and the formulas of integration by parts.

In addition to calculating the above integrals, at the stage of reduction there is a problem of determining the value of

$$Y(T)x_* + \int\limits_0^T Y(T)Y^{-1}(t)d(t)dt.$$

To do this, we need to use the library *odeint* to calculate the solution of the Cauchy problem for the system (1.1) with zero control

$$\dot{\tilde{x}} = A(t)\tilde{x} + d(t), \quad x(0) = x_*$$

and find the value of $\tilde{x}(T)$.

Solution Scheme

The solution to this problem has been implemented in Python 3. The code of the programs is available in appendix A, but now here is the upper-level scheme of the solution.

1. Setting the conditions of the optimal control problem (1.1)–(1.3) with the objective function (1.5) or (1.7).

2. Selecting the control class: piecewise constant (1.4), piecewise linear (3.1) or piecewise quadratic (3.8).
3. Calculating the parameters of a mathematical programming problem, including the use of
   - algorithms for solving systems of linear ordinary differential equations using the function *odeint*.
   - formulas for calculating definite integrals with the function *quad*.
4. Solving a mathematical programming problem with the *coptpy* optimization package.
5. Forming control functions based on the solution of the mathematical programming problem.
6. Finding the solution of the original ODE system, closed by the optimal control, using the function *odeint*.
7. Plotting solutions through functions of the package *matplotlib*.

3.2 Optimal Control with Constraints on Phase Variables and Control Components

In this section we consider the problem of optimal control in the presence of various convex and non-convex constraints on the phase vector and on the control vector.

The basic model will be the control problem (1.1)–(1.5) with a linear system of ordinary differential equations, terminal conditions, direct constraints on controls, piecewise constant controls, and a linear objective functional.

This problem is reduced to a linear programming problem (1.9)–(1.11). All constraints obtained in this chapter will be considered additional to the main model.

3.2.1 Control Components Constraints

Convex Linear Constraints

We supplement the linear problem of optimal control with a set of constraints with respect to the vector of controls u:

$$Du \leqslant d.$$

Here D is a matrix of size $q \times r$, consisting of columns d_i, and d is a q-dimensional vector.

Then we can reduce these constraints to linear inequalities with respect to the variables u_{ik}:

$$\sum_{i=1}^{r} d_i u_{ik} \leqslant d, \quad k = \overline{1, N}. \tag{3.22}$$

Thus we obtained qN linear inequalities, which should be added to the linear (quadratic) programming problem.

Non-Convex Linear Constraints

We now assume that constraints of the «OR» type are imposed on the controls:

$$\left[\begin{array}{l} D^{(1)}u \leqslant d^{(1)}, \\ D^{(2)}u \leqslant d^{(2)}, \\ \dots \\ D^{(s)}u \leqslant d^{(s)}. \end{array}\right. \tag{3.23}$$

Given that the control has two-sided bounds (1.3), for any row $\boldsymbol{\gamma}$ we can find a parameter M such that for any admissible control the inequality

will be satisfied

$$\gamma u \leqslant M.$$

To do this, it is sufficient to determine M as follows:

$$M = \sum_{\substack{i=1 \\ \gamma_i<0}}^{r} \gamma_i l_{*i} + \sum_{\substack{i=1 \\ \gamma_i>0}}^{r} \gamma_i l_i^*.$$

Similarly, we can define a vector $\tilde{d}^{(\alpha)}$ for the matrix $D^{(\alpha)}$. Then for any admissible control the following will be true

$$D^{(\alpha)}u \leqslant \tilde{d}^{(\alpha)}, \quad \alpha = \overline{1,s}.$$

Then, given that the constraint blocks from (3.22) need not always be executed, we introduce binary functions $\delta_\alpha(t)$ that are indicators of which block α must be executed at the moment t:

$$\delta_\alpha(t) = \begin{cases} 1, \text{ if the inequalities } D^{(\alpha)}u(t) \leqslant d^{(\alpha)} \text{ must be satisfied,} \\ 0, \text{ if the inequalities } D^{(\alpha)}u(t) \leqslant d^{(\alpha)} \text{ can be violated,} \end{cases}$$

where $\alpha = \overline{1,s}$. At the same time, there must always be one active block:

$$\sum_{\alpha=1}^{s} \delta_\alpha(t) = 1. \tag{3.24}$$

Then we can rewrite (3.23) as a system of inequalities:

$$\begin{cases} D^{(1)}u \leqslant d^{(1)}\delta_1(t) + \tilde{d}^{(1)}(1-\delta_1(t)), \\ D^{(2)}u \leqslant d^{(2)}\delta_2(t) + \tilde{d}^{(2)}(1-\delta_2(t)), \\ \dots \\ D^{(s)}u \leqslant d^{(s)}\delta_s(t) + \tilde{d}^{(s)}(1-\delta_s(t)). \end{cases} \tag{3.25}$$

Now we can move from dynamic constraints to static ones. Since the control is selected from the class of piecewise constant functions, without limiting the generality of the foregoing, we can also consider $\delta_\alpha(t)$ as a piecewise constant function:

$$\delta_\alpha(t) = \delta_{\alpha k}, \quad k = \overline{1,N}, \quad \alpha = \overline{1,s},$$

where $\delta_{\alpha k}$ are binary variables. Then the equality (3.24) will transform to N linear equations:

$$\sum_{\alpha=1}^{s} \delta_{\alpha k} = 1, \quad k = \overline{1,N}. \tag{3.26}$$

And the system (3.25) can be rewritten as:

$$\sum_{i=1}^{r} d^{i,\alpha} u_{ik} \leqslant d^{(\alpha)} \delta_{\alpha k} + \tilde{d}^{(\alpha)}(1 - \delta_{\alpha k}), \quad k = \overline{1,N}, \quad \alpha = \overline{1,s}. \tag{3.27}$$

Here $d^{i,\alpha}$ is the i-th column of $D^{(\alpha)}$.

Example

Assume that for some control problem there is a two-dimensional vector of controls with the following direct constraints:

$$0 \leqslant u_1(t) \leqslant 10, \quad 0 \leqslant u_2(t) \leqslant 10.$$

In addition, there are two linear constraints:

$$\begin{cases} u_1(t) + u_2(t) \leqslant 17, \\ u_1(t) - 2u_2(t) \leqslant 6. \end{cases}$$

Moreover, the controls must not belong to the rectangle described by the following inequalities:

$$\begin{cases} 2 \leqslant u_1(t) \leqslant 5, \\ 2 \leqslant u_2(t) \leqslant 7. \end{cases}$$

In Fig. 3.3 the set of admissible controls is shown in blue.

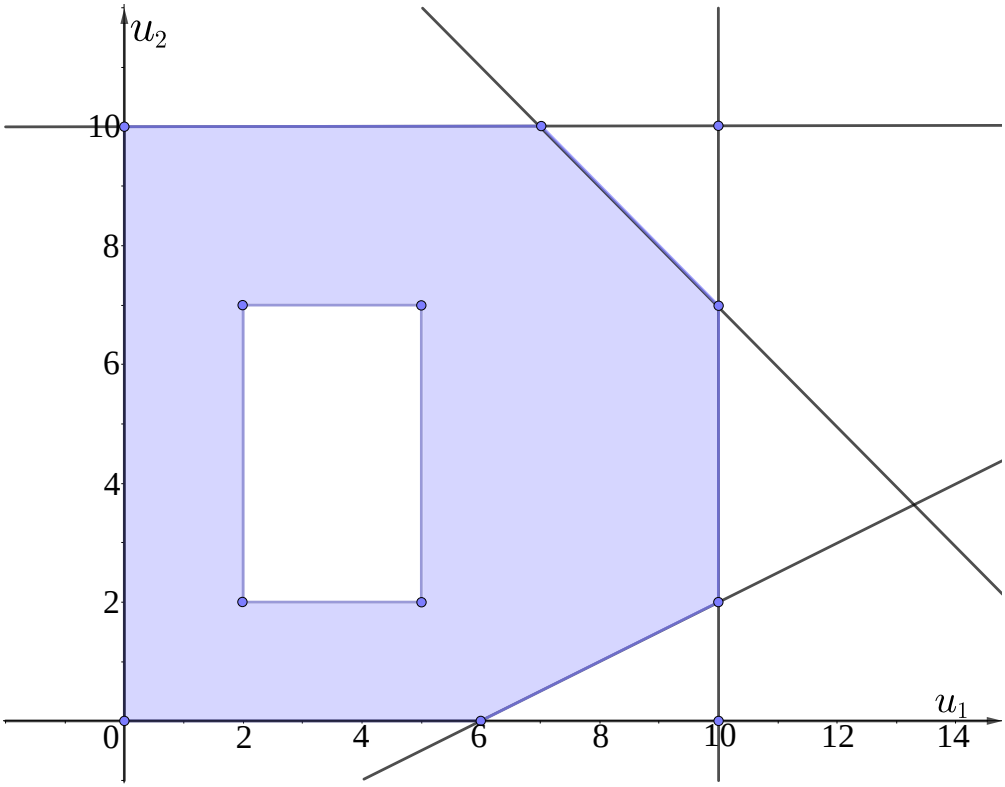

Figure 3.3 — Non-Convex Set of Admissible Controls

Let's convert the requirements for not belonging to the set to the four constraints of type «OR»:

$$\begin{cases} u_1(t) \leqslant 2, \\ u_1(t) \geqslant 5, \\ u_2(t) \leqslant 2, \\ u_2(t) \geqslant 7. \end{cases} \tag{3.28}$$

To obtain a set of equations, we introduce the binary functions $\delta_1(t)$, $\delta_2(t)$, $\delta_3(t)$, $\delta_4(t)$ whose sum is equal to one:

$$\delta_1(t) + \delta_2(t) + \delta_3(t) + \delta_4(t) = 1.$$

Now we can rewrite (3.28) as

$$\begin{cases} u_1(t) \leqslant 2\delta_1(t) + 10(1 - \delta_1(t)), \\ -u_1(t) \leqslant -5\delta_2(t), \\ u_2(t) \leqslant 2\delta_3(t) + 10(1 - \delta_3(t)), \\ -u_2(t) \leqslant -7\delta_4(t). \end{cases}$$

As a result, the set of admissible controls will be described by the following system:

$$\begin{cases} 0 \leqslant u_1(t) \leqslant 10, \\ 0 \leqslant u_2(t) \leqslant 10, \\ u_1(t) + u_2(t) \leqslant 17, \\ u_1(t) - 2u_2(t) \leqslant 6, \\ u_1(t) \leqslant 2\delta_1(t) + 10(1 - \delta_1(t)), \\ -u_1(t) \leqslant -5\delta_2(t), \\ u_2(t) \leqslant 2\delta_3(t) + 10(1 - \delta_3(t)), \\ -u_2(t) \leqslant -7\delta_4(t), \\ \delta_1(t) + \delta_2(t) + \delta_3(t) + \delta_4(t) = 1, \\ \delta_1(t),\ \delta_2(t),\ \delta_3(t),\ \delta_4(t) \in \{0, 1\}. \end{cases}$$

After that, it is necessary to reduce such a system to a system of linear algebraic equations by the algorithm described in the previous paragraphs. Since such a reduction is rather trivial, we will not give it in this example.

3.2.2 Phase Variables Constraints

Similarly to the previous section, we consider convex and non-convex constraints on phase variables in the knots — jumps discontinuity of piecewise constant control functions.

We will also pay special attention to the constraints on the entire time interval $[0, T]$. The execution of such constraints is a more time-consuming operation.

Convex Linear Constraints

At each point t_k, let there be linear constraints on the phase variables

$$\hat{H}_k x(t_k) \leqslant \hat{g}_k^0, \quad k = \overline{1,N}. \tag{3.29}$$

Such constraints strongly resemble terminal conditions (1.2). We will transform them in the same way. Letss write out the Cauchy formula at the point t_k:

$$x(t_k) = Y(t_k)x_* + \int_0^{t_k} Y(t_k)Y^{-1}(t)\left(B(t)u(t) + d(t)\right)dt.$$

We replace the piecewise constant control function with $u(t)$:

$$\begin{gathered} x(t_k) = Y(t_k)x_* + \int_0^{t_k} Y(t_k)Y^{-1}(t)d(t)dt + \\ + \sum_{i=1}^{r}\sum_{\varkappa=1}^{k}\int_{t_{\varkappa-1}}^{t_\varkappa} Y(t_k)Y^{-1}(t)b_i(t)dtu_{i\varkappa}. \end{gathered} \tag{3.30}$$

Let's include this formula into the inequality (3.29):

$$\begin{gathered} \sum_{i=1}^{r}\sum_{\varkappa=1}^{k}\hat{H}_k\int_{t_{\varkappa-1}}^{t_\varkappa} Y(t_k)Y^{-1}(t)b_i(t)dtu_{i\varkappa} \leqslant \\ \leqslant \hat{g}_k^0 - \hat{H}_k Y(t_k)x_* - \hat{H}_k\int_0^{t_k} Y(t_k)Y^{-1}(t)d(t)dt. \end{gathered}$$

After replacing the variables, we obtain linear inequalities with respect to the variables $u_{i\varkappa}$:

$$\sum_{i=1}^{r}\sum_{\varkappa=1}^{k}\hat{h}_k^{i\varkappa}u_{i\varkappa} \leqslant \hat{g}_k, \quad k = \overline{1,N}, \tag{3.31}$$

with the notation

$$\hat{h}_k^{i\varkappa} = \hat{H}_k \int\limits_{t_{\varkappa-1}}^{t_\varkappa} Y(t_k)Y^{-1}(t)b_i(t)dt,$$

$$\hat{g}_k = \hat{g}_k^0 - \hat{H}_k Y(t_k)x_* - \hat{H}_k \int\limits_0^{t_k} Y(t_k)Y^{-1}(t)d(t)dt.$$

As a result, the conditions (3.29) have been reduced to linear constraints (3.31).

Remark 3.2. Similarly, the constraints on phase variables at random points on the segment $[0,\, T]$ can be taken into account.

Non-Convex Linear Constraints

Now let's consider the requirements for phase variables with constraints of the «OR» type:

$$\begin{bmatrix} A_k^{(1)}x(t_k) \leqslant b_k^{(1)}, \\ A_k^{(2)}x(t_k) \leqslant b_k^{(2)}, \\ \dots \\ A_k^{(v)}x(t_k) \leqslant b_k^{(v)}, \end{bmatrix} \quad k = \overline{1, N}. \tag{3.32}$$

Constraints of these types appear in non-convex spaces. For example, when the control object must bypass some third-party object.

The main difference between this case and non-convex control constraints is that there are no direct constraints on the phase variables. However, it is possible to define bounds on the variables in one of two ways:

- Calculate analytically, based on some logical assumptions,
- Find the minimum and maximum for each component $x(t_k)$ by writing out its representation using $u_{i\varkappa}$, and applying knowledge of the boundaries for the controls.

We assume that we have found such vectors $\tilde{b}_k^{(\beta)}$ that any object position $x(t_k)$, which can be achieved by choosing an admissible control, satisfies

$$A_k^{(\beta)} x(t_k) \leqslant \tilde{b}_k^{(\beta)}, \quad k = \overline{1, N}, \quad \beta = \overline{1, v}.$$

Now let's introduce binary variables $\sigma_{\beta k}$ signaling that the constraint $A_k^{(\beta)} x(t_k) \leqslant b_k^{(\beta)}$ is active at time t_k. Then we proceed from the requirements (3.32) to a system of constraints:

$$\begin{cases} A_k^{(1)} x(t_k) \leqslant b_k^{(1)} \sigma_{1k} + \tilde{b}_k^{(1)} (1 - \sigma_{1k}), \\ A_k^{(2)} x(t_k) \leqslant b_k^{(2)} \sigma_{2k} + \tilde{b}_k^{(2)} (1 - \sigma_{2k}), \\ \dots \\ A_k^{(v)} x(t_k) \leqslant b_k^{(v)} \sigma_{vk} + \tilde{b}_k^{(v)} (1 - \sigma_{vk}), \end{cases} \quad k = \overline{1, N}. \tag{3.33}$$

By replacing the right-hand side of (3.30) with $x(t_k)$ in (3.33), we obtain linear inequalities of the following type:

$$\sum_{i=1}^{r} \sum_{\varkappa=1}^{k} a_k^{i\varkappa\beta} u_{i\varkappa} \leqslant q_k^{(\beta)} + b_k^{(\beta)} \sigma_{\beta k} + \tilde{b}_k^{(\beta)} (1 - \sigma_{\beta k}), \quad k = \overline{1, N}, \quad \beta = \overline{1, v}, \tag{3.34}$$

where

$$a_k^{i\varkappa\beta} = A_k^{(\beta)} \int_{t_{\varkappa-1}}^{t_\varkappa} Y(t_k) Y^{-1}(t) b_i(t) dt,$$

$$q_k^{(\beta)} = -A_k^{(\beta)} Y(t_k) x_* - A_k^{(\beta)} \int_0^{t_k} Y(t_k) Y^{-1}(t) d(t) dt.$$

The final thing we need to do is to indicate that at each moment t_k one block of inequalities (3.33) must be active:

$$\sum_{\beta=1}^{v} \sigma_{\beta k} = 1, \quad k = \overline{1,N}. \tag{3.35}$$

Thus, it was shown how the requirements (3.32) can be reduced to a system of linear constraints with binary variables (3.34), (3.35).

Continuous Constraints

The constraints on phase variables over the entire time interval are of great practical interest:

$$Gx(t) \leqslant q, \quad t \in [0,\ T].$$

Unfortunately, such a problem is extremely complex and generally cannot be reduced to a convenient mathematical programming problem.

In this case, several approaches can be proposed to increase the chances of these inequalities being satisfied for any t:

- Increase the number of points in time at which constraints are imposed

$$Gx(t_s) \leqslant q, \quad t_s = \frac{sT}{N_1}, \quad s = \overline{1, N_1}. \quad N_1 \to \infty.$$

- Increase the constraints on those segments in which a strong change in the object's state is expected.
- Estimate the maximum possible deviation of the object's state and put these deviations in the constraints.

$$\begin{array}{l} x_j(t_s) - m_j^1 \leqslant x_j(t) \leqslant x_j(t_s) + m_j^2, \\ j = \overline{1,n}, \quad t \in [t_{s-1},\ t_s] \end{array} \Rightarrow \left\{ \begin{array}{l} G\left(x(t_s) - m^1\right) \leqslant q, \\ G\left(x(t_s) + m^2\right) \leqslant q, \end{array} \right. \quad s = \overline{1, N_1}.$$

Note that the problem with continuous constraints was also studied in [75].

3.3 Chapter 3 Conclusion

Paragraph 3.1 presented methods for reducing the optimal control problem to an optimization problem in the case of piecewise linear and piecewise quadratic controls. Such classes of functions were considered as an alternative to the basic class of piecewise constant functions presented in Chapter 1.

When considering control in the class of piecewise linear functions, the problem with linear system and linear (quadratic) criterion of quality can be reduced to the linear (quadratic) programming problem. Moreover, when considering the control in the class of piecewise quadratic functions, the problem with the linear system and linear (quadratic) criterion of quality is reduced to the convex programming problem with quadratic constraints and linear (quadratic) objective function. Note that for options with a quadratic functional, convexity is preserved when passing to mathematical programming problems if the matrix Q is non-negative definite..

The question of which class of control is worth choosing should be answered by focusing on the properties and criteria of the sought solution. For example, the smoothness of control actions may be an important requirement for the system being studied. In the case of freedom in the choice of class, most likely, it is worth using simpler classes, because solutions for them have less complexity, and by increasing parameter N one can achieve qualitative results in a short time.

Further complication of the class of controls seems unreasonable, since it leads to a disproportionate increase in the complexity of the problem. Even when considering piecewise cubic controllers it will be extremely difficult to reduce direct bounds to static constraints.

Additionally, it is worth noting that the transition to classes of higher orders does not necessarily lead to better solutions, because additional conditions on smoothness may be added. Using the example with damping of Lagrange top one can notice that the minimal value of the functional was obtained by using the piecewise linear class.

In Paragraph 3.2 a linear optimal control problem from the class of piecewise constant functions was considered. Additional restrictions were imposed on this problem.

– **Convex constraints on control.** Such constraints are obviously reduced to linear inequalities with respect to u_{ik} and are easily incorporated into the linear programming problem.
– **Non-convex constraints on control.** To describe a non-convex set, linear constraints of type «OR» are used — at least one of several blocks of equations must be satisfied. For each block a binary variable is introduced. As a result, additional linear constraints with binary variables are introduced into the linear programming problem.
– **Convex constraints on phase variables.** At the control switching points t_k, linear inequalities on the variables x are given. Like terminal conditions, these conditions are reduced to linear constraints on u_{ik}. Also, such constraints can be given for any finite number of predetermined points in time. Thus, the number of constraints added to the model depends on the number of points under consideration and the number of inequalities in each of the points.
– **Non-convex constraints on phase variables.** To describe non-convex sets in knots t_k, linear constraints of type «OR» are set. Similarly to constraints on controls, for each block a binary variable is introduced and linear constraints containing these variables are added to the model.

If all four types of constraints are used, the original problem will be reduced to an optimization problem described by the equations (1.9)–(1.11), (3.22), (3.26), (3.27), (3.31), (3.34), (3.35). According to the generally accepted classification, such a model belongs to the class of mixed integer linear programming (MILP) problems and can be solved using various solvers (see Paragraph 1.4). Additionally, let's note that if we replace linear objective function (1.5) with quadratic one (1.7), we obtain mixed integer quadratic programming problem (1.9), (1.10), (1.11), (3.22), (3.26), (3.27), (3.31), (3.34), (3.35).

The topic of taking into account inequalities on phase variables on the whole interval $[0, T]$ was also mentioned. However, for the approach considered in this dissertation work, it does not seem possible to reduce these conditions to static equations in order to add to the mathematical programming problem.

Finally, let's consider the construction of reachability and controllability sets under additional constraints. In Chapter 2, the problem of constructing sets for the problem (1.1)–(1.4) was reduced to the linear mapping of a r-dimensional cube, since only two-sided constraints on the elements of the control vector were given. In the presence of additional constraints or when considering the control in the class of piecewise linear functions, the problem of determining the sets will consist in a linear mapping of the polyhedron. When piecewise quadratic control is used, the problem will be even more laborious. Therefore the question of construction of sets for such cases is beyond the scope of the current paper.

CHAPTER 4. NON-LINEAR OPTIMAL CONTROL PROBLEM

4.1 Problem Formulation

We assume that the process is described by a system of differential equations, non-linear with respect to the phase variables and linear with respect to the control:

$$\dot{x} = f(t, x) + B(t)u. \tag{4.1}$$

Let's suppose that there are partial derivatives $\frac{\partial f_i(t,x)}{\partial x_j}$ and that the Jacobian matrix $F(t, x)$ of the function $f(t,x)$ is bounded:

$$\|F(t, x)\| \leqslant M, \quad t \in [0, T], \quad x \in \mathbb{R}^n. \tag{4.2}$$

Under this constraint, the Lipschitz condition will be satisfied. Consequently, there exists a single solution of the Cauchy problem for the system (4.1) with the initial condition $x(0) = x_*$ for any given control $u(t)$, satisfying the conditions (1.3), (1.4).

By adding terminal conditions (1.2), control constraints (1.3), (1.4), and one of the quality criteria (1.5)–(1.7), we form a non-linear optimal control problem.

Remark 4.1. By default, we will consider the combination of phase variables (1.5) as the objective function, implying that it can be replaced by resource consumption criterion (1.6) or by quadratic functional (1.7).

The goal of Chapter 4 is to develop a numerical method for finding the control for the given problem (4.1), (1.2)–(1.5). As will be shown later, the control can in some sense be considered suboptimal. As a result, algorithms for two modes will be proposed:

– *Program Mode.* Before the controlled object moves, in several iterations of the algorithm, the approximate control function $u(t)$ and the

corresponding dynamics of changes in the phase variables $x(t)$ are found.

– *Real-Time Control Mode (Positional Mode).* An initial control function is defined. The object moves under the force of this control up to the nearest switch point t_k. Then the control is recalculated based on the current position of the object, taking into account deviations in the phase trajectory caused by inaccurate approximation of the non-linear function and external disturbances. Up to the next point t_{k+1} the object moves under the new control.

4.2 Literature Review

As a continuation of the review presented in Paragraph 1.2, we will describe here the main directions in solving non-linear optimal control problems. The classical approaches mentioned earlier (the Pontryagin's maximum principle [15], dynamic programming [16] and the method of successive approximations [18]) can be extended to non-linear problems.

In modern foreign literature the following basic algorithms for solving a non-linear problem with piecewise constant control [27] can be distinguished:

1. *Direct Single Shooting* [76]. Taking into account the type of control (1.4), we numerically solve the Cauchy problem with the system (4.1) as a system of ordinary differential equations with parameters u_{ik}. Thus, the non-linear dependence of the position $x(T)$ on control values is determined. After that, the problem (4.1), (1.2)–(1.5) is most often reduced to a non-convex non-linear programming problem with Nr unknowns.

2. *Direct Multiple Shooting* [77]. We consider the Cauchy problem with the system (4.1) on the interval $[t_{k-1}, t_k)$:

$$\dot{x}^{(k)} = f(t, x^{(k)}) + \sum_{i=1}^{r} b_i(t) u_{ik}, \quad x^{(k)}(t_{k-1}) = s_k, \quad t \in [t_{k-1}, t_k).$$

In this context, $x^{(k)}$ denotes the solution of the said problem. It can be found numerically as a solution to the problem with $n + r$ parameters s_k and u_{ik}. Then the continuity conditions are added:

$$x^{(k-1)}(t_{k-1}) = s_k, \quad k = \overline{2, N}.$$

After that the original problem is reduced to a non-linear programming problem with $N(n + r)$ unknowns.

3. *Direct Collocation* [78]. A family of algorithms whose basic idea is to replace the phase variable with polynomials. For example, there are popular implementations with replacement of $x(t)$ by a piecewise linear function or by a cubic spline.
4. *Pseudospectral Discretization* [79]. The phase variables are replaced by a linear combination of basic functions, which, unlike the previous approach, are not necessarily polynomials.

The problem of constructing an optimal control for the problem (4.1), (1.2)–(1.5) has been studied in the works of the research team of R. F. Gabasov [38; 80; 81]. Algorithms for two cases were proposed:

1. *Local Domain Optimization:* in case of small deviations of the function $f(x)$ from its linearization at the initial point x_*, this non-linear function is replaced by the linear approximation and the linear optimal control problem is solved, after which the obtained solution is corrected.
2. *Global Domain Optimization:* in the case of large deviations of the non-linear function from its linear approximation, the space of phase variables is divided into regions, and in each region the function $f(x)$ is approximated by a linear function. Then the original non-linear

problem is replaced by a piecewise linear optimal control problem. After solution, the correction of the obtained control is carried out. Examples of such an approach are described in the articles [80; 82].

In addition, in [83] the optimality conditions for the solution of the considered non-linear problem are described.

The author of the thesis in the article [7] formulated a method of positional control of a non-linear system based on successive linearizations of the non-linear system, after which it was covered in detail in the article [8].

Additionally, we note the class of problems with bilinear systems, which is a particular case of non-linearity, and, unlike the system (4.1), the products of phase variables and controls $u_i(t)x(t)$ are allowed. For such systems, the problem of stabilization of a set of program controls (multiprogram stabilization) is solved [84–86].

Finally, there are a number of software programs for solving the optimal control problem. Let's highlight some of them:

1. **CasADI** [87]. Open package for solving non-linear optimization problems. Solutions for optimal control models and model predictive control problems can be computed as well. Direct single shooting, direct multiple shooting, and direct collocation algorithms are used to find the optimal control. The package is provided as a library in Matlab, Python, and C++.
2. **ACADO** [88]. A Matlab tool that implements various optimal control algorithms, including direct single shooting and direct multiple shooting.
3. **DIDO** [89]. A tool for solving the optimal control problem in Matlab, based on pseudospectral algorithms.

4.3 Method of Control Construction

This paragraph describes the main results of the chapter — algorithms for constructing control in the program and real-time modes.

4.3.1 Auxiliary Linear Problem

We consider a non-linear function of the $n+1$ variables $f(t,x)$. Since there exist derivatives $\frac{\partial f_i(t,x)}{\partial x_j}$, the sum of the first two items of the Taylor series can be taken as the linear approximation of the function with respect to the variables x:

$$f(t,x) \approx f\left(t, x^{(0)}\right) + F\left(t, x^{(0)}\right)\left(x - x^{(0)}\right),$$

where $F\left(t, x^{(0)}\right) = \left.\frac{\partial f(t,x)}{\partial x}\right|_{x=x^{(0)}}$. Note that the right-hand side contains a function linear with respect to x and non-linear with respect to t. If we replace the point $x^{(0)}$ with a function $x^{(0)}(t)$, then the linearity property of x is preserved:

$$f(t,x) \approx f\left(t, x^{(0)}(t)\right) + F\left(t, x^{(0)}(t)\right)\left(x - x^{(0)}(t)\right). \tag{4.3}$$

We will call the right-hand side of formula (4.3) the linear approximation of the function $f(t,x)$ along the trajectory $x^{(0)}(t)$.

Let's consider a linear inhomogeneous system of ordinary differential equations:

$$\dot{x} = F\left(t, x^{(0)}(t)\right)\left(x - x^{(0)}(t)\right) + f\left(t, x^{(0)}(t)\right) + B(t)u. \tag{4.4}$$

Compared to equation (4.1), the non-linear function has been replaced by a linear approximation along the trajectory $x^0(t)$ (4.3). If we draw an analogy

with equation (1.1), we can compare

$$A(t) = F\left(t, x^{(0)}(t)\right),$$
$$d(t) = f\left(t, x^{(0)}(t)\right) - F\left(t, x^{(0)}(t)\right) x^0(t).$$

Thus it is clear that the problem (4.4), (1.2)–(1.5) is a linear optimal control problem. The solution for it can be found using the approach described in Chapter 1. We will call this problem an auxiliary problem with respect to the main problem (4.1), (1.2)–(1.5).

4.3.2 Algorithm of Program Mode Construction

Let some initial trajectory of motion $x^{(0)}(t)$ be given for $t \in [0, T]$. In this case, it is not guaranteed that the terminal conditions $Hx(T) = g^0$ are satisfied.

Step 1. We write the system with a linear approximation along the trajectory $x^{(0)}(t)$:

$$\dot{\xi}^{(1)} = F\left(t, x^{(0)}(t)\right)\left(\xi^{(1)} - x^{(0)}(t)\right) + f\left(t, x^{(0)}(t)\right) + B(t)u. \tag{4.5}$$

Let's impose the terminal conditions on the desired trajectory $\xi^{(1)}(t)$:

$$\xi^{(1)}(0) = x_*, \quad H\xi^{(1)}(T) = g^0, \tag{4.6}$$

and as a objective function consider

$$c^T \xi^{(1)}(T) \longrightarrow \min. \tag{4.7}$$

Then the control problem (4.5), (4.6), (1.3), (1.4), (4.7) is similar to the problem (1.1)–(1.5) and can be solved by the method described in Chapter 1: reducing to a linear programming problem followed by LP problem solving by standard methods.

Let's denote the optimal solution of this auxiliary problem by the symbol $u^{(1)}(t)$. Then we close the system (4.1) with this control and find the solution of the Cauchy problem:

$$\dot{x}^{(1)} = f\left(t, x^{(1)}\right) + B(t)u^{(1)}(t),$$
$$x^{(1)}(0) = x_*.$$

The solution $x^{(1)}(t)$ is the motion of the controlled object under the control $u^{(1)}(t)$. The process can be continued iteratively.

Step s. Let $x^{(s-1)}(t)$ be found after $s-1$ iterations. We write an auxiliary linear optimal control problem with a linear approximation of the function $f(t,x)$ along the trajectory $x^{(s-1)}(t)$.

$$\begin{gathered} c^T\xi^{(s)}(T) \longrightarrow \min, \\ \dot{\xi}^{(s)} = F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)} - x^{(s-1)}(t)\right) + f\left(t, x^{(s-1)}(t)\right) + B(t)u, \\ \xi^{(s)}(0) = x_*, \quad H\xi^{(s)}(T) = g^0, \\ l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1,r}, \\ u_i(t) = u_{ik}, \quad t \in [t_{k-1},\ t_k), \quad k = \overline{1,N}, \quad i = \overline{1,r}. \end{gathered} \tag{4.8}$$

After finding the optimal control of the problem (4.8), the function $u^{(s)}(t)$, we form a closed system:

$$\begin{gathered} \dot{x}^{(s)} = f\left(t, x^{(s)}\right) + B(t)u^{(s)}(t), \\ x^{(s)}(0) = x_*. \end{gathered} \tag{4.9}$$

Thus, having as input the vector-function $x^{(s-1)}(t)$ calculated at the previous iteration, we must first solve the linear optimal control problem (4.8) by constructing the optimal control $u^{(s)}(t)$ for it, and then calculate the solution to the Cauchy problem (4.9) — the function $x^{(s)}(t)$.

Initial Trajectory

As an initial trajectory $x^{(0)}(t)$ an approximate solution based on known information about the original problem can be taken. In the absence of such knowledge, we can take $x^{(0)}(t) \equiv x_*$. Then the first step of the algorithm will consider the linear approximation $f(x)$ at the starting point x_*. An alternative option is to set

$$x^{(0)}(t) = (\hat{x} - x_*)\,\frac{t}{T} + x_*,$$

where $\hat{x}$ is some point satisfying the system $H\hat{x} = g^0$. In this case, the trajectory $x^{(0)}(t)$ represents a segment drawn from the initial position to a given plane, so it satisfies the terminal conditions (1.2).

Stopping Condition

As the main convergence metrics of the algorithm we propose to consider two functionals:

$$\begin{gathered} \left\| Hx^{(s)}(T) - g^0 \right\|, \\ \left| J_s - J_{s-1} \right|. \end{gathered} \tag{4.10}$$

The first functional denotes the distance of the final position of the object calculated at the s-th iteration of the algorithm, $x^{(s)}(T)$ from the plane $Hx = g^0$, where the object should have been according to the terminal condition (1.2). However, it is likely that this requirement deviates due to approximation error of the non-linear function $f(t, x)$: the control $u^{(s)}(t)$ ensures that (1.2) is satisfied for the auxiliary system — $H\xi^{(s)}(T) = g^0$, but not for the closed-system solution $x^{(s)}(T)$.

In the case of the expected convergence $\left\| Hx^{(s)}(T) - g^0 \right\| \to 0$ when $s \to \infty$, we can set a constant δ_T and consider the terminal condition satisfied

if the constraint is met

$$\left\| Hx^{(s)}(T) - g^0 \right\| \leqslant \delta_T.$$

The second functional in (4.10) means the change in the value of the objective function compared to the result in the previous iteration. J_s is the value of the objective function for the trajectory $x^{(s)}(t)$ and control $u^{(s)}(t)$. Thus, for the criterion (1.5)

$$J_s = c^T x^{(s)}(T),$$
$$|J_s - J_{s-1}| = \left| c^T \left(x^{(s)}(T) - x^{(s-1)}(T) \right) \right|.$$

The convergence to a single value of the objective function may indicate that it is in an optimum (possibly local). Therefore, let's consider stopping the algorithm when the

$$|J_s - J_{s-1}| \leqslant \delta,$$

where δ is some predetermined constant, the value of which depends on the specifics of the problem, the dimensions of the values and requirements for the accuracy of the result.

As a result, the completion of the algorithm occurs when at least one of the three conditions is met:

- the solution is in the «area of convergence», i. e. the values of the criteria do not exceed the given limits,
- reaching the maximum predetermined number of iterations S_l of the algorithm,
- the end of the allotted time T_l to perform the calculation.

Existence of Solution

The question of the existence of admissible solutions is important. It can be considered in several contexts:

1. Is there a solution to the original problem (4.1), (1.2)–(1.5)?
2. Is there a solution to the auxiliary linear problem (4.8)?
3. If at some iteration s of the auxiliary problem (4.8) the set of possible solutions is empty, does this mean that there are no solutions to (4.1), (1.2)–(1.5) either?

First, we answer the second question. For simplicity, suppose that the matrix H is square, then the constraint on the right end can be represented as a requirement to get to a given position: $x(T) = x^*$. Then, according to the calculations given in Chapter 2, we can construct an reachability set of the form

$$\hat{b}_* \leqslant \hat{A}x(T) \leqslant \hat{b}^*. \tag{4.11}$$

This system of inequalities describes the set of all possible states to which an object whose motion is described by an auxiliary linear system can get from a position x_* in time T under the action of a piecewise constant control satisfying (1.3), (1.4).

Consequently, to check the existence of admissible solutions, it is sufficient to substitute the position in the system (4.11). In addition, it is possible to estimate the closeness of the desired position to the boundary of i-th constraint:

$$\min\left(\hat{a}_i x^* - \hat{b}_{i*}, \hat{b}_i^* - \hat{a}_i x^*\right). \tag{4.12}$$

If $\operatorname{rank} H < n$, i.e., at the moment T the object has to get to the given hyperplane, then it is necessary to check whether there exists a solution to the system of inequalities:

$$\hat{b}_* \leqslant \hat{A}\chi \leqslant \hat{b}^*,$$
$$H\chi = g^0,$$

with respect to an unknown n-dimensional vector χ.

Now let's deal with questions 1 and 3. Unfortunately, the construction of the reachability set of a non-linear problem seems too complicated, so it remains to estimate this only by some indirect signs, one of which is to estimate the reachability set of auxiliary linear problems.

If at some iteration of the algorithm the auxiliary linear problem turned out to be incompatible, it is necessary to estimate the value of the criterion $\left\|Hx^{(s)}(T) - g^0\right\|$. In case the value turned out to be close enough to zero, you can terminate the algorithm and choose the solution obtained at the previous iteration as the best found (non-optimal) admissible solution. If the value of the metric at all performed iterations exceeds the given threshold, we can conclude that an admissible solution cannot be found.

In the case of an unfixed right-hand end ($\operatorname{rank} H < n$) we can assume that the further the value of phase variables of the solution of the auxiliary problem at the final time $\xi^{(s)}(T)$ from the boundaries of the set of admissible solutions, the less the deviation of the closed-loop system $\left\|Hx^{(s)}(T) - g^0\right\|$ will be. The logic is as follows: finding a point $x^{(s)}(T)$ near the boundary of the set described by (4.11) suggests that the control is used «at the limit» to reach this point, i.e., if the control $u(t)$ changes slightly, it may go beyond (1.4). Then, when moving from one linear approximation to another, there is a risk that the parameters of the problem $A(t)$ and $d(t)$ will be rearranged such that an admissible solution cannot be found. Therefore, we can assume that it is «safer» to get the right end away from the boundary of the admissible solutions set. This can be ensured by introducing the distance to the bounds (4.12) into the objective function. To do this, we introduce the variable z into the problem and impose constraints on it:

$$\begin{aligned} z &\leqslant \boldsymbol{\gamma}^T \left(\hat{A}x^{(s)}(T) - \hat{b}_*\right), \\ z &\leqslant \boldsymbol{\gamma}^T \left(\hat{b}^* - \hat{A}x^{(s)}(T)\right). \end{aligned} \tag{4.13}$$

Here $\boldsymbol{\gamma}$ is a vector of conditional weights of inequalities, in the simplest case, we can set all weights equal to one. Thus, the constraints (4.13) ensure that z is less than the distance from $x^{(s)}(T)$ to the boundary of the set of admissible solutions. The geometric meaning of this approach is illustrated in Fig. 4.1. Next step: update the objective function:

$$J^* = J - \mu z \longrightarrow \min. \tag{4.14}$$

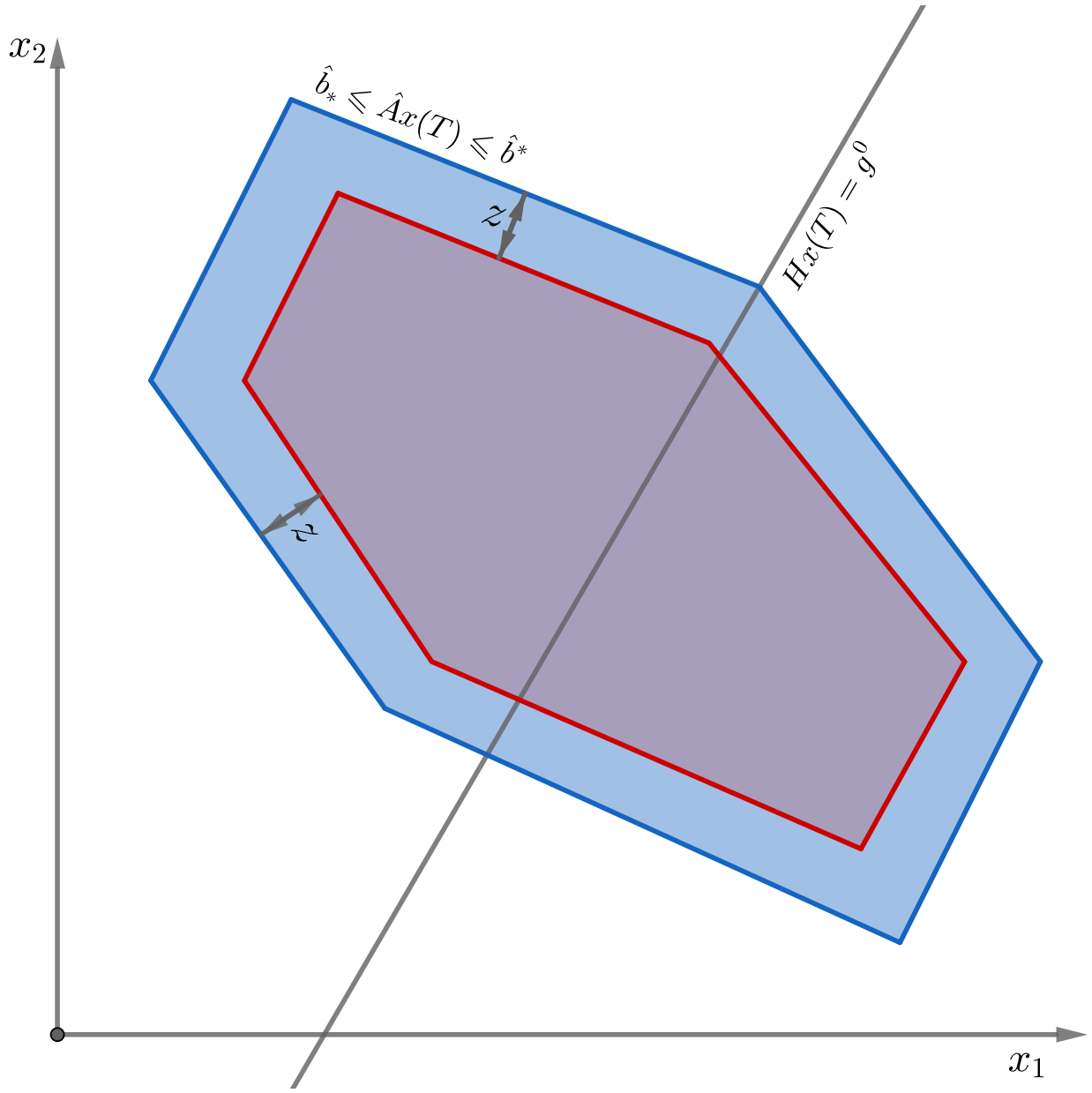

Figure 4.1 — Geometric Meaning of Constraints (4.13)

In formula (4.14) J is one of the objective functions (1.5)–(1.7), μ is a non-negative parameter governing the optimization focus. With small values of μ the focus will be on obtaining solutions closer to the optimal one, with large values the focus will be on more «reliable values», when the value of $x(T)$ lies far from the boundary of the set of acceptable solutions.

4.3.3 Algorithm of Positional Mode Construction

Let's assume that the motion of the controlled object cannot be predicted in advance. For example, this may be the case when a random external disturbance is applied:

$$\dot{x} = f(t, x) + B(t)u + \eta(t), \tag{4.15}$$

$\eta(t)$ is some random process with zero mathematical expectation.

In such a case, real-time control — realignment of control during the process of object motion — is a good way out. In addition, this approach can be useful in the following cases:

- While constructing a program control, the terminal conditions could not be met, and it is known that without recalculating the control the requirement $Hx(T) = g^0$ will not be met either.
- The terminal condition itself has changed. Now the constraint $\tilde{H}x(T) = \tilde{g}^0$ must be satisfied. For example, it is possible if the goal is to reach a moving object.
- For some reason the criterion for the quality of the solution has changed.

Since the control is a piecewise constant function with sampling period h, the control can only be realigned at points t_k.

Step 1. Suppose that some program control $\bar{u}^{(0)}(t)$ and its corresponding motion $\bar{x}^{(0)}(t)$ were constructed. At the moment t_1 the object is in the position $\tilde{x}^{(1)}$. For certain reasons, the control needs to be rebuilt. In such a case, it is necessary to solve the following non-linear optimal control problem:

$$c^T x(T) \longrightarrow \min,$$
$$\dot{x} = f(t, x) + B(t)u + \eta(t),$$
$$x(t_1) = \tilde{x}^{(1)}, \quad Hx(T) = g^0,$$
$$l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1, r},$$
$$u_i(t) = u_{ik'}, \quad t \in [t_{k'-1},\ t_{k'}), \quad k' = \overline{2, N}, \quad i = \overline{1,r},$$

which can be solved by an iterative algorithm similar to the one proposed for the program mode:

$$c^T\xi^{(s)}(T) \longrightarrow \min,$$
$$\dot{\xi}^{(s)} = F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)} - x^{(s-1)}(t)\right) + f\left(t, x^{(s-1)}(t)\right) + B(t)u,$$
$$\xi^{(s)}(t_1) = \tilde{x}^1, \quad H\xi^{(s)}(T) = g^0,$$
$$l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1,r},$$
$$u_i(t) = u_{ik'}, \quad t \in [t_{k'-1},\ t_{k'}), \quad k = \overline{2,N}, \quad i = \overline{1,r}.$$

The initial trajectory for the control recalculation process can be the trajectory in the program mode $x(t)$.

Remark 4.2. The number of iterations of the algorithm is equal to S_1 and is determined by the conditions of exit from the algorithm to construct the program mode.

Remark 4.3. Since the calculation of the new trajectory will not occur instantaneously, it will obviously be necessary to recalculate the trajectory before the time t_1. It is logical to choose the moment $t_1 - T_l$ as the time when recalculation begins, where T_l is the time limit for constructing the program mode. However, in this case the position of the object $\tilde{x}^1$ at time t_1 will not be known for certain. Therefore, it is more correct to say that $\tilde{x}^1$ is a prediction of the object's position at the moment t_1, constructed at $t_1 - T_l$.

Denote the found control by the symbol
$baru^{(1)}(t)$ and the motion by $\bar{x}^{(1)}(t)$. Let's continue the process of the object's motion.

Step k. Let at the moment $t_k - T_l$ assume that at the time t_k the object will be in the position $\tilde{x}^{(k)}$. Let's construct the problem of controlling the object from this position:

$$
\begin{gathered}
c^T x(T) \longrightarrow \min, \\
\dot{x} = f(t,x) + B(t)u + \eta(t), \\
x(t_k) = \tilde{x}^{(k)}, \quad Hx(T) = g^0, \\
l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1,r}, \\
u_i(t) = u_{ik'}, \quad t \in [t_{k'-1},\ t_{k'}), \quad k' = \overline{k+1,N}, \quad i = \overline{1,r}.
\end{gathered} \tag{4.16}
$$

In this case $k = \overline{1,N-1}$. The non-linear optimal control problem (4.16) can be solved using the iterative algorithm:

$$
\begin{gathered}
c^T \xi^{(s)}(T) \longrightarrow \min, \\
\dot{\xi}^{(s)} = F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)} - x^{(s-1)}(t)\right) + f\left(t, x^{(s-1)}(t)\right) + B(t)u, \\
\xi^{(s)}(t_k) = \tilde{x}^{(k)}, \quad H\xi^{(s)}(T) = g^0, \\
l_{*i} \leqslant u_i(t) \leqslant l_i^*, \quad i = \overline{1,r}, \\
u_i(t) = u_{ik'}, \quad t \in [t_{k'-1},\ t_{k'}), \quad k' = \overline{k+1,N}, \quad i = \overline{1,r}.
\end{gathered} \tag{4.17}
$$

The initial trajectory to start the iterative algorithm can be the function $\bar{x}^{(k-1)}(t)$, taken on the interval $[t_k, T]$. We denote the found control and trajectory by $\bar{u}^{(k)}(t)$ and $\bar{x}^{(k)}(t)$, respectively.

Thus, the constructed control will be used each time on an interval of length h, after which it will be reconstructed depending on the position of the object. Let's write the final implementation of the control:

$$
\bar{u}(t) = \begin{cases} \bar{u}^{(k)}(t), \quad t \in [t_k,\ t_k + 1), \\ k = \overline{0,N-1}, \end{cases} \tag{4.18}
$$

The function $\bar{u}(t)$ is called the positional control of the problem (4.15), (1.2)–(1.5).

Remark 4.4. The optimality of the control $\bar{u}(t)$ depends on the optimality of program controls $\bar{u}^{(k)}(t)$. If the optimality (suboptimality) of the problem solutions (4.16) is shown, then we call the function $\bar{u}(t)$ as the optimal (suboptimal) positional control.

Existence of Solution

In addition to questions about the existence of an admissible control mentioned in Section 4.3.2, it is necessary to think about the loss of controllability at moments of control recalculation: there is a possible situation when the set of admissible solutions of the problem (4.16) at step $k-1$ was not empty, however, at step k it turned out that there were no admissible solutions. This can happen due to the following factors:

1. Error of approximation of non-linear function.
2. Deviation from the program motion due to external disturbances.
3. Change of problem statement.

In order to ensure safer motion (in terms of exiting the controllability set), it is proposed to introduce the variable z and constraints (4.13) into problems (4.16) and (4.17), and to change the objective function according to (4.14).

4.4 Theoretical Justification

4.4.1 Convergence of Trajectories in Program Mode

Let's subtract auxiliary linear system (4.8) from non-linear system (4.9):

$$\dot{x}^{(s)} - \dot{\xi}^{(s)} = f\left(t, x^{(s)}\right) + B(t)u^{(s)}(t) -$$
$$-F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)} - x^{(s-1)}(t)\right) - f\left(t, x^{(s-1)}(t)\right) - B(t)u^{(s)}(t) =$$
$$= f\left(t, x^{(s)}\right) - f\left(t, x^{(s-1)}(t)\right) - F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)} - x^{(s-1)}(t)\right).$$

Let's introduce variable $\Delta^{(s)}(t)$ denoting the difference between the solution of the original problem and the auxiliary one at the s-th iteration

of the algorithm $x^{(s)}(t) - \xi^{(s)}(t)$. Then, we can write a system of differential equations with respect to variable $\Delta^{(s)}(t)$:

$$\begin{aligned}\dot{\Delta}^{(s)} = f\left(t, \xi^{(s)}(t) + \Delta^{(s)}\right) - f\left(t, x^{(s-1)}(t)\right) - \\ - F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)}(t) - x^{(s-1)}(t)\right),\end{aligned} \tag{4.19}$$

under the initial condition $\Delta^{(s)}(0) = x^{(s)}(0) - \xi^{(s)}(0) = x_* - x_* = 0$.

Lemma 4.1. *Let there be a time derivative* $\frac{\partial f(t,x)}{\partial t}$ *and second derivatives* $\frac{\partial^2 f(t,x)}{\partial x_j \partial t}$ *and* $\frac{\partial^2 f(t,x)}{\partial x_{j_1} \partial x_{j_2}}$. *Then the deviation of the solution of non-linear closed-loop system (4.9) from the solution of optimal control problem (4.8) is an infinitesimal value with respect to* t^2, *i. e.* $\Delta^{(s)}(t) = o\left(t^2\right)$.

P r o o f . Let's write out the Taylor formula for the function $\Delta^{(s)}(t)$ at the point 0:

$$\Delta^{(s)}(t) = \Delta^{(s)}(0) + \dot{\Delta}^{(s)}(0)t + \ddot{\Delta}^{(s)}(0)\frac{t^2}{2} + o\left(t^2\right).$$

We will show that the first three items are zero. For the deviation at the starting point it is obvious $\Delta^{(s)}(0) = x^{(s)}(0) - \xi^{(s)}(0) = x_* - x_* = 0$. Let's put the value $t = 0$ into formula (4.19):

$$\dot{\Delta}^{(s)}(0) = f(0, x_*) - f(0, x_*) - F(0, x_*)(x_* - x_*) = 0.$$

Next, find the second derivative of the deviation by differentiating by t formula (4.19).

$$\ddot{\Delta}^{(s)}(t) = \frac{\partial f\left(t, \xi^{(s)}(t) + \Delta^{(s)}(t)\right)}{\partial t} +$$
$$+F\left(t, \xi^{(s)}(t) + \Delta^{(s)}(t)\right)\left(\dot{\xi}^{(s)}(t) + \dot{\Delta}^{(s)}(t)\right) -$$
$$-\frac{\partial f\left(t, x^{(s-1)}(t)\right)}{\partial t} - F\left(t, x^{(s-1)}(t)\right)\dot{x}^{(s-1)}(t) -$$
$$-\left(\frac{\partial F\left(t, x^{(s-1)}(t)\right)}{\partial t} + \sum_{j=1}^{n} \left.\frac{\partial F\left(t, x\right)}{\partial x_j}\right|_{x=x^{(s-1)}(t)} \dot{x}_j^{(s)}(t)\right)\left(\xi^{(s)}(t) - x^{(s-1)}(t)\right) -$$
$$-F\left(t, x^{(s-1)}(t)\right)\left(\dot{\xi}^{(s)}(t) - \dot{x}^{(s-1)}(t)\right) =$$
$$= F\left(t, \xi^{(s)}(t) + \Delta^{(s)}(t)\right)\dot{\Delta}^{(s)}(t) + \frac{\partial f\left(t, \xi^{(s)}(t) + \Delta^{(s)}(t)\right)}{\partial t} -$$
$$-\frac{\partial f\left(t, x^{(s-1)}(t)\right)}{\partial t} + \left(F\left(t, \xi^{(s)}(t) + \Delta^{(s)}(t)\right) - F\left(t, x^{(s-1)}(t)\right)\right)\dot{\xi}^{(s)}(t) -$$
$$-\left(\frac{\partial F\left(t, x^{(s-1)}(t)\right)}{\partial t} + \sum_{j=1}^{n} \left.\frac{\partial F\left(t, x\right)}{\partial x_j}\right|_{x=x^{(s-1)}(t)} \dot{x}_j^{(s)}(t)\right)\left(\xi^{(s)}(t) - x^{(s-1)}(t)\right).$$

Hereinafter $\frac{\partial F(t,x)}{\partial t} = \frac{\partial f(t,x)}{\partial x \partial t}$, and $\frac{\partial F(t,x)}{\partial x_j} = \frac{\partial f(t,x)}{\partial x \partial x_j}$. Now we get the value at the starting point:

$$\ddot{\Delta}^{(s)}(0) = F\left(0, x_*\right)\dot{\Delta}^{(s)}(0) + \left.\frac{\partial f\left(t, x_*\right)}{\partial t}\right|_{t=0} - \left.\frac{\partial f\left(t, x_*\right)}{\partial t}\right|_{t=0} +$$
$$+\left(F\left(0, x_*\right) - F\left(0, x_*\right)\right)\dot{\xi}^{(s)}(0) -$$
$$-\left(\left.\frac{\partial F\left(t, x_*\right)}{\partial t}\right|_{t=0} + \sum_{j=1}^{n} \left.\frac{\partial F\left(0, x\right)}{\partial x_j}\right|_{x=x_*} \dot{x}_j^{(s)}(0)\right)\left(x_* - x_*\right) = 0.$$

Thus, it is shown that $\Delta^{(s)}(t) = o\left(t^2\right)$. □

Let's study the dependence of the algorithm convergence on the change of the vector of controls. We introduce

$$\delta_s = \max_{t\in[0,T]} \left\| B(t)\left(u^{(s)}(t) - u^{(s-1)}(t)\right)\right\|, \tag{4.20}$$

which is the maximum deviation of the current control from the control at the previous step on the interval $[0, T]$ (multiplied by $B(t)$). Let's analyze the effect of this parameter on the deviation of the original problem solution from the auxiliary problem solution.

Lemma 4.2. *The deviation of the auxiliary linear problem solution from the non-linear system solution, closed by the control obtained at the previous step of the algorithm, satisfies the inequality*

$$\left\|\xi^{(s)}(t) - x^{(s-1)}(t)\right\| \leqslant \frac{1}{M}\left(e^{Mt} - 1\right)\delta_s, \quad \forall s \geqslant 2, \quad t \in [0, T]. \tag{4.21}$$

P r o o f . Let's write out a system of differential equations for x^{s-1} and ξ^s:

$$\dot{x}^{(s-1)}(t) = f\left(t, x^{(s-1)}(t)\right) + B(t)u^{(s-1)}(t),$$
$$\dot{\xi}^{(s)}(t) = F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)}(t) - x^{(s-1)}(t)\right) + f\left(t, x^{(s-1)}(t)\right) + B(t)u^{(s)}(t).$$

Subtract the first equation from the second:

$$\begin{aligned}\dot{\xi}^{(s)}(t) - \dot{x}^{(s-1)}(t) = F\left(t, x^{(s-1)}(t)\right)\left(\xi^{(s)}(t) - x^{(s-1)}(t)\right) + \\ + B(t)\left(u^{(s)}(t) - u^{(s-1)}(t)\right).\end{aligned} \tag{4.22}$$

Equality (4.22) is a system of differential equations with respect to $\xi^{(s)}(t) - x^{(s-1)}(t)$ under the initial condition $\xi^{(s)}(0) - x^{(s-1)}(0) = x_* - x_* = 0$. Then we can find the solution using the Cauchy formula:

$$\xi^{(s)}(t) - x^{(s-1)}(t) = Y_s(t)\int\limits_0^t Y_s^{-1}(\tau)B(\tau)\left(u^{(s)}(\tau) - u^{(s-1)}(\tau)\right)d\tau, \tag{4.23}$$

where $Y_s(t)$ is the state-transition matrix of the system (4.22) normalized at 0. Let's estimate the norm of the difference in the left part of (4.23):

$$\left\|\xi^{(s)}(t) - x^{(s-1)}(t)\right\| \leqslant \|Y_s(t)\|\int\limits_0^t \left\|Y_s^{-1}(\tau)\right\| \cdot \left\|B(\tau)\left(u^{(s)}(\tau) - u^{(s-1)}(\tau)\right)\right\| d\tau.$$

Then, taking into account state-transition matrix properties and notation (4.20):

$$\left\|\xi^{(s)}(t) - x^{(s-1)}(t)\right\| \leqslant e^{Mt}\int\limits_0^t e^{-M\tau}\delta_s d\tau = \frac{1}{M}\left(e^{Mt} - 1\right)\delta_s.$$

□

Lemma 4.3. *For the difference of solutions of the non-linear system, closed by the controls found at the current and previous steps of the algorithm, the constraint is valid:*

$$\left\|x^{(s)}(t) - x^{(s-1)}(t)\right\| \leqslant \frac{1}{M}\left(e^{Mt} - 1\right)\delta_s, \quad \forall s \geqslant 2, \quad t \in [0, T]. \tag{4.24}$$

P r o o f . Let's consider the difference of $x^{(s)}(t)$ and $x^{(s-1)}(t)$:

$$\begin{gathered} \dot{x}^{(s)}(t) - \dot{x}^{(s-1)}(t) = \\ = f\left(t, x^{(s)}(t)\right) - f\left(t, x^{(s-1)}(t)\right) + B(t)\left(u^{(s)}(t) - u^{(s-1)}(t)\right). \end{gathered} \tag{4.25}$$

Let's integrate expression (4.25) on the interval $[0, t]$ taking into account initial condition $x^{(s)}(0) - x^{(s-1)}(0) = x_* - x_* = 0$:

$$x^{(s)}(t) - x^{(s-1)}(t) = \int_0^t \left(f\left(\tau, x^{(s)}(\tau)\right) - f\left(\tau, x^{(s-1)}(\tau)\right)\right) d\tau + \\ + \int_0^t B(\tau)\left(u^{(s)}(\tau) - u^{(s-1)}(\tau)\right) d\tau.$$

Let's use the Lagrange mean value theorem:

$$f\left(t, x^{(s)}(t)\right) - f\left(t, x^{(s-1)}(t)\right) = F\left(t, \mu(t)\right)\left(x^{(s)}(t) - x^{(s-1)}(t)\right),$$

where $\mu(t)$ is a function taking values between $x^{(s-1)}(t)$ and $x^{(s)}(t)$ for any t. Next, we estimate $x^{(s)}(t) - x^{(s-1)}(t)$ by the norm:

$$\left\|x^{(s)}(t) - x^{(s-1)}(t)\right\| \leqslant \int_0^t \left\|F\left(\tau, \mu(\tau)\right)\right\| \cdot \left\|\left(x^{(s)}(\tau) - x^{(s-1)}(\tau)\right)\right\| d\tau + \\ + \int_0^t \left\|B(\tau)\left(u^{(s)}(\tau) - u^{(s-1)}(\tau)\right)\right\| d\tau \leqslant \\ \leqslant M \int_0^t \left\|\left(x^{(s)}(\tau) - x^{(s-1)}(\tau)\right)\right\| d\tau + \delta_s t.$$

Finally, using a strengthened form of Gronwall's lemma [90], we obtain the inequality:

$$\left\|x^{(s)}(t) - x^{(s-1)}(t)\right\| \leqslant \frac{1}{M}\left(e^{Mt} - 1\right)\delta_s$$

□

Lemma 4.4. *The norm of deviation of the non-linear closed-loop system solution (4.9) from the solution of corresponding auxiliary linear optimal control problem (4.8) can be bounded from above:*

$$\left\|x^{(s)}(t) - \xi^{(s)}(t)\right\| \leqslant \frac{2}{M}\left(e^{Mt} - 1\right)\delta_s, \quad \forall s \geqslant 2, \quad t \in [0, T]. \tag{4.26}$$

P r o o f . Let's use the triangle inequality and relations (4.21) and (4.24):

$$\begin{gathered}\left\|x^{(s)}(t) - \xi^{(s)}(t)\right\| = \left\|x^{(s)}(t) - x^{(s-1)}(t) + x^{(s-1)}(t) - \xi^{(s)}(t)\right\| \leqslant \\ \leqslant \left\|x^{(s)}(t) - x^{(s-1)}(t)\right\| + \left\|\xi^{(s)}(t) - x^{(s-1)}(t)\right\| \leqslant \\ \leqslant \frac{1}{M}\left(e^{Mt} - 1\right)\delta_s + \frac{1}{M}\left(e^{Mt} - 1\right)\delta_s = \frac{2}{M}\left(e^{Mt} - 1\right)\delta_s.\end{gathered}$$

□

Corollary 4.1. *There is an upper estimate of the distance of the final object position from the plane given by terminal condition $Hx(T) = g^0$:*

$$\left\|Hx^{(s)}(T) - g^0\right\| \leqslant \frac{2\|H\|}{M}\left(e^{MT} - 1\right)\delta_s \quad s \geqslant 2. \tag{4.27}$$

P r o o f . Note that according to the conditions of the problem (4.8) $H\xi^{(s)}(T) = g^0$ is valid. Then, given (4.27):

$$\begin{gathered}\left\|Hx^{(s)}(T) - g^0\right\| = \left\|H\left(x^{(s)}(T) - \xi^{(s)}(T)\right)\right\| \leqslant \\ \leqslant \|H\| \cdot \left\|x^{(s)}(T) - \xi^{(s)}(T)\right\| \leqslant \|H\|\,\frac{2}{M}\left(e^{MT} - 1\right)\delta_s.\end{gathered}$$

□

Corollary 4.2. *Let there exist a limit of the sequence $\lim\limits_{s\to+\infty} u^{(s)}(t) = u(t)$, $\forall t \in [0, T]$. Then the solutions of auxiliary linear optimal control problems (4.8) and the solutions of closed non-linear systems of differential equations (4.9) converge to a single limit:*

$$\lim_{s\to+\infty} \xi^{(s)}(t) = \lim_{s\to+\infty} x^{(s)}(t) = x(t), \quad t \in [0, T]. \tag{4.28}$$

Moreover, the function $x(t)$ is an admissible solution to problem (4.1), (1.2)–(1.5).

P r o o f . Since the convergence of the controls implies that $\delta_s \to 0$ at $s \to +\infty$, the solution $x^{(s)}(t)$ converges to $\xi^{(s)}(t)$ according to (4.26) and to $x^{(s-1)}(t)$ according to (4.24). The limit $x(t)$ is an admissible solution of the original problem since $Hx(T) = g^0$, based on inequality (4.27). □

4.4.2 Convergence of Trajectories in Positional Mode

The previous statements were devoted to the program mode. Let's now consider the real-time control mode. An important characteristic of the control problem (4.15), (1.2)–(1.5) is the number of control switching points N. For real-time control, the same number shows the number of control recalculation points. Assume that the number N is a hyperparameter of the model and can be varied. On the one hand, increasing N will make the problem more complicated. So, for real-time control we will have to solve the control problem (4.16) more often, and the problem itself will have a larger dimension than for small N. On the other hand, increasing this parameter will make it possible to achieve better convergence.

Lemma 4.5. *Let $\bar{x}(t)$ be the solution of system (4.15) closed by positional control (4.18), with initial condition $\bar{x}(0) = x_*$. Then as N increases, final position $\bar{x}(t)$ tends to plane $Hx = g^0$:*

$$\lim_{N \to +\infty} \left\| H\bar{x}(T) - g^0 \right\| = 0.$$

P r o o f . The solution of problems (4.15), (1.2)–(1.5) consists in the consecutive solution of N problems (4.16). The last step will solve the problem of finding the control on the half-interval $[t_{k-1}, t_k)$ or what is the same $[(N-1)h, Nh)$. It is not difficult to show that for such a problem constraint (4.27) will be

$$\left\| H\bar{x}^{(s)}(T) - g^0 \right\| \leqslant \frac{2\|H\|}{M} \left(e^{Mh} - 1 \right) \delta_s.$$

In this context, $\bar{x}^{(s)}(t)$ is the solution to the closed system

$$\dot{\bar{x}}^{(s)} = f(t, \bar{x}^{(s)})(t) + B(t)\bar{u}^{(s)}(t),$$
$$\bar{x}\left((N-1)h\right) = \tilde{x}^{(N-1)},$$

and the control $\bar{u}^{(s)}(t)$ is found at the s-step of problem (4.17) at $k = N-1$. Then, given that M and $\|H\|$ are constants, and δ_s according to (4.20) and (1.3) is bounded from above, the expression

$$\left\| H\bar{x}^{(s)}(T) - g^0 \right\|$$

tends to zero at $h \to 0$. And since $h = T/N$, the same conclusion is true for $N \to +\infty$. □

Remark 4.5. The proof of Lemma 4.5 is given for the case $s \geqslant 2$, but it can be shown that the statement is also true for $s = 1$.

Remark 4.6. Lemma 4.5 is valid only in the absence of external perturbations: $\eta(t) \equiv 0$.

4.4.3 Convergence of Controls in Program Mode

The most important question is the convergence of controls when constructing program control. As was shown in Corollary 4.2, it depends on this property whether there exists a limit to the sequence of constructed solutions and whether the algorithm converges to an admissible solution of the original problem. Let us return to formula (4.23) describing the difference between the solution of the auxiliary linear problem and the solution of the non-linear system obtained in the previous step of the algorithm. Substitute $t = T$ and take into account terminal condition (1.2):

$$H\xi^{(s)}(T) = Hx^{(s-1)}(t) + Y_s(T) \int_0^T Y_s^{-1}(t)B(t)\left(u^{(s)}(t) - u^{(s-1)}(t)\right) dt = g^0.$$

Similar to the algorithm for the reduction of terminal conditions presented in Chapter 1, this equality can be transformed into a system of linear equations:

$$\sum_{i=1}^{r}\sum_{k=1}^{N} h_{ik}^{(s)}\left(u_{ik}^{(s)} - u_{ik}^{(s-1)}\right) = g^0 - Hx^{(s-1)}(T),$$
$$h_{ik}^{(s)} = \int_{t_{k-1}}^{t_k} HY_s(T)Y_s^{-1}(t)b_i(t)dt, \quad i = \overline{1,r}, \quad k = \overline{1,N}.$$

Here $u_{ik}^{(s-1)}$ and $u_{ik}^{(s)}$ are the values of i-th element of the vector of controls at the k-th time interval at the previous and the current step of the algorithm, respectively. Then problem (4.8) can be rewritten as a linear programming problem:

$$\begin{gathered}\sum_{i=1}^{r}\sum_{k=1}^{N} c_{ik}^{(s)}u_{ik}^{(s)} \longrightarrow \min,\\ \sum_{i=1}^{r}\sum_{k=1}^{N} h_{ik}^{(s)}\left(u_{ik}^{(s)} - u_{ik}^{(s-1)}\right) = g^0 - Hx^{(s-1)}(T),\\ l_{*i} \leqslant u_{ik}^{(s)} \leqslant l_i^*, \quad i = \overline{1,r}, \quad k = \overline{1,N}.\end{gathered} \tag{4.29}$$

The numbers $c_{ik}^{(s)}$ have an upper index s because they depend on the state-transition matrix $Y_s(t)$. Note that unlike this objective function, criteria (1.6) and (1.7) do not depend on the current linear approximation of function $f(t,x)$.

Let's analyze possible scenarios for the relationship between controls at neighbor steps $u_{ik}^{(s-1)}$ and $u_{ik}^{(s)}$.

1. $u_{ik}^{(s-1)}$ is not an admissible solution of problem (4.29). In such a case, the optimal solution $u_{ik}^{(s)}$ must be constructed, which differs from the control in the previous iteration. In this case there will not necessarily be an approximation to the plane $Hx = g^0$ or an improvement of the objective function.
2. $u_{ik}^{(s-1)}$ is an admissible, but not optimal solution of problem (4.29). It is easy to see that the control from the previous step will only be admissible if $Hx^{(s-1)}(T) = g^0$. Hence, the control $u^{(s-1)}(t)$ is also an admissible control of original problem (4.1), (1.2)–(1.5). However, there

exists a control $u_{ik}^{(s)}$ such that

$$\sum_{i=1}^{r}\sum_{k=1}^{N} c_{ik}^{(s)}\left(u_{ik}^{(s)} - u_{ik}^{(s-1)}\right) < 0.$$

Then the new control will be related to the old relation $\sum\limits_{i=1}^{r}\sum\limits_{k=1}^{N} h_{ik}^{(s)}\left(u_{ik}^{(s)} - u_{ik}^{(s-1)}\right) = 0$, but nevertheless it is not guaranteed that such a solution would be an admissible solution to the non-linear problem. Moreover, since the objective function depends on the current linearization, the value of the quality criterion may be worse than in the previous iteration.

3. $u_{ik}^{(s-1)}$ is an admissible and optimal solution of problem (4.29). The most favorable scenario. As in the previous case, $u^{(s-1)}(t)$ is the admissible control of original problem (4.15), but $u^{(s)}(t) = u^{(s-1)}(t)$. This guarantees that the same solution will be constructed in the next iterations, and, according to the stopping criteria, an exit from the loop will occur. It can be stated that an admissible suboptimal solution to problem (4.1), (1.2)–(1.5) is found.

4.5 Applications of Algorithms

Testing on a number of test problems showed the ability of the algorithm to reach an admissible suboptimal solution to a non-linear problem. We will demonstrate the properties of the algorithm using several examples.

The algorithms for control construction in program and positional modes were implemented in Python 3. The function *odeint* was used to solve the Cauchy problem, and *quad* was used to calculate definite integrals, both functions are provided in the library *scipy.integrate*. The search for optimal solutions for linear and quadratic programming problems was performed with

the library *coptpy* that accesses the COPT solver. The graphs were drawn using the package *matplotlib.pyplot*. The code of the program is given in Appendix B.

4.5.1 Scalar Non-Linear Problem

Consider the following optimal control problem:

$$\int_0^5 u^2(t)dt \longrightarrow \min_u,$$
$$\dot{x} = \frac{1}{x^2+1} + u,$$
$$x(0) = 0, \quad x(5) = 8,$$
$$0 \leqslant u(t) \leqslant 2, \quad N = 10.$$

We can easily verify that the derivative of the non-linear function

$$\frac{d}{dx}\left(\frac{1}{x^2+1}\right) = \frac{-2x}{(x^2+1)^2}$$

exists for any real value of x and is bounded modulo.

The program control for the presented problem was constructed using the algorithm outlined in 4.3.2. Twenty iterations of the algorithm were performed for the experiment. The computation time was 2.1 seconds. The objective function took a value of 10.3807.

The results are illustrated in the graphs. Fig. 4.2 shows the convergence of the algorithm in terms of stopping criteria (4.10). Fig. 4.3 demonstrates the tendency of the control function in different iterations of the algorithm to converge to a single function, and Fig. 4.4 shows the convergence of $x^{(s)}(t)$ to $\xi^{(s)}(t)$ as s increases.

The results of the algorithm testing on the selected task may include the following conclusions:

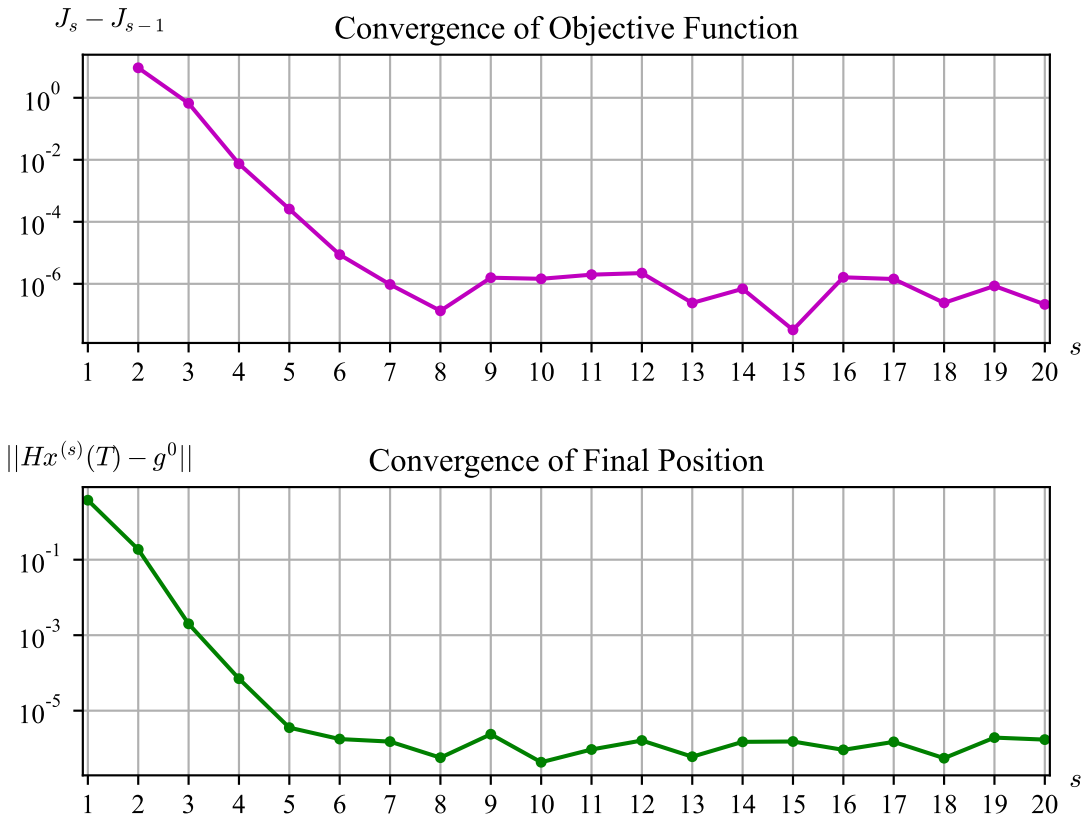

Figure 4.2 — Algorithm Convergence

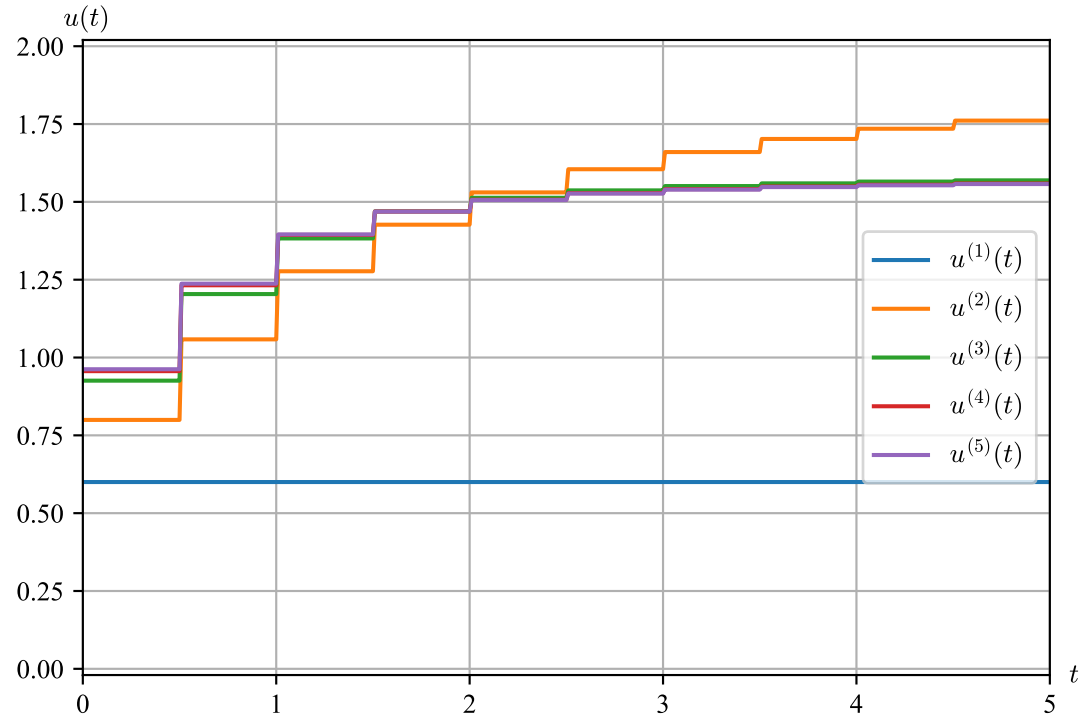

Figure 4.3 — Control Convergence

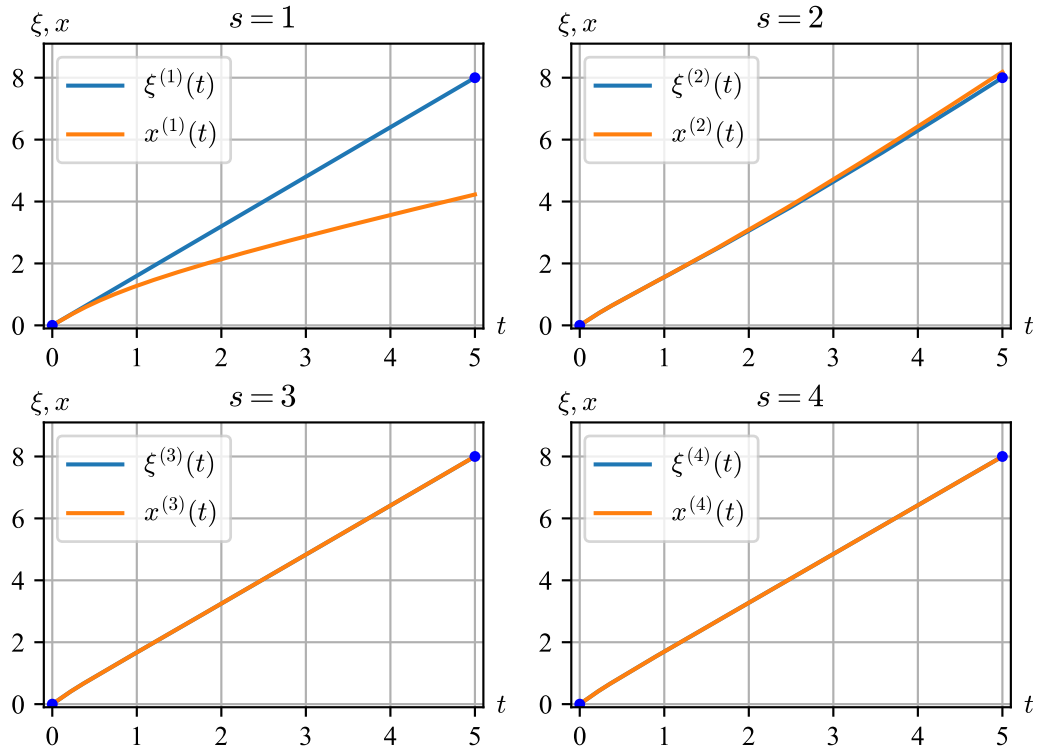

Figure 4.4 — Phase Variable Convergence

1. As a result of the algorithm, an admissible suboptimal solution to the non-linear optimal control problem was constructed.
2. After six iterations of the algorithm, the following values of the stopping criteria were obtained:

$$\left\| Hx^{(s)}(T) - g^0 \right\| = 1.77 \cdot 10^{-6},$$
$$|J_s - J_{s-1}| = 8.71 \cdot 10^{-6}.$$

   Further these values remained approximately at the same level, which indicates that the limit values were reached. A further descent to zero is impossible due to the presence of computational inaccuracies.
3. The control function practically stopped changing after six iterations. For example, $\delta_7 = 2.18 \cdot 10^{-4}$.
4. Despite the fact that the initial approximation was not very good and the final position of the object deviated from the required by almost two times, the solution of the closed non-linear system almost

converged to the corresponding solution of the auxiliary linear system already after 4 steps of the algorithm.

4.5.2 Pendulum Control

Consider the problem of damping of a mathematical pendulum.

$$\int_0^6 |u(t)|\,dt \longrightarrow \min_u,$$

$$\ddot{x} + \sin x = u,$$

$$-4 \leqslant u(t) \leqslant 4,$$

$$x(0) = \pi, \quad \dot{x}(0) = 1, \quad x(6) = \dot{x}(6) = 0.$$

The problem in this formulation was considered in [8]. Let us make some transformations and write the problem in a more convenient form:

$$\int_0^6 (v_1(t) + v_2(t))\,dt \longrightarrow \min_v,$$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\sin x_1 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix},$$

$$0 \leqslant v_1(t) \leqslant 4, \quad 0 \leqslant v_2(t) \leqslant 4,$$

$$x_1(0) = \pi, \quad x_2(0) = 1, \quad x_1(6) = x_2(6) = 0.$$

Calculate the Jacobian of the function $f(x_1, x_2) = (x_2, -\sin x_1)^T$:

$$F(x_1, x_2) = \begin{pmatrix} 0 & 1 \\ -\cos x_1 & 0 \end{pmatrix}.$$

Obviously, the matrix $F(x_1, x_2)$ is bounded for any x_1 and x_2.

First, let's analyze the results of control construction in the program mode. First, 10 steps of the algorithm were carried out at $N = 4$. The computation time was 3.58 seconds. Then the solution for $N = 20$ was

constructed. The 10 iterations took 7.42 seconds. For both variants, the deviation from the terminal condition stabilized at the fourth decimal place after 4–5 steps of the algorithm. This deviation is related to the given accuracy of calculating the solutions of the differential equations. If we increase the accuracy, we can achieve better convergence, but the total computation time will grow. The value of the objective function for $N = 20$ turned out to be about 24% better. For more details on the performance for different s, see Table 4.1.

Table 4.1 — Convergence of the algorithm for $N = 4$ and $N = 20$

N	s	J_s	$\left\|Hx^{(s)}(T) - g^0\right\|$	N	s	J_s	$\left\|Hx^{(s)}(T) - g^0\right\|$
4	1	8.40579	4.067779	20	1	7.39461	4.194625
4	2	4.97461	0.774187	20	2	4.20623	0.930559
4	3	4.26182	0.054745	20	3	3.22596	0.005873
4	4	4.22590	0.000955	20	4	3.22113	0.000569
4	5	4.22529	0.000424	20	5	3.22107	0.000614
4	6	4.22524	0.000376	20	5	3.22113	0.000680
4	7	4.22527	0.000393	20	7	3.22108	0.000633
4	8	4.22521	0.000321	20	8	3.22102	0.000562
4	9	4.22528	0.000362	20	9	3.22110	0.000559
4	10	4.22526	0.000387	20	10	3.22103	0.000585

Fig. 4.5 shows the phase trajectories of the non-linear and auxiliary linear systems at $N = 4$. The graphs show that at $s = 1$ the solution of the non-linear system at the final moment of time has significantly deviated from the origin of coordinates. However, at $s = 4$, the trajectory of the non-linear system can no longer be visually distinguished from the trajectory of the corresponding linear system.

To test the real-time control, let's perform an experiment: assume that, due to time constraints, it is possible to perform only one iteration of the algorithm at the beginning of motion and at the control switching points. Let's calculate the trajectory of motion for positional control. Fig. 4.6 shows the

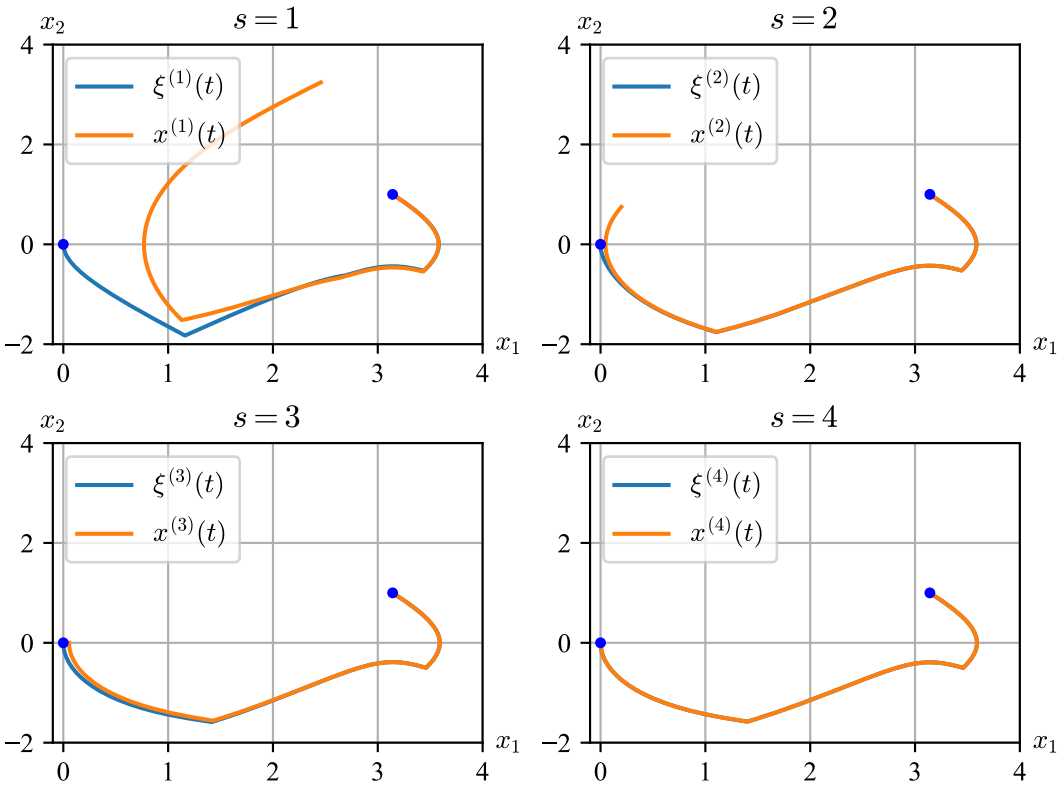

Figure 4.5 — Phase trajectories for different s at $N = 4$

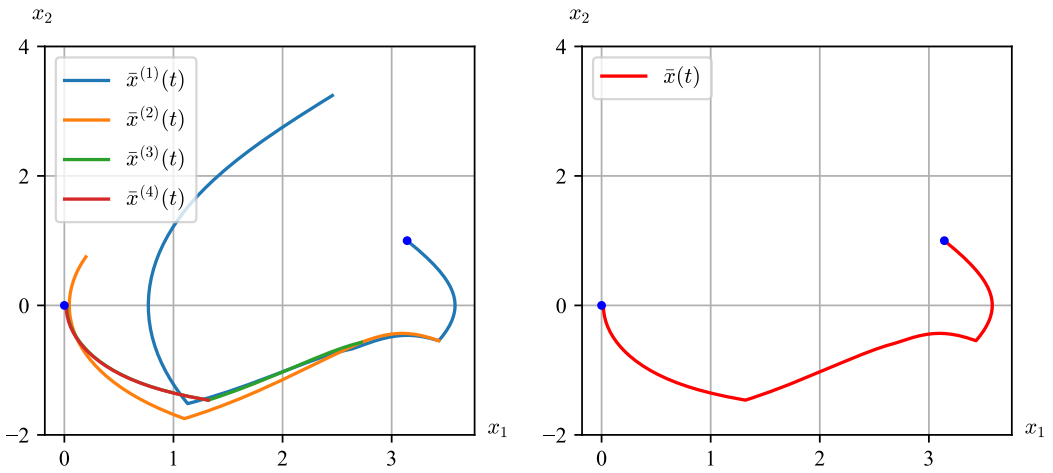

Figure 4.6 — Phase trajectories for real-time control mode at $N = 4$ and $S_l = 1$

phase trajectories. The graph on the left shows trajectories plotted at various points along the path of the object. Thus, at $t = 0$, the trajectory $\bar{x}^{(1)}(t)$ was calculated. The object moved along this trajectory until the moment $t = 1.5$, after which the trajectory $\bar{x}^{(2)}(t)$ was formed. At $t = 3$ the object moved to the trajectory $\bar{x}^{(3)}(t)$, and on the segment $[4.5, 6]$ it moved according to the function $\bar{x}^{(4)}(t)$. The final trajectory of the object's movement is shown in the graph on the right. As a result, the deviation of the final position from zero was 0.0163, and the objective function took a value of 4.4667.

Compared to the program mode, the deviation $\left\|H\bar{x}(T) - g^0\right\|$ was better than in the first three steps of the algorithm, and the value of the objective function surpassed the result of the first two iterations. Consequently, it can be concluded that with «limited resources» (small values of hyperparameters S_l and N), the solution in the program mode managed not to deviate strongly from the terminal constraint, but the generated positional control appeared to be non-optimal.

Table 4.2 — Real-time convergence at $S_l = 1$

N	S_l	J	$\left\|H\bar{x}(T) - g^0\right\|$
4	1	4.4667	$1.63 \cdot 10^{-2}$
10	1	3.4952	$1.62 \cdot 10^{-5}$
20	1	3.2475	$8.07 \cdot 10^{-6}$
40	1	3.2162	$1.45 \cdot 10^{-6}$
60	1	3.2130	$4.42 \cdot 10^{-7}$
100	1	3.2093	$1.87 \cdot 10^{-6}$

Let's explore the dependence of the convergence of solutions on the values of the hyperparameter N. Table 4.2 shows the results of the algorithm for different N with fixed $S_l = 1$. It is worth noting that in this configuration the final position of the object was closer to the desired one than in the program mode control, which can be explained by the recalculation of control in the last sections of motion in the proximity of the point $(0, 0)^T$. On the other hand,

the value of the objective function turned out to be slightly worse than the results in program mode at large s. Thus, at $N = 4$ the solution dropped by 9.8%, while at $N = 20$ the program mode result was exceeded by only 0.8%. The reason for this non-optimality lies in inaccurate approximations of the non-linear function at the beginning of the motion process. Nevertheless, based on this experiment, it can be assumed that the deviation can be leveled out by increasing the conversion points. Finally, it can be observed that the value of the objective function improves when the N increases up to 100 of switching points.

Table 4.3 — Real-time convergence at $N = 4$

N	S_l	J	$\left\|H\bar{x}(T) - g^0\right\|$
4	1	4.46672	$1.63 \cdot 10^{-2}$
4	2	4.30173	$1.38 \cdot 10^{-5}$
4	3	4.23098	$1.34 \cdot 10^{-5}$
4	4	4.22538	$1.39 \cdot 10^{-5}$
4	5	4.22534	$1.41 \cdot 10^{-5}$

Let's now vary the parameter S_l at $N = 4$ and record the results in Table 4.3. The obvious conclusion is that real-time control recalculation improves the results compared to the program mode control with similar values of S_l. For $S_l = 1$ the gain is 47%, for $S_l = 2$ — 13.5%, then continues to decrease. Since $S_l = 5$ there is no improvement due to the fact that the approximation of the non-linear function in the program mode is accurate and does not lead to deviations. In addition, the real-time computation of the control contributes to the reduction of the deviation from the desired position of the object.

As a result of the experiment we can formulate conclusions for the real-time control algorithm in addition to the conclusions for the program mode control given in 4.5.1:

1. Real-time control allows us to smooth out the approximation error of the non-linear function arising during the construction of the program control.

2. If an exact solution was originally constructed (in terms of criteria (4.10)), real-time recalculation of the control will not result in a significant improvement. A small gain is possible, but, in general, the same solution will be obtained.
3. If possible, the more correct approach is to initially calculate the exact solution by increasing S_l, rather than constructing an inexact program control with further recalculation.
4. Increasing the number of control switching and recalculation points N improves the value of the objective function and reduces the deviation from the terminal condition, but leads to a slower calculation.

4.6 Chapter 4 Conclusion

We considered an optimal control problem in which the system of ordinary differential equations describing the dynamics of motion of the controlled object contains non-linearity of the general form in phase variables. This problem is difficult to consider for two reasons:

- There is no analytical formula for solving a Cauchy problem for a system of this kind. As a consequence, it is impossible to reduce the original problem to some mathematical programming problem, as was done in Chapter 1 for a linear system.
- There are no effective algorithms for constructing sets of reachability and controllability for such a problem. Therefore, it is not possible to check whether there exists an admissible solution to the problem like the algorithm from Chapter 2 for the linear problem.

Approximate iterative algorithms for control construction in program and positional modes were proposed. The main idea of these algorithms is to linearize the non-linear function with a linear approximation along the

trajectory of the object. This approach makes it possible to avoid approximation error when moving in the proximity of a given trajectory. The algorithm for constructing the program mode consists in the consecutive recalculation of the object trajectory based on the trajectory information obtained at the previous iteration. Real-time control rebuilding is based on the same principles, but occurs as the motion progresses, based on the current position of the object.

Some useful properties of algorithms were formulated and proved in the form of lemmas. Thus, for the program mode, it was shown that when the control functions computed at different iterations tend to a common limit, the trajectory of object motion converges to an admissible trajectory. For the positional mode, it is proved that the constructed approximate solution converges to an admissible solution when the control recalculation points increase.

The algorithms were tested on two examples. The experiments showed good convergence of the algorithms. Even with an unsuccessful choice of the initial trajectory, the generated solutions quickly tended to be acceptable. The convergence of the objective function was also shown, which indicates, at least, the suboptimality of the solution. Additionally, the behavior of the positional mode construction algorithm was tested with different numbers of iterations of the algorithm and different numbers of control recalculation points. Test examples demonstrate the convergence of the trajectory generated under positional mode control to an admissible trajectory when the parameter N is increased.

Prospects for further study of the problem may be associated with further elaboration of the theoretical basis of algorithms, with the consideration of convergence conditions of controls and optimality criteria.

CONCLUSION

This paper presents methods of solving the optimal control problem in various formulations. The general idea is to reduce the initial problem to some mathematical programming problem. Thus, the basic model described in Chapter 1 is solved in two stages: first, the transition to a linear (quadratic) programming problem is performed, and then the search for the optimal solution of the resulting problem is carried out. This approach is reasonable because static optimization problems with constraints are well studied: solution algorithms with polynomial complexity have been derived for some classes, and there are many software packages for solution.

To summarize the results of this work, let's highlight the main achievements:

1. Algorithms for finding program and positional control for the optimal control problem with a non-linear system and a piecewise constant control subject to two-sided constraints have been developed. Both algorithms consist in the consecutive solution of the linearized problem. The program code of the algorithms is implemented.
2. Algorithms for construction of optimal control in a linear problem are presented for the choice of control in the class of piecewise linear and piecewise quadratic functions. In the first case the reduction to a linear (quadratic) programming problem is shown, in the second case — to a programming problem with quadratic constraints. The program code of solution of linear problem with piecewise constant, piecewise linear or piecewise quadratic control is implemented.
3. For a linear problem with piecewise constant control in the presence of convex and non-convex constraints an algorithm for the transition to a mixed integer linear (quadratic) programming problem is described.

4. A method for constructing sets of reachability and controllability for a linear problem with piecewise constant control and direct constraints on it is proposed. A theoretical proof of the method is given.

The proposed algorithms and their implementation in the form of program code can be applied to various applied tasks, as well as to create a software product that solves the general problem of optimal control.

Let's highlight the following tasks as areas for further development on this topic:

1. Consideration of wider classes of non-linearities in the right-hand side of a system of differential equations, including: functions with unbounded Jacobi matrix and functions with non-linearities on control.
2. Proof of optimality of calculated controls in a non-linear problem.
3. Construction of algorithms for finding control in a non-linear problem when selecting control in the class of piecewise linear and piecewise quadratic functions. This is a logical direction, since the solution algorithm consists of consecutive linearizations, and solution methods for linear models have already been derived for the mentioned classes.
4. Research on the reachability and controllability sets of a non-linear problem.

REFERENCES

1. Popkov A. S., Baranov O. V. On optimal control of the rotational movement of the electric motor shaft // Control processes and stability. 2014. Vol. 1, no. 1. P. 31–36. (in Russian)

2. Popkov A. S., Baranov O. V., Smirnov N. V. Application of adaptive method of linear programming for technical objects control // 2014 International Conference on Computer Technologies in Physical and Engineering Applications (ICCTPEA). 2014. P. 141–142.

3. Popkov A. S. Identification of dynamic input-output model for the russian economy and the optimal investment distribution // Control processes and stability. 2015. Vol. 2, no. 1. P. 696–701. (in Russian)

4. Popkov A. S., Smirnov N. V., Baranov O. V. Real-time quadrocopter optimal stabilization // «Stability and Control Processes» in Memory of V.I. Zubov (SCP), 2015 International Conference. 2015. P. 123–125.

5. Popkov A. S. Multicriteria Regulation of Investments in the Economy of the Russian Federation // Proceedings of the International Workshop on Applications in Information Technology (IWAIT-2015), The University of Aizu Press. 2015. P. 68–70.

6. Belousova M. V., Popkov A. S. The construction of dynamic input–output model based on the WIOD // Control processes and stability. 2016. Vol. 3, no. 1. P. 601–606. (in Russian)

7. Popkov A. S. Application of the adaptive method for optimal stabilization of a nonlinear object // 2016 International Conference Stability and Oscillations of Nonlinear Control Systems (Pyatnitskiy's Conference). 2016. P. 1–3.

8. Popkov A. S., Smirnov N. V., Smirnova T. E. On modification of the positional optimization method for a class of nonlinear systems // ACM International Conference Proceeding Series. 2018. P. 46–51.

9. Popkov A. S. Optimal control in non-convex sets // XV International Conference on Stability and Oscillations of Nonlinear Control Systems (Pyatnitskiy's Conference). 2020. P. 350–353. (in Russian)

10. Popkov A. S. Optimal program control in the class of quadratic splines for linear systems // Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes. 2020. Vol. 16, no. 4. P. 462–470.

11. Popkov A. S. Construction of reachability and controllability sets in a special linear control problem // Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes. 2021. Vol. 17, no. 3. P. 294–308.

12. The Certificate on Official Registration of the Computer Program № 2021662324 in Russian Federation. "A program for solving the problem of quadrocopter control using the adaptive Gabbasov method" (AdaptCopter) : № 2021615264 : req. 12.04.2021 : published 26.07.2021 / O. V. Baranov, A. S. Popkov, N. V. Smirnov ; applicant Federal State Budgetary Educational Institution of Higher Education "St. Petersburg State University". (in Russian).

13. Vinogradov I. M. Matematicheskaja jenciklopedija. Tom 3: Koordinaty – Odnochlen. M. : «Sovetskaja jenciklopedija», 1982. 592 p. (in Russian)

14. Kalman R. E. On the general theory of control systems // IFAC Proceedings Volumes. 1961. Vol. 1, no. 1. P. 521–547.

15. Pontrjagin L. S., Boltjanskij V. G., Gamkrelidze R. V., Mishhenko E. F. Matematicheskaja teorija optimal'nyh processov. M. : Nauka, 1969. 564 p. (in Russian)

16. Bellman R. E., Gliksberg I. L., Gross O. A. Some aspects of the mathematical theory of control processes. Santa Monica, CA : RAND Corporation, 1958. 263 p.

17. Zubov V. I. Matematicheskie metody issledovanija sistem avtomaticheskogo regulirovanija. L. : Mashinostroenie, 1974. 334 p. (in Russian)

18. Zubov V. I. Lekcii po teorii upravlenija. M. : Nauka, 1975. 600 p. (in Russian)

19. Kwakernaak H., Sivan R. Linear Optimal Control Systems. Wiley-Interscience, 1972. 608 p.

20. Trentelman H., Stoorvogel A., Hautus M. Control Theory for Linear Systems. University of Groningen, Johann Bernoulli Institute for Mathematics and Computer Science, 2002. 403 p.

21. Khlebnikov M. V., Shcherbakov P. S., Chestnov V. N. Linear-quadratic regulator. I. A new solution // Autom. Remote Control. 2015. Vol. 76, no. 12. P. 2143–2155. (in Russian)

22. Srochko V. A., Aksenyushkina E. V., Antonik V. G. Resolution of a Linear–quadratic Optimal Control Problem Based on Finite-dimensional Models // The bulletin of Irkutsk State university. Series «Mathematics». 2021. Vol. 37. P. 3–16. (in Russian)

23. Matveev A. S., Jakubovich V. A. Abstraktnaja teorija optimal'nogo upravlenija. SPb. : Izdanie S.-Peterburgskogo universiteta, 1994. 364 p. (in Russian)

24. Matveev A. S., Jakubovich V. A. Optimal'nye sistemy upravlenija: obyknovennye differencial'nye uravnenija. Special'nye zadachi: ucheb. posobie dlja vuzov. SPb. : Izd-vo SPb. gos. un-ta, 2003. 537 p. (in Russian)

25. Maciejowski J. M. Predictive control with constraints: Pearson Education Limited. Harlow, UK : Prentice-Hall, 2002. 352 p.

26. Propoi A. I. Application of linear programming methods for the synthesis of automatic sampled-data systems // Avtomat. i Telemekh. 1963. Vol. 24, no. 7. P. 912–920. (in Russian)

27. Rawlings J. B., Mayne D. Q. Model Predictive Control: Theory and Design. Madison : Nob Hill Pub, 2009. 533 p.

28. Ponomarev A. A. Suboptimal control construction for the model predictive controller // Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes. 2017. Vol. 13, no. 2. P. 193–208.

29. Faulwasser T., Findeisen R. Nonlinear Model Predictive Control for Constrained Output Path Following // IEEE Transactions on Automatic Control. 2016. Vol. 61, no. 4. P. 1026–1039.

30. Faulwasser T., Weber T., Zometa P., Findeisen R. Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot // IEEE Transactions on Control Systems Technology. 2017. Vol. 25. P. 1505–1511.

31. Balashevich N. V., Gabasov R., Kirillova F. M. Numerical methods for open-loop and closed-loop optimization of linear control systems // Computational Mathematics and Mathematical Physics. 2000. Vol. 40, no. 6. P. 799–819.

32. Al'sevich V. V., Gabasov R., Glushenkov V. S. Optimizacija linejnyh jekonomicheskih modelej. Minsk : Izd-vo BGU, 2000. 210 p. (in Russian)

33. Kantorovich L. V. Matematicheskie metody organizacii planirovanija proizvodstva. L. : LGU, 1939. 68 p. (in Russian)

34. Dantzig G. B. Linear Programming and Extensions. Santa Monica, CA : RAND Corporation, 1963. 641 p.

35. Dikin I. I. Iterative solution of problems of linear and quadratic programming // Dokl. Akad. Nauk SSSR. 1967. Vol. 174, no. 4. P. 747–748. (in Russian)

36. Evtushenko Y. G., Zhadan V. G. A relaxation method for solving problems of non-linear programming // USSR Computational Mathematics and Mathematical Physics. 1977. Vol. 17, no. 4. P. 73–87. (in Russian)

37. Karmarkar N. A New Polynomial-Time Algorithm for Linear Programming // Combinatorica. 1984. Vol. 4. P. 373–395.

38. Gabasov R., Kirillova F. M., Tjatjushkin A. I. Konstruktivnye metody optimizacii. Minsk : Izd-vo «Universitetskoe», 1984. Vol. 1. 216 p. (in Russian)

39. Khachiyan L. G. A polynomial algorithm in linear programming // Dokl. Akad. Nauk SSSR. 1979. Vol. 244, no. 5. P. 1093–1096. (in Russian)

40. Forsgren A., Gill P. E., Wong E. Active-set methods for convex quadratic programming // arXiv: Optimization and Control. 2015.

41. Monteiro R., Adler I. Interior path following primal-dual algorithms: Part II: Convex quadratic programming // Mathematical Programming. 1989. Vol. 44. P. 43–66.

42. Lobo M. S., Vandenberghe L., Boyd S., Lebret H. Applications of second-order cone programming // Linear Algebra and its Applications. 1998. Vol. 284, no. 1. P. 193–228.

43. Andersen E. D., Roos K., Terlaky T. On implementing a primal-dual interior-point method for conic quadratic optimization // Mathematical Programming. 2003. Vol. 95. P. 249–277.

44. Land A. H., Doig A. G. An Automatic Method of Solving Discrete Programming Problems // Econometrica. 1960. Vol. 28. P. 497–520.

45. Cornuéjols G. Revival of the Gomory cuts in the 1990's // Annals of Operations Research. 2007. Vol. 149, no. 1. P. 63–66.

46. Marchand H., Martin A., Weismantel R., Wolsey L. Cutting planes in integer and mixed integer programming // Discrete Applied Mathematics. 2002. Vol. 123, no. 1. P. 397–446.

47. Gilmore P. C., Gomory R. E. A Linear Programming Approach to the Cutting-Stock Problem // Operations Research. 1961. Vol. 9, no. 6. P. 849–859. Access mode: http://www.jstor.org/stable/167051.

48. Gilmore P. C., Gomory R. E. A Linear Programming Approach to the Cutting-Stock Problem – Part II // Operations Research. 1963. Vol. 11, no. 6. P. 863–888. Access mode: https://doi.org/10.1287/opre.11.6.863.

49. Desrosiers J., Lübbecke M. A Primer in Column Generation // Column Generation. 2006. 03. P. 1–32.

50. Dantzig G. B., Wolfe P. Decomposition Principle for Linear Programs // Operations Research. 1960. Vol. 8, no. 1. P. 101–111.

51. Desrosiers J., Lübbecke M. Branch-Price-and-Cut Algorithms // Wiley Encyclopedia of Operations Research and Management Science. 2011. 01.

52. Rothberg E. An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions // INFORMS Journal on Computing. 2007. 11. Vol. 19. P. 534–541.

53. Fischetti M., Lodi A. Local branching // Mathematical Programming. 2003. 09. Vol. 98. P. 23–47.

54. Danna E., Rothberg E., Pape C. Exploring relaxation induced neighborhoods to improve MILP solutions // Mathematical Programming. 2005. 01. Vol. 102. P. 71–90.

55. Decision Tree For Optimization Software. Access mode: http://plato.asu.edu/bench.html (online; accessed: 30.06.2022).

56. Gurobi — The Fastest Solver — Gurobi. Access mode: https://www.gurobi.com (online; accessed: 30.06.2022).

57. CPLEX Optimizer | IBM. Access mode: https://www.ibm.com/analytics/cplex-optimizer (online; accessed: 30.06.2022).

58. Cardinal Optimizer (COPT) — Cardinal Operations. Access mode: https://www.shanshu.ai/copt (online; accessed: 30.06.2022).

59. SCIP. Access mode: https://www.scipopt.org/index.php (online; accessed: 30.06.2022).

60. coin-or/Cbc: COIN–OR Branch–and–Cut solver. Access mode: https://github.com/coin-or/Cbc (online; accessed: 30.06.2022).

61. Corporation G. D. Solver Manuals. Access mode: https://www.gams.com/latest/docs/S_MAIN.html#SOLVERS_MODEL_TYPES (online; accessed: 10.08.2022).

62. Babadzhanjanc L. K., Potockaja I. J. Upravlenie po kriteriju rashoda v mehanicheskih sistemah. SPb. : Saint Petersburg State University, 2003. 137 p. (in Russian)

63. Formal'skij A. M. Upravljaemost' i ustojchivost' sistem s ogranichennymi resursami. M. : Nauka, 1973. 368 p. (in Russian)

64. Chernous'ko F. L. Ocenivanie fazovyh sostojanij dinamicheskih sistem. Metod jellipsoidov. M. : Nauka, 1988. 319 p. (in Russian)

65. Gayek J. Approximating reachable sets for a class of linear control systems // International Journal of Control. 1986. Vol. 43. P. 441–453. Access mode: https://doi.org/10.1080/00207178608933477.

66. Baier R., Büskens C., Chahma I. A., Gerdts M. Approximation of reachable sets by direct solution methods for optimal control problems // Optimization Methods and Software. 2007. Vol. 22, no. 3. P. 433–452.

67. Baier R., Matthias G. A computational method for non-convex reachable sets using optimal control // European Control Conference (ECC). 2009. P. 97–102.

68. Gusev M. I., Osipov I. O. Asimptotika mnozhestv dostizhimosti nelinejnyh upravljaemyh sistem na malyh promezhutkah vremeni // Dinamicheskie sistemy: ustojchivost', upravlenie, optimizacija. Materialy Mezhdunarodnoj nauchnoj konferencii pamjati professora R.F. Gabasova. 2021. P. 85–87. (in Russian)

69. Ekimov A. V., Balykina Y. E., Svirkin M. V. On the estimation of the attainability set of nonlinear control systems // AIP Conference Proceedengs. 2015. 03. Vol. 1648. P. 450008.

70. Polovinkin E. S., Balashov M. V. Jelementy vypuklogo i sil'no vypuklogo analiza. M. : FIZMATLIT, 2004. 416 p. (in Russian)

71. Dantzig G. B., Eaves B. C. Fourier–Motzkin elimination and its dual // J. Combinatorial Theory Ser. A. 1973. Vol. 14. P. 288–297.

72. Park J. J., Kuipers B. A smooth control law for graceful motion of differential wheeled mobile robots in 2D environment // IEEE International Conference on Robotics and Automation. 2011. P. 4896–4902.

73. Balashevich H. V., Gabasov R., Kirillova F. M. Algorithms for open-loop and closed-loop optimization of control systems with intermediate state contraints // Computational Mathematics and Mathematical Physics. 2001. Vol. 41, no. 10. P. 1410–1428. (in Russian)

74. Zhabko A. P., Chizhova O. N., Kotina E. D. Differencial'nye uravnenija i ustojchivost' / ed. by Cherezova N. V. SPb. : Lan', 2015. 320 p. (in Russian)

75. Balashevich H. V., Gabasov R., Kirillova F. M. Calculating an optimal program and an optimal control in a linear problem with a state constraint // Computational Mathematics and Mathematical Physics. 2005. Vol. 45, no. 12. P. 2030–2048. (in Russian)

76. Sargent R., Sullivan G. R. The Formulation of Optimal Control Problems as Nonlinear Programmes. 1977.

77. Bock H. G., Plitt K. J. A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems // IFAC Proceedings Volumes. 1984. Vol. 17, no. 2. P. 1603–1608.

78. Hargraves C., Paris S. Direct Trajectory Optimization Using Nonlinear Programming and Collocation // AIAA J. Guidance. 1987. Vol. 10. P. 338–342.

79. Garg D., Patterson M., Hager W., Rao A., Benson D. R., Huntington G. T. An overview of three pseudospectral methods for the numerical solution of optimal control problems. 2017.

80. Gabasov R., Kirillova F. M., Ruzhickaja E. A. Dempfirovanie i stabilizacija majatnika pri bol'shih nachal'nyh vozmushhenijah // Izvestija RAN. Teorija i sistemy upravlenija. 2001. no. 1. P. 29–38. (in Russian)

81. Balashevich N. V., Gabasov R., Kalinin A. I., Kirillova F. M. Optimal control of nonlinear systems // Computational Mathematics and Mathematical Physics. 2002. Vol. 42, no. 7. P. 931–956.

82. Zgonnikov A. V. Sintez optimal'noj obratnoj svjazi v odnoj nelinejnoj mehanicheskoj sisteme // Processy upravlenija i ustojchivost': Trudy 40-j mezhdunarodnoj nauchnoj konferencii aspirantov i studentov. 2009. P. 21–26. (in Russian)

83. Gabasov R., Kirillova F. M., Al'sevich V. V., Kalinin A. I., Krahotko V. V., Pavljonok N. S. Metody optimizacii: posobie. Minsk : Izdatel'stvo «Chetyre chetverti», 2011. 472 p. (in Russian)

84. Zubov V. I. Sintez mnogoprogrammnyh ustojchivyh upravlenij // Dokl. AN SSSR. 1991. Vol. 318, no. 2. P. 274–277. (in Russian)

85. Smirnov N. V., Solov'eva I. V. Application of the positional optimization method for the multiprogrammed stabilization of the bilinear systems // Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes. 2009. no. 3. P. 251–259.

86. Smirnov N. V. Multiprogram Control for Dynamic Systems: A Point of View // Proceedings of the 2012 Joint International Conference on Human–Centered Computer Environments. New York, NY, USA : Association for Computing Machinery. 2012. HCCE 12. P. 106–113. Access mode: https://proxy.library.spbu.ru:2060/10.1145/2160749.2160773.

87. Andersson J. A. E., Gillis J., Horn G., Rawlings J. B., Diehl M. CasADi – A software framework for nonlinear optimization and optimal control // Mathematical Programming Computation. 2019. Vol. 11, no. 1. P. 1–36.

88. Houska B., Ferreau H. J., Diehl M. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization // Optimal Control Applications and Methods. 2011. Vol. 32, no. 3. P. 298–312.

89. DIDO — Optimal control software — Third-Party Products & Services — MATLAB & Simulink. Access mode: https://www.mathworks.com/products/connections/product_detail/dido.html (online; accessed: 09.08.2022).

90. Lizorkin P. I. Kurs differencial'nyh i integral'nyh uravnenij s dopolnitel'nymi glavami analiza. M. : Nauka, 1981. 384 p. (in Russian)

91. GitHub repository — piecewise_control: Algorithm of constructing optimal solution for linear optimal control problem in the class of piecewise constant, piecewise linear or piecewise quadratic functions. Access mode: https://github.com/alexander-popkov/piecewise_control (online; accessed: 11.08.2022).

92. GitHub repository — nonlinear_control: Algorithms for construct program and positional solutions in nonlinear optimal control problem with piecewise constant control. Access mode: https://github.com/alexander-popkov/nonlinear_control (online; accessed: 11.08.2022).

APPENDIX A

Program Code for Solving Linear Optimal Control Problem in Different Classes of Control

This appendix presents a description of the program code structure of the algorithm for solving a linear optimal control problem in the class of piecewise constant, piecewise linear, or piecewise quadratic functions. The code is uploaded to public repository [91].

The software package consists of the following scripts:

- *run.py*: a function that starts the algorithm, here the control class is selected.
- *problem/ProblemStatement.py:* formulation of the optimal control problem with a linear system, setting the parameters of the model.
- *control/PiecewiseConstantControl.py:* construction of optimal piecewise constant control. The initial problem is reduced to a linear or quadratic programming problem. Then the solution of the resulting problem is calculated using the COPT solver (*coptpy* library).
- *control/PiecewiseLinearControl.py:* construction of optimal piecewise linear control. The initial problem is reduced to a linear or quadratic programming problem. Then the solution of the resulting problem is calculated using the COPT solver (*coptpy* library).
- *control/QuadraticSplineControl.py:* construction of optimal piecewise quadratic control. The original problem is reduced to a convex programming problem with quadratic constraints. Then the solution of the resulting problem is computed using the COPT solver (*coptpy* library).

- *movement/PCCObjectMovement.py:* calculation of the trajectory of object motion as a solution of a system of differential equations closed by piecewise constant control.
- *movement/PLCObjectMovement.py:* calculation of the trajectory of object motion as a solution of a system of differential equations, closed by piecewise linear control.
- *movement/QSCObjectMovement.py:* calculation of the trajectory of object motion as a solution of a system of differential equations closed by piecewise quadratic control.
- *numerical/DefiniteIntegral.py:* solution of the Cauchy problem with a linear homogeneous system using the function *odeint* from the package *scipy.integrate*.
- *numerical/DifferentialEquation.py:* calculation of a definite integral with the function *quad* from the package *scipy.integrate*.
- *numerical/Interpolation.py:* interpolation of a table-defined function with cubic splines using the function *interp1d* from the package *scipy.interpolate*.
- *plotting/Plotting.py:* functions for plotting graphs. Tools of the *matplotlib.pyplot* library are used.

APPENDIX B

Program Code for Solving Non-Linear Optimal Control Problem in Program and Positional Modes

This appendix presents a description of the structure of the program code of the algorithm for solving a non-linear optimal control problem in the class of piecewise constant functions in the program and positional modes. The code is available in public repository [92].

The software package consists of the following scripts:

- *run_program_control.py:* a function that starts the algorithm for calculating the program control for a non-linear problem. Here the number of iterations of the algorithm S_l is set.
- *run_positional_control.py:* a function that starts the algorithm for calculating the positional control for a non-linear problem. The number of iterations of the algorithm S_l is determined.
- *task/Task.py:* a class in which the non-linear optimal control problem and the main parameters of the model are specified.
- *task/nonlinear_scalar_task.py:* implementation of the *Task* class as the non-linear model presented in 4.5.1.
- *task/pendulum_control.py:* implementation of the *Task* class as the pendulum model given in 4.5.2.
- *problems/NonLinearProblem.py:* a class for working with a non-linear model. The functions of linearization of the non-linear right part and calculation of the trajectory of object motion as a solution of the Cauchy problem with the non-linear system, closed by the found control, are implemented here.
- *problems/LinearProblem.py:* a class for working with a linear model. Here the reduction to a linear or quadratic programming problem is

carried out. For this purpose, the functions of solving a system of differential equations (*scipy.integrate.odeint*), calculating a definite integral (*scipy.integrate.quad*) and constructing a matrizant are implemented.

- *lin_prog_task/LPTask.py:* a class in which the parameters of the interval linear or quadratic programming problem are stored.
- *lin_prog_model/LPModel.py:* a class that works with a linear (quadratic) programming problem. With the library *coptpy* the model is initialized, variables, constraints and objective function are set. Then, using the COPT solver, the optimal solution is searched for.
- *common/common.py:* a number of auxiliary functions.
- *plotting/plotting.py:* plotting using the *matplotlib.pyplot* package in order to demonstrate how the algorithm works.