1 Обзор методов планирования траекторий мобильных робототехнических систем

Для начала рассмотрим задачу планирования траекторий движения мобильных робототехнических систем. Данная область знания развивается уже сравнительно давно и имеет в своем арсенале достаточно богатый инструментарий для решения практических задач [48]. Первоначально, работы в данном направлении были сосредоточены на методах анализа и синтеза траекторий для автоматизированных станков на производстве. Совместно с системами числового программного управления данные разработки позволили поднять эффективность и точность обработки деталей. Следующим значимым этапом стало бурное развитие промышленной робототехники. Ярким представителем данного класса устройств стали роботы-манипуляторы, имеющие сложные кинематические схемы с множеством степеней свободы. Эффективное использования таких систем потребовало новых подходов и дало сильный толчок к дальнейшим исследованиям, которые продолжаются до сих пор. Вообще говоря, сам термин "планирование траекторий" в современном сознании связан в первую очередь с манипуляционными роботами [20, 68, 70].

Отдельным направлением развития стали методы планирования траекторий для мобильных систем. Одной из самых значимых работ в данном направлении была статья [30], в которой Л.Дубинс поставил задачу синтеза C^1-гладкой траектории, наложив ограничение на максимальную кривизну. Позже эта задача была сформулирована как задача оптимального управления, решение которой имеет ясный геометрический смысл, что оптимальная по скорости траектория между двумя произвольными положениями на плоскости строится из отрезков прямых и дуг окружностей. Этот результат сильно повлиял на дальнейшее развитие исследований. Несмотря на детальную проработанность данной задачи, современные исследователи по-прежнему в своих работах используют разнообразные вариаций данного подхода ввиду его простоты, ясности и содержательности [3, 15, 23, 36, 61, 68, 72].

Если сравнивать представленные выше направления в методологии планирования траекторий, то становится очевидным, что несмотря на общую основу, методы планирования пути движения для мобильных систем имееют свою специфику. Размеры рабочей области роботов-манипуляторов, автоматизированных обрабатывающих станков и прочего технологического оборудования ограниченны кинематическими схемами и геометрическими размерами самих устройств, в то время как для мобильных систем такими ограничениями являются энергетические и коммуникационные ресурсы, позволяющие работать на расстояниях в десятки километров. Исходя из этого, за исключением специфических

задач, для мобильных систем участки прямолинейного движения преобладают над участками маневрирования, поэтому траектория следования задается с помощью реперных точек (way points в английской терминологии), которые, как правило, привязываются к географическим координатам. Необходимо заметить, что для сухопутных роботов, работающих в ограниченном пространстве, все возможные трассы следования можно задать заранее, с помощью специальной разметки или кабельных линий, что превращает задачу в чисто инженерно-техническую.

Как было указано выше, основным методом задания пути движения для мобильных систем являются реперные точки. В свою очередь, задача планирования траектории ставиться, как задача синтеза совокупности функций, геометрически связывающих реперные точки в определённом порядке. Естественно, что в для такой постановки, было разработано множество разнообразным методов решения. Чтобы как-то охарактеризовать данные методы можно начать с конечного вида получаемого результата. В зависимости от постановки задачи, можно получить аналитическое представление траектории в виде одной гладкой функции или в виде кусочно-гладкой. Данное разделение является весьма условным, так как для упрощения описания получившегося решения можно декомпозировать большую и сложную исходную задачу на совокупность простых подзадач и получить результирующее описание в виде кусочно-гладкой функции, но в данном контексте имеется ввиду, что кусочно-гладкое представление характеризуется тем, что анализируются именно участки или между заданными реперными точками, которые формируют "куски"результирующей функции. Если класс функций, применяемых для описания пути движения ограничен, то можно назвать данный подход как метод планирования траектории на основе базовых элементов. Забегая вперёд, можно обозначить, что в данной работе исследуется именно задача синтеза кусочно-гладкого описания в классе неявных функций. Метод представления пути движения в форме единой функции, в свою очередь, подразделяется на две подгруппы: подходы основанные на интерполяции [22, 25, 36, 44, 45, 69] и подходы основанные на аппроксимации [2, 16, 35, 57, 66, 77]. Исходя из названий, основное различие в этих подходах заключается в том, что проходит или нет результирующая траектория через реперные точки или нет. Как правило, в интерполяционном подходе используют разнообразные сплайны не менее третьей степени. В апроксимационных методиках используются как сплайны, так и кривые Безье. В апроксимационных можно отдельно выделить работы [10, 18], в которых авторы работают с неявным представление кривых. Это можно объяснить тем, что в этих работах синтезируются законы управления на основе методологии стабилизации геометрических многообразий в выходном пространстве объектов управления.

Второй подход, основанный на синтезе гладких участков между заданны-

ми реперными точками, или метод планирования траектории на основе базовых элементов в свою очередь делится на подгруппы, исходя из разнообразия типов функций, используемых при синтезе. Наиболее простым случаем является использование для соединения реперных точек отрезков прямых [23, 34, 36, 72]. Данный подход отличается предельной простотой, но формирует лишь C^0-гладкую кривую. Более совершенным методом планирования траектории является использование отрезков прямых и дуг окружностей (см. рис. 1) [7, 8, 23, 36, 42, 43, 47, 61, 72].

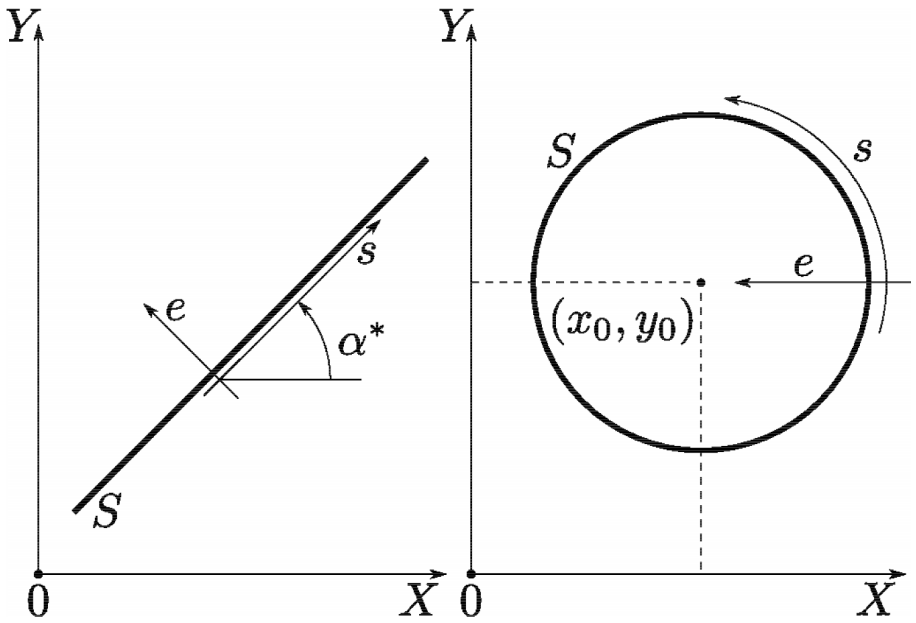

Рис. 1: Базовые элементы.

Данный подход позволяет синтезировать C^1-гладкую кривую (см. рис. 2) и обеспечивает лучшую точность следования вдоль заданного маршрута, но попрежнему, остается физически нереализуемой, вследствие скачка кривизны при переходе от прямого участка к круговому.

Несмотря на это, данный подход получил самое широкое распространение на практике вследствие своей простоты. Для того, чтобы синтезировать траекторию движения с гладкостью C^2 и выше отрезки прямых и дуги окружностей дополняют участками перехода, в виде отрезков разнообразных спиралей [24, 28, 38, 51, 52, 67, 69, 74, 75]. Также для этих целей возможно использования сплайнов или кривых Безье [2, 22, 35, 69, 77]. Необходимо обратить внимание на следующий нюанс. Как правило, за исключением редких случаев, траектории синтезируются в виде параметрических функций. Такой подход численно эффективен и обеспечивает компактную форму представления, но появляется серьёзный недостаток с точки зрения синтеза алгоритмов управления а именно сопоставление своей позиции в пространстве с позицией на кривой. На

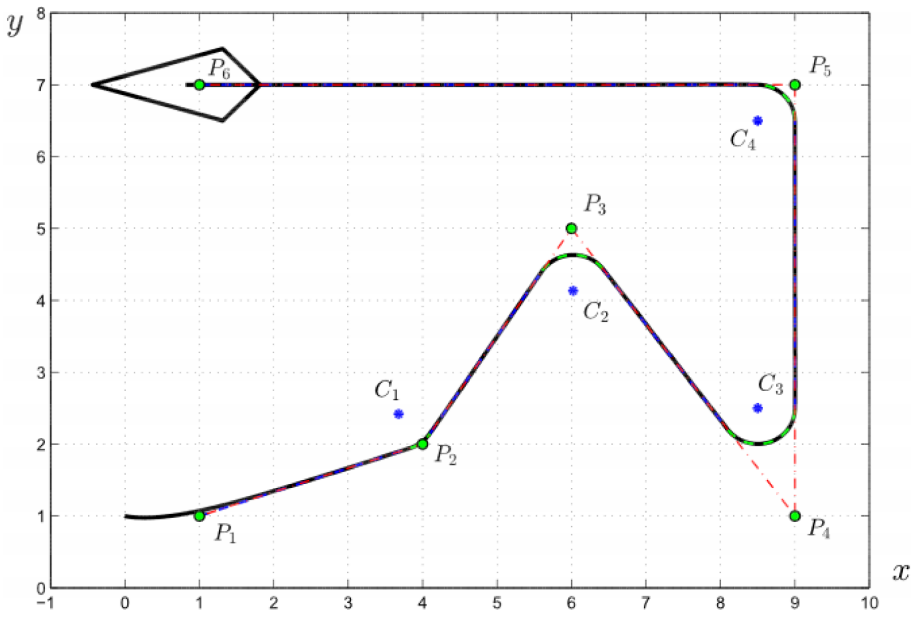

Рис. 2: C^1-гладкая траектория.

практике это решается тем, что вдоль траектории движения запускается виртуальная цель("virtual target"), к которой притягивается объект управления или же использование разнообразных методов оптимизации для поиска ближайшей точки. С этой позиции намного удобнее использовать для задании траектории неявные функции, которые образуют линии уровня в рабочем пространстве и мы всегда знаем нашу текущую позицию относительно заданной кривой, но, сложность аналитического представления неявных кривых произвольного вида сильно ограничивают использование этого метода на практике.

2 Планирование траекторий движения мобильных робототехнических систем

Выше были рассмотрены широко используемые методы планирования траекторий мобильных робототехнических систем. Проанализированы методологические подходы, их достоинства и недостатки. Данная глава посвящена одному из широко используемых на практике методу планирования траектории на основе базовых элементов. Он позволяет построить траекторию движения мобильных робототехнических систем достаточной гладкости, обеспечивая при этом простоту аналитического представления. Вследствие рассматриваемого подхода к синтезу траекторных алгоритмов управления на основе стабилизации многообразий в выходном пространстве объекта управления задача планирования траектории формулируется как задача поиска таких многообразий из класса

неявных функций. Рассмотрен метод планирования траекторий движения робототехнических систем на основе базовых элементов, обеспечивающих второй порядок гладкости, представленных в виде неявных функций, таких как отрезок прямой, дуга окружности и участки перехода между ними в виде ветвей кубической параболы.

2.1 Постановка задачи

Для начала рассмотрим случай планирования пути на плоскости. Будем считать, что траектория движения задается с помощью реперных точек $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, ..., $p_n = (x_n, y_n)$ в правосторонней декартовой системе координат $X_I Y_I$ (см. рис. 3).

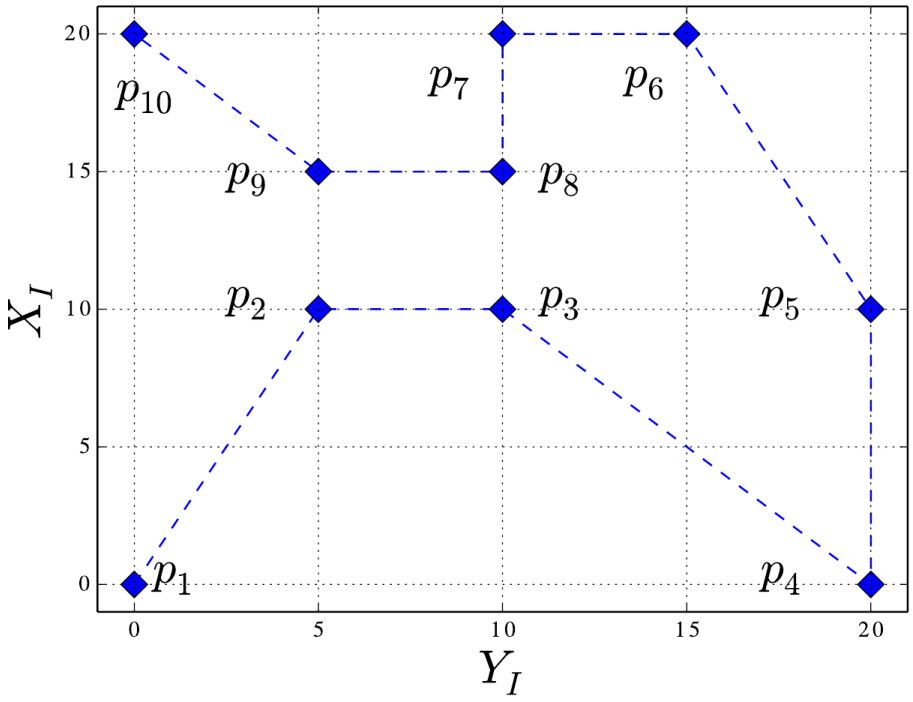

Рис. 3: Пример задания траектории с помощью реперных точек $p_1, p_2, \ldots, p_{10}$.

Обозначим через букву S итоговую траекторию движения, построенную на основе заданных реперных точек. Определим максимальное условие на кривизну траектории $\xi(x, y)$ в виде

$$\xi\,(x, y) \leq \xi_{\max} = \frac{1}{R_{\max}}, \tag{1}$$

где $\xi_{\max}$ – максимально допустимая кривизна траектории, определяемая физическими особенностями объекта управления и $R_{\max}$ – соответствующий максимальный радиус кривизны.

Таким образом можно сформулировать задачу:

Задача 2.1. Необходимо найти C^2-гладкое аналитическое описание траектории движения мобильной робототехнической системы на плоскости S, описываемое в терминах неявных функций с и ограничением на максимальную кривизну траектории (1).

Необходимо отметить, что данная постановка задачи рассмотрена во многих работах и хорошо изучена. Новизна данного исследования заключается в том, что результирующее описание должно быть представлено в виде неявных функций. Вообще говоря, существует несколько подходов для перехода от параметрического вида к неявному изложенные в работах [26, 29, 40, 65, 71, 76]. Одним из таких методов, получивший широкое распространение, является метод с использования матрицы Сильвестра [76].

Данная задача формулируется следующим образом. Пусть задано параметрическое представление кривой на плоскости заданное с помощью двух полиномов степени $n \geq 1$ и $m \geq 1$:

$$x(s) = a_n s^n + \ldots + a_1 s + a_0, \tag{2}$$

$$y(s) = b_m s^m + \ldots + b_1 s + b_0, \tag{3}$$

где s – параметр. Тогда неявное описание может быть найдено из вычисления выражения

$$\det S = 0, \tag{4}$$

где S – матрица Сильвестра вида

$$S = \begin{bmatrix} a_n & \ldots & a_1 & a_0 - x & 0 & \ldots & \ldots & 0 \\ 0 & a_n & \ldots & a_1 & a_0 - x & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & 0 & a_n & \ldots & a_1 & a_0 - x \\ b_m & \ldots & b_1 & b_0 - y & 0 & \ldots & \ldots & 0 \\ 0 & b_m & \ldots & b_1 & b_0 - y & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & 0 & b_m & \ldots & b_1 & b_0 - y \end{bmatrix}. \tag{5}$$

Это достаточно удобный метод получения неявного описания кривых, но для высоких порядков n и m получаются достаточно сложные и громоздкие выражения, которые сильно ограничивают данный метод для практических применений, поэтому наши дальнейшие рассуждения будут посвящены методу планирования траектории на основе базовых элементов, позволяющему получить достаточно компактное представление и при этом обеспечить выполнение поставленной задачи.

2.2 Планирование траектории на основе базовых элементов

Мы начнем рассмотрение предлагаемого методики с рассмотрения простейшего случая планирования траектории на основе отрезков прямых и затем будем постепенно вводить новые базовые элементы. Как было сказано выше, исходные реперные точки $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, ..., $p_n = (x_n, y_n)$ заданы в правосторонней декартовой системе координат $X_I Y_I$. Для рассмотрения методики соединения точек нам не нужно использовать для анализа все точки сразу, а достаточно рассмотреть три ближайших. Результирующая задача в свою очередь сводится к итеративному обходу всех точек по тройка и "склеивание"результатов. Назовём данную подзадачу как "задача о трёх точках"(см. рис.4).

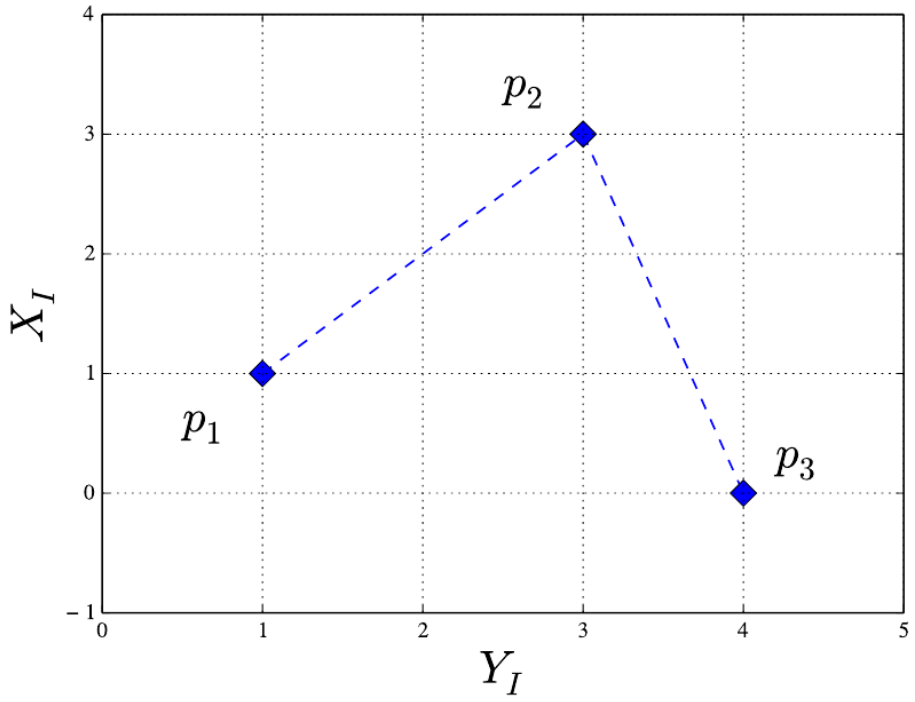

Рис. 4: Исходная "задача о трёх точках".

Для упрощения дальнейших рассуждений введём в рассмотрение матрицу поворота $R_I^P \in SO(2)$, описывающая переход из глобальной системы координат $X_I Y_I$ в локальную, задачно-ориентированную $X_P Y_P$, связанную с точкой $p_1 = (x_1, y_1)$ в виде

$$R_I^P(\psi) = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix},$$

где ψ – угловая ориентация первого участка пути $\left[p_1,\ p_2\right]$, которая находится

из выражения

$$\psi = \arctan 2 \left(\frac{y_2 - y_1}{x_2 - x_1} \right).$$

Матрицу обратного перехода из $X_P Y_P$ в $X_I Y_I$ определим как $R_P^I = \left(R_I^P\right)^T$.

Таким образом, переход из глобальной системы координат $X_I Y_I$ в локальную $X_P Y_P$ для точек $p_i, i = \overline{1,3}$ определяется как

$$p_{iP} = R_I^P(\psi)\left(p_i - p_1\right), i = \overline{1,3}. \tag{6}$$

Трансформированная "задача о трёх точках"представлена на рисунке 5.

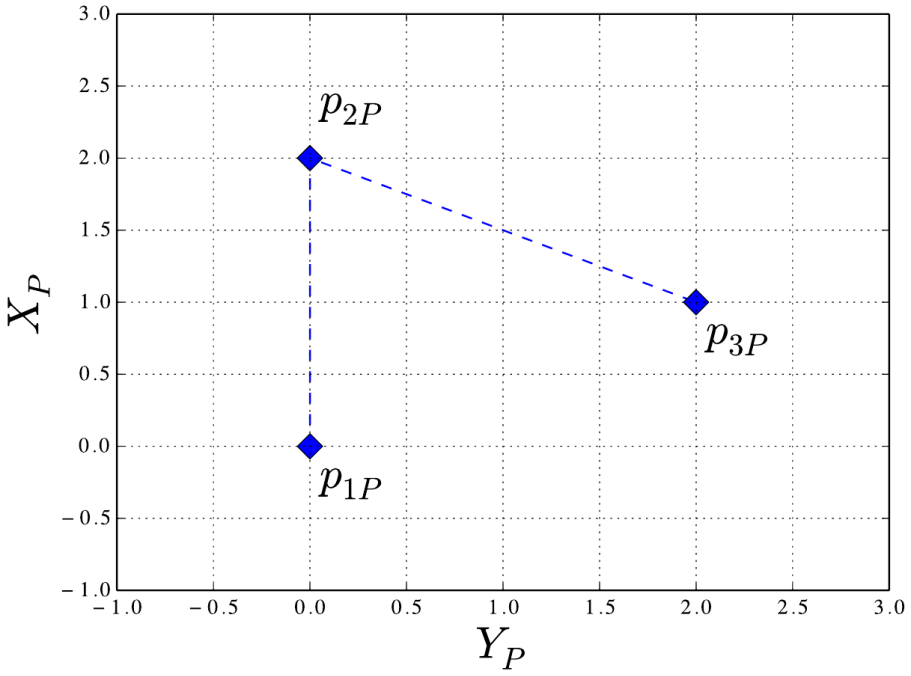

Рис. 5: Трансформированная "задача о трёх точках".

После преобразования нетрудно заметить, что неявное уравнение отрезка прямой, соединяющей точки может быть описано простым выражением в локальной системе координат:

$$Y_P = 0.$$

Это же выражение с учётом преобразования (6) для абсолютной системы координат примет вид

$$\begin{bmatrix} -\sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} x - x_1 \\ y - y_1 \end{bmatrix} = 0.$$

Таким образом получаем, что результирующая траектория S имеет вид

$$S : \bigcup_{i=1}^{n-1} \begin{cases} -\sin\psi_i \left(x - x_i\right) + \cos\psi_i \left(y - y_i\right) = 0, \\ \psi_i = \arctan\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right). \end{cases} \tag{7}$$

Здесь необходимо остановить, чтобы рассмотреть ещё один вопрос. Каким образом регулятор будет переключаться с одного участка на другой. Для параметризованных траекторий возможно использовать значение параметра. Тогда логика может выглядеть, к примеру, таким образом, что если параметр достиг определённого значения, то перейти к следующему участку. Для неявных кривых эта техника не очень подходит, так как для её использования становится необходимым ввести в рассмотрение функцию, имеющая смысл координаты вдоль пути следования, что не всегда можно легко сделать. Наиболее приемлемыми альтернативами являются методы, предложенные в работе [23]:

1. Метод шаровой окрестности

2. Метод полуплоскости

Метод шаровой плоскости заключается в том, что вокруг реперной точки, следующей по направлению движения строится окрестность заданного радиуса r_{sw}. Переключение на следующий участок происходит при попадании объекта управления внутрь окрестности, то есть, когда

$$(x - x_{wp})^2 + (y - y_{wp})^2 \leq r_{sw}^2,$$

где x, y – текущее положение робота, x_{wp}, y_{wp} – положение реперной точки, вокруг которой строится окрестность. Данный метод достаточно прост и был реализован в работах [7, 8, 47], но у него есть существенный недостаток. Если объект управления окажется на расстоянии большем, чем заданный радиус r_{sw}, то переключения не произойдёт. Чтобы избежать такой проблемы был разработан второй подход - метод полуплоскости. Он заключается в том, что относительно точки переключения строится неявная прямая $Q(x, y)$, разделяющая всю плоскость на две части. Принадлежность к той или иной части можно определить подставив текущие координаты в неравенство $Q(x, y) \leq 0$. Таким образом в любой точке плоскости известно относительное положение объекта управления и кривой переключения и переключение произойдёт в любом случае.

На основе этого дополним описание кривой (7) алгоритмом переключения между отдельными участками. Получаем следующее выражение

$$S : \bigcup_{i=1}^{n-1} \begin{cases} -\sin\psi_i (x - x_i) + \cos\psi_i (y - y_i) = 0, \\ Q_i(x, y) = \cos\psi_i (x - x_i) + \sin\psi_i (y - y_i) - r_i, \\ i = i + 1, \text{ when } Q_i(x, y) > 0, \\ \psi_i = \arctan\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right), \\ r_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}. \end{cases} \tag{8}$$

Далее дополним описание кривой участками состоящими из дуг окружностей. Неявное уравнение окружности радиуса R с центром в точке x_{c0}, y_{c0} задается уравнением вида

$$(x - x_{c0})^2 + (y - y_{c0})^2 - R^2 = 0. \tag{9}$$

Таким образом для дополнения описания (8) дугами окружностей нам необходимо определить радиус, координаты центра и, в соответствии с выбранной методикой, плоскости переключений. Допустим, что радиус выбирается вручную с учётом ограничения (1), т.е. $R \leq R_{\max}$. Координаты центра в "задаче о трех точках"можно легко определить с помощью простейших геометрических расчетов. Получаются следующие выражения:

$$\begin{bmatrix} x_{c0} \\ y_{c0} \end{bmatrix} = \begin{bmatrix} x_{1P} \\ y_{1P} \end{bmatrix} + R_I^P(\delta) \begin{bmatrix} d_{cc} \\ 0 \end{bmatrix}, \tag{10}$$

$$d_{cc} = \left| \frac{R}{\sin \frac{\sigma}{2}} \right|, \tag{11}$$

$$\sigma = \begin{cases} \pi - \arctan 2\left(\frac{y_{3P}-y_{2P}}{x_{3P}-x_{2P}}\right), \arctan 2\left(\frac{y_{3P}-y_{2P}}{x_{3P}-x_{2P}}\right) > 0 \\ -\pi - \arctan 2\left(\frac{y_{3P}-y_{2P}}{x_{3P}-x_{2P}}\right), \arctan 2\left(\frac{y_{3P}-y_{2P}}{x_{3P}-x_{2P}}\right) \leq 0 \end{cases}, \tag{12}$$

$$\delta = \pi - \frac{\sigma}{2}. \tag{13}$$

В результате этих вычислений получаются координаты центра окружности радиусом R, вписанной в угол между соседними участками. Теперь необходимо найти выражения для плоскостей переключения, которые обозначим буквами $Q_1(x, y)$ и $Q_2(x, y)$. В дальнейшем для компактности записи будем опускать аргументы, подразумевая, что они присутствуют. Для этого необходимо найти расстояния от точки p_2 до точек касания окружности, которое вычисляется с помощью выражения

$$d = \left| \frac{R}{\tan \frac{\sigma}{2}} \right|. \tag{14}$$

Пример вышеприведённых расчетов со значением $R = 0.8$ и вычисленной точкой цента окружности $p_{c0} = (x_{c0}, y_{c0})$ для исходной задачи представлен на рисунке 6.

Теперь можно расширить исходное функциональное описание (8), учитывая

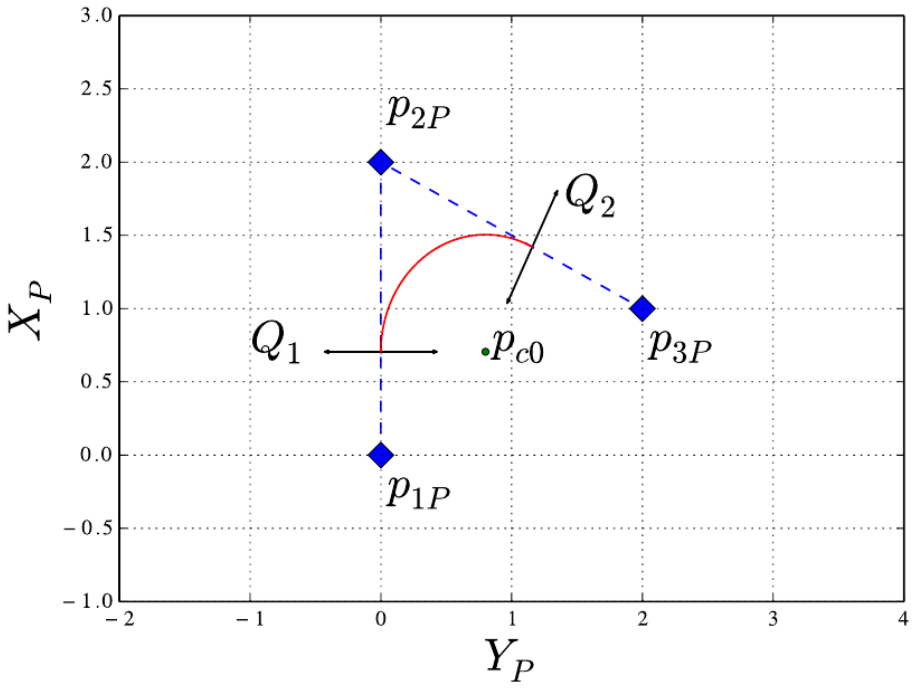

Рис. 6: Сглаживание траектории с помощью окружности.

(9)-(14).

$$
S : \bigcup_{i=1}^{n-1} \begin{cases}
-\sin\psi_i\,(x-x_i)+\cos\psi_i\,(y-y_i)=0, \text{ when } Q_{1i}\le 0,\\
(x-x_{ci})^2+(y-y_{ci})^2-R^2=0, \text{ when } Q_{2i}\le 0,\\
\begin{bmatrix} x_{ci}\\ y_{ci}\end{bmatrix}=\begin{bmatrix} x_{i+1}\\ y_{i+1}\end{bmatrix}+R_P^I(\psi_i)\,R_I^P(\delta_i)\begin{bmatrix} d_{ci}\\ 0\end{bmatrix},\\
d_{ci}=\left|\frac{R}{\sin\frac{\sigma}{2}}\right|,\\
\delta_i=\pi-\frac{\sigma_i}{2},\\
\sigma_i=\begin{cases}\pi-(\psi_{i+1}-\psi_i), \text{ when } (\psi_{i+1}-\psi_i)>0\\ -\pi-(\psi_{i+1}-\psi_i), \text{ when } (\psi_{i+1}-\psi_i)\le 0\end{cases},\\
Q_{1i}=\cos\psi_i\,(x-x_i)+\sin\psi_i\,(y-y_i)-r_i+d_i,\\
Q_{2i}=\cos\psi_{i+1}\,(x-x_i)+\sin\psi_{i+1}\,(y-y_i)-d_i,\\
i=i+1, \text{ when } Q_{2i}>0,\\
\psi_i=\arctan 2\left(\frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right),\\
r_i=\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2},\\
d_i=\left|\frac{R}{\tan\frac{\sigma}{2}}\right|.
\end{cases}
\tag{15}
$$

В результате работы алгоритма (15) получается C^1-гладкая кривая. В принципе, для многих практических применений достаточно и такого описания. Пример синтеза управления для траектории данного класса можно найти в работах [7, 8, 47].

Теперь, чтобы выполнить условия поставленной задачи и обеспечить C^2-гладкость, необходимо ввести в рассмотрение ещё один базовый элемент, соединяющий отрезки прямых и дуги окружностей. Как правило, для этого используют разнообразные спирали. Наиболее распространённым подходом является использование клотоиды или спирали Корню [24, 38, 51, 52, 67, 69, 74, 75]. Данная кривая примечательно тем, что её кривизна возрастает линейно, относительно её длины. Недостатком же является то, что использовать её на практике возможно только в параметрическом виде, так как она может быть записана через координаты базиса только в виде интегралов Френеля, которые в общем виде не выражаются в виде элементарных функций. Альтернативным путем решения является использование кривой Ферма [28], но в данном подходе для использования её при планировании траектории нет возможности аналитически найти точки касания спирали с окружностью. Также для этих целей возможно использования сплайнов или кривых Безье [2, 16, 22, 35, 44, 45, 57, 66, 69, 77], но результаты обладают большой размерностью и сложностью. В нашем подходе предлагается использовать в качестве сегмента, соединяющего прямые с окружностью кубическую параболу (см. рис. 7).

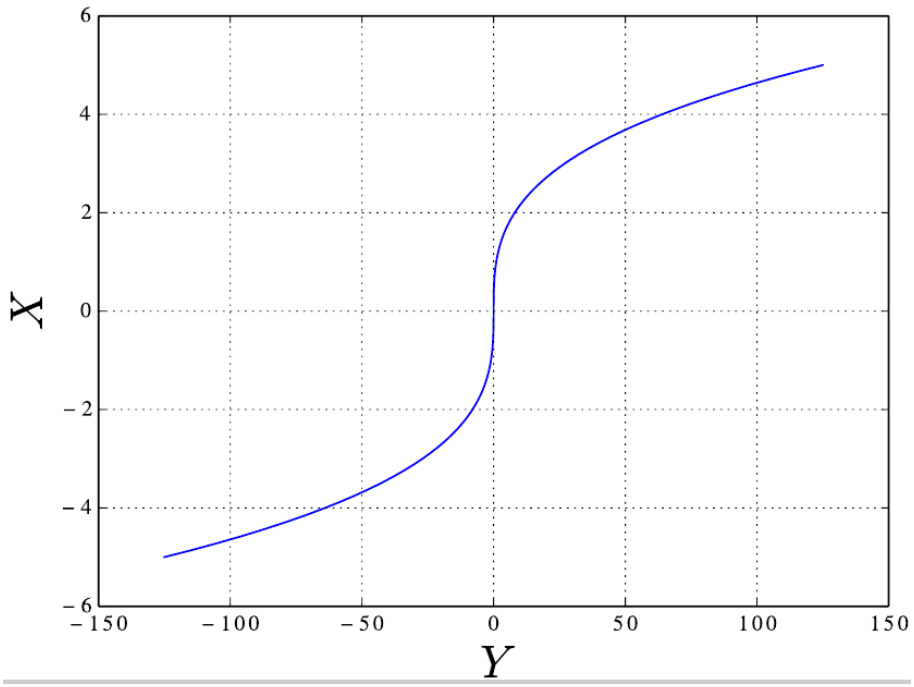

Рис. 7: График кубической параболы, задаваемой уравнение (16) при $k = 1$.

Неявное уравнение кубической параболы имеет вид

$$kx^3 - y = 0, \tag{16}$$

где k – настраиваемый параметр.

Как уже упоминалось выше, для участков перехода выбирают кривую, обладающую рабочим участком с линейно возрастающей кривизной. Для анализа данного свойства у предложенной кривой (16) найдём функцию, характеризующую кривизну. Для этого воспользуемся формулой кривизны для неявной кривой в виде

$$\xi\left(x,y\right)=\left|\frac{\frac{\partial^2\varphi}{\partial x^2}\left(\frac{\partial\varphi}{\partial y}\right)^2+\frac{\partial^2\varphi}{\partial y^2}\left(\frac{\partial\varphi}{\partial x}\right)^2-2\frac{\partial^2\varphi}{\partial x\partial y}\frac{\partial\varphi}{\partial x}\frac{\partial\varphi}{\partial y}}{\left(\sqrt{\frac{\partial\varphi}{\partial x}^2+\frac{\partial\varphi}{\partial y}^2}\right)}\right|. \tag{17}$$

где $\xi\left(x,y\right)$ – кривизна неявной кривой в точке (x,y). В результате получаем уравнение

$$\xi\left(x,y\right)=\left|\frac{6kx}{\left(1+9k^2x^4\right)^{\frac{3}{2}}}\right|. \tag{18}$$

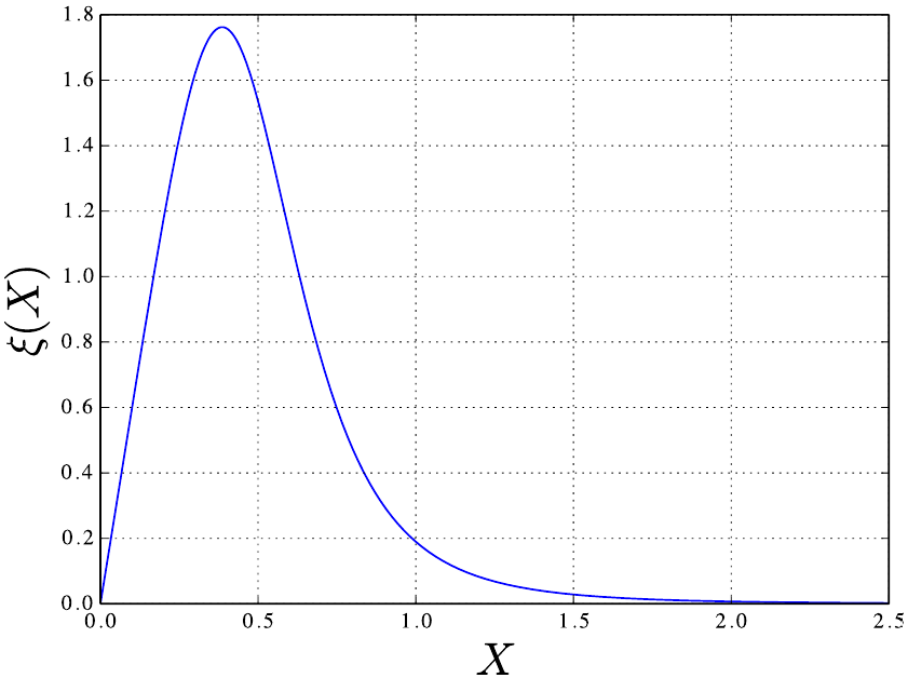

Рис. 8: Зависимость кривизны от координаты (16) при $k=1$.

Как видно из рисунка 8, участок от нуля до максимума практически линейный. Линеаризуем функцию (18), разложив её в ряд Тейлора в окрестности нулевой точки. Получаем выражение

$$\hat{\xi}\left(x\right)=6kx-81k^3x^5+\frac{3645k^5x^9}{4}+O(x^{13}), \tag{19}$$

где $\hat{\xi}(x)$ – функция кривизны, разложенная в ряд Тейлора. Точка максимум функции (18) находится в виде

$$x_{\max} = \frac{1}{\sqrt{3k}\sqrt[4]{5}}, \xi_{\max}(x_{\max}) = \frac{5\sqrt{k}\sqrt[4]{5}}{3\sqrt{2}}. \tag{20}$$

Как можно заметить, исходя из выражений (19) и (20), мы можем, варьируя параметр k, задавать максимум, тем самым обеспечивая выполнения ограничения (1), а также добиваясь приемлемой точности линеаризации функции $\hat{\xi}(x)$. К примеру, для параметра $k = 1$ линейная функция $\hat{\xi} = 6kx$ обеспечивает приемлемую точность где-то на половине диапазона $[0, \xi_{\max}]$ (см. рис. 9).

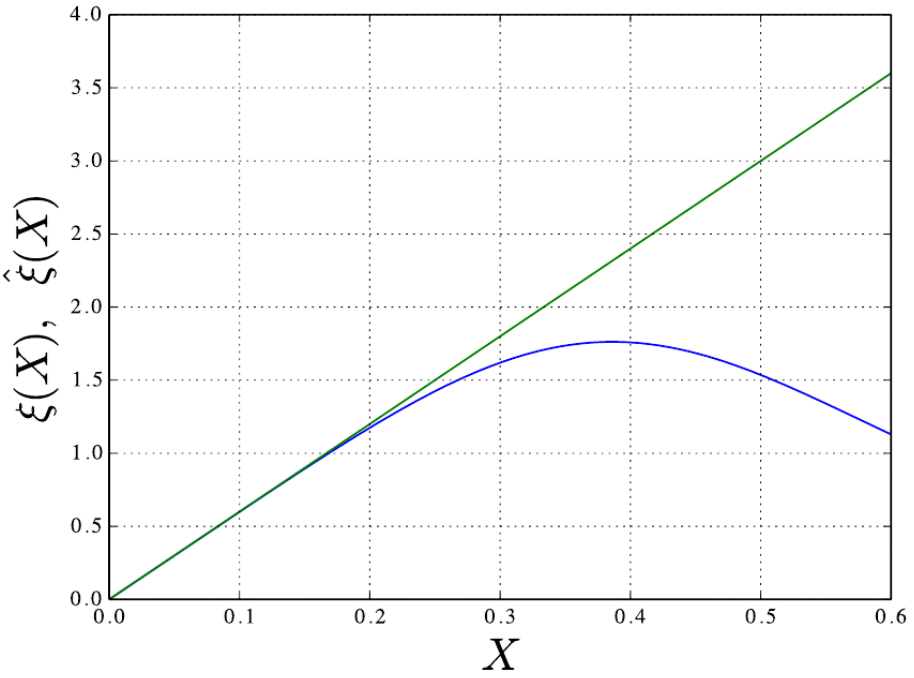

Рис. 9: Линеаризация (16) функцией $\hat{\xi}(x) = 6kx$ при $k = 1$.

Теперь, разобравшись с применимостью предложенной функции, дополним наше описание S (15) криволинейным участком, образуемым кубической параболой (16). Для начала, определим крайние точки параболы исходя из заданного радиуса окружности R. Начальный точки расположены на плоскостях переключения Q_1 и Q_2. Для конечных точек получаем

$$\hat{\xi}(x) = 6kx = \frac{1}{R} \Rightarrow x = \frac{1}{6kR}, y = k\left(\frac{1}{6kR}\right)^3.$$

Необходимо заметить, что в выражение для y необходимо ввести функцию знака, в зависимости от того в какую сторону происходит разворот. Кроме того, вторую параболу, соединяющую окружность со следующим участком пути,

необходимо зеркально отразить относительно оси Y:

$$y_1 = sign\,(\sigma)\,k\left(\frac{1}{6kR}\right)^3,$$

$$y_2 = -sign\,(\sigma)\,k\left(\frac{1}{6kR}\right)^3,$$

где y_1 – конечная точка первой параболы, y_2 – конечная точка второй параболы. Далее делаем параллельные переносы и повороты, чтобы совместить параболы и плоскости Q_1 и Q_2. После этого необходимо рассчитать новые положения окружностей с помощью элементарных геометрических преобразований. И найти плоскости переключения Q_3 и Q_4 для перехода от криволинейного участка к круговому и обратно. Пример работы алгоритма представлен на рисунке 10.

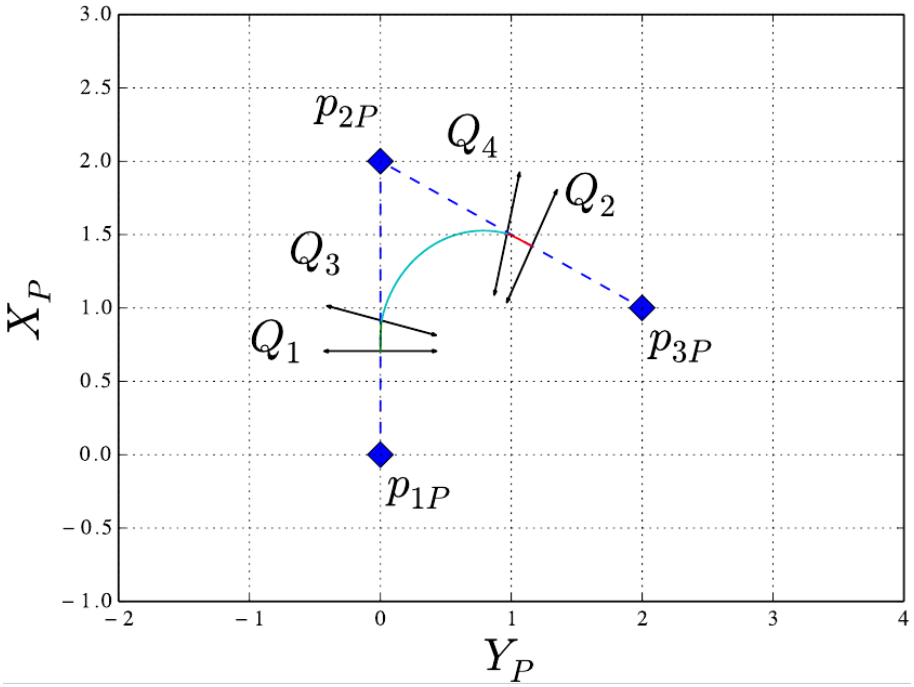

Рис. 10: Пример работы алгоритма с использованием криволинейных участков.

Результирующее описание имеет вид

$$S_1 : \begin{cases} -\sin\psi_i\,(x - x_i) + \cos\psi_i\,(y - y_i) = 0, \\ \psi_i = \arctan 2\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right), \end{cases}$$

$$
S_2 : \begin{cases} k\left(x_L\right)^3 - y_L = 0, \\ \begin{bmatrix} x_L \\ y_L \end{bmatrix} = \begin{bmatrix} x_{i+1} - d_i \\ y_{i+1} \end{bmatrix} + R_I^P\left(\psi_i\right) \begin{bmatrix} x \\ y \end{bmatrix}, \\ d_i = \left| \frac{R}{\tan\frac{\sigma}{2}} \right|, \\ \psi_i = \arctan 2\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right), \end{cases}
$$

$$
S_3 : \begin{cases} \left(x - x_{ci}\right)^2 + \left(y - y_{ci}\right)^2 - R^2 = 0, \\ \begin{bmatrix} x_{ci} \\ y_{ci} \end{bmatrix} = \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} + R_P^I\left(\psi_i\right) R_I^P\left(\delta_i\right) \begin{bmatrix} d_{ci} \\ 0 \end{bmatrix}, \\ d_{ci} = d_{1i} + d_{2i}, \\ d_{1i} = \sqrt{R^2 - \left(h_i/2\right)^2}, \\ d_{i2} = \sqrt{h_{i2}^2 - \left(h_i/2\right)^2}, \\ h_i = \sqrt{\left(x_{S_2e} - x_{S_1e}\right)^2 + \left(y_{S_2e} - y_{S_1e}\right)^2}, \\ h_{i2} = \sqrt{\left(x_{S_2e} - x_{i+1}\right)^2 + \left(y_{S_2e} - y_{i+1}\right)^2}, \\ \delta_i = \pi - \frac{\sigma_i}{2}, \\ \sigma_i = \begin{cases} \pi - \left(\psi_{i+1} - \psi_i\right), \text{ when } \left(\psi_{i+1} - \psi_i\right) > 0 \\ -\pi - \left(\psi_{i+1} - \psi_i\right), \text{ when } \left(\psi_{i+1} - \psi_i\right) \leq 0, \end{cases} \\ \psi_i = \arctan 2\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right), \end{cases}
$$

$$
S_4 : \begin{cases} -k\left(x_L\right)^3 - y_L = 0, \\ \begin{bmatrix} x_L \\ y_L \end{bmatrix} = \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} + R_I^P\left(\psi_{i+1}\right) \begin{bmatrix} x + d_i \\ y \end{bmatrix}, \\ d_i = \left| \frac{R}{\tan\frac{\sigma}{2}} \right|, \\ \psi_i = \arctan 2\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right), \end{cases}
$$

$$
S : \bigcup_{i=1}^{n-1} \begin{cases}
S_1, \text{ when } Q_{1i} \leq 0, \\
S_2, \text{ when } Q_{2i} \leq 0, \\
S_3, \text{ when } Q_{3i} \leq 0, \\
S_4, \text{ when } Q_{4i} \leq 0, \\
i = i + 1, \text{ when } Q_{2i} > 0, \\
Q_{1i} = \cos\psi_i (x - x_i) + \sin\psi_i (y - y_i) - r_i + d_i, \\
Q_{2i} = \cos\psi_{i+1} (x - x_i) + \sin\psi_{i+1} (y - y_i) - d_i, \\
Q_{3i} = \cos\alpha_1 (x - x_i) + \sin\alpha_1 (y - y_i) + t_1, \\
Q_{4i} = \cos\alpha_2 (x - x_i) + \sin\alpha_2 (y - y_i) - t_2, \\
t_1 = \begin{bmatrix} x_{S_1e} & y_{S_1e} \end{bmatrix}^T, \\
t_2 = \begin{bmatrix} x_{S_2e} & y_{S_2e} \end{bmatrix}^T, \\
\alpha_1 = \frac{\psi_{i+1} - \beta}{2}, \\
\alpha_2 = \psi_{i+1} - \frac{\psi_{i+1} - \beta}{2}, \\
\beta = \arccos\left(1 - \frac{h_i^2}{2R^2}\right), \\
h_i = \sqrt{(x_{S_2e} - x_{S_1e})^2 + (y_{S_2e} - y_{S_1e})^2}, \\
\psi_i = \arctan 2 \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right), \\
r_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \\
d_i = \left|\frac{R}{\tan\frac{\sigma}{2}}\right|,
\end{cases}
$$

где x_{S_1e}, x_{S_2e}, y_{S_1e} и y_{S_2e} – конечные точки парабол S_2 и S_4. Результат работы окончательного алгоритма представлен на рисунке 11.

Теперь сделаем небольшое дополнение, чтобы решить задачу планирования траектории для пространственного случая. Неявная траектория в пространстве задается с помощью пересечения двух поверхностей, выраженных в неявном виде. Естественно, в зависимости от выбора типов этих поверхностей можно строить траектории разной сложности. В нашем случае, мы можем ограничить выбор одной их в виде плоскости. Это решение отлично согласуется с представленным выше подходом к решению "задачи о трёх точках так как на этих трёх точках можно построить одну поверхность в виде плоскости и редуцировать исходную подзадачу к рассмотренному выше случаю планирования траектории на плоскости. Для реализации данного подхода необходимо найти матрицу преобразования $R_I^P(\alpha, \beta, \gamma)$, где α, β и γ – углы поворотов относительно осей $X_I Y_I Z_I$ соответственно. Выберем выражения для углов поворота в виде

$$
\beta = \arctan 2 \left(\frac{-z_{i+1} + z_i}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \right),
$$

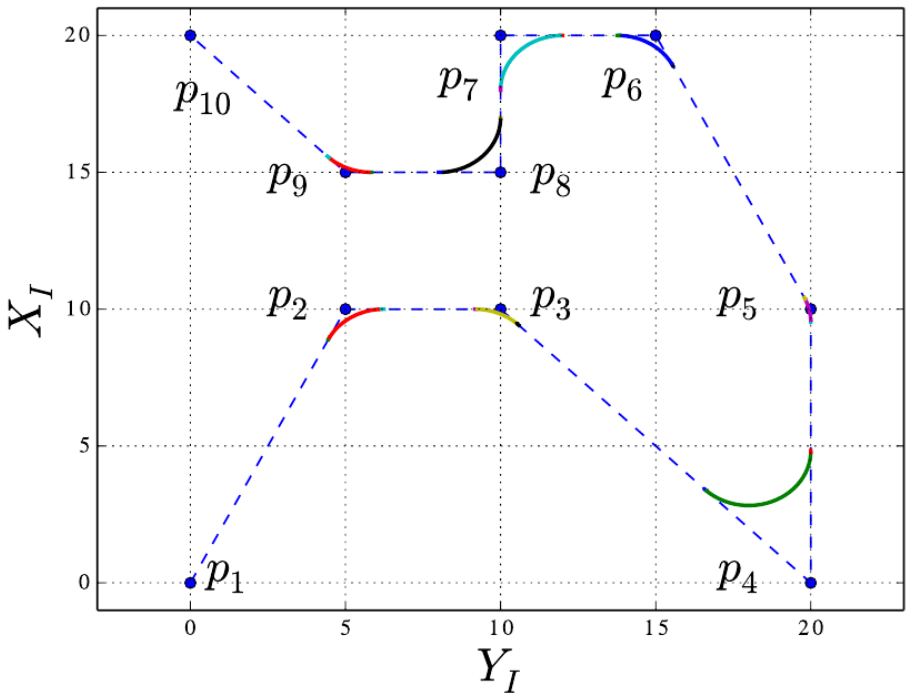

Рис. 11: Результирующая траектория.

$$\gamma = \arctan 2 \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right).$$

Соответствующие матрицы поворота записываются как

$$R_1(\gamma) = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$R_2(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}.$$

Теперь, если мы используем их для перехода в локальную систему координат, то получим систему координат с осью X_P, проходящую через две точки и ориентированную вдоль соединяющих их прямой. В математической записи это будет выглядеть так:

$$p_{Ri} = R_2(\beta) R_1(\gamma) p_i.$$

Теперь надо найти угол α, соответствующий вращению плоскости вокруг оси X_P, который совместит плоскость и третью точку. Его можно найти с помощью выражения

$$\alpha = \arctan 2 \left(\frac{z_{R3} - z_{R2}}{y_{R3} - y_{R2}} \right),$$

где y_{Ri} и z_{Ri} – координаты i-ой точки p_{Ri}. Соответствующая матрица поворота выглядит так:

$$R_3(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}.$$

Таким образом, используя матрицу поворота $R_I^P(\alpha, \beta, \gamma) = R_3(\alpha) R_2(\beta) R_1(\gamma)$ можно перейти из абсолютной системы координат $X_I Y_I Z_I$ в локальную $X_P Y_P Z_P$, для которой выполняется соотношение $Z_P = 0$, задающее нам уравнение плоскости, тем самым, позволяя использовать полученные выше результаты, чтобы найти вторую поверхность. Соответственно, обратный переход задается как

$$p_i = R_P^I(\alpha, \beta, \gamma)\, p_{Ri} = \left(R_I^P(\alpha, \beta, \gamma)\right)^T p_{Ri}.$$

Пример работы алгоритма представлен на рисунке 12.

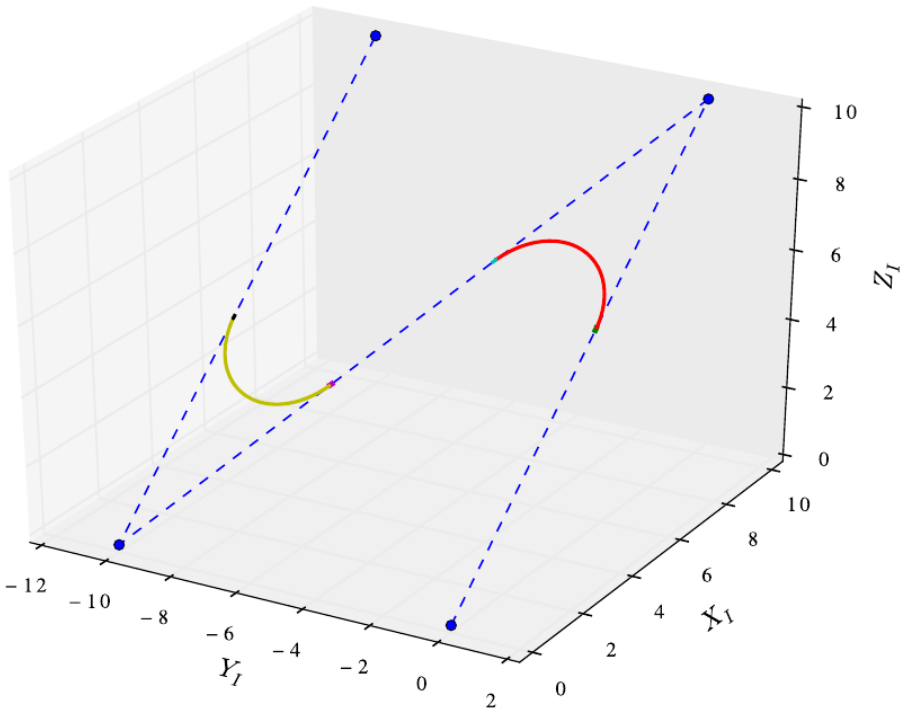

Рис. 12: Результирующая траектория в пространстве.

Рассмотренный подход позволяет и далее улучшать характеристики получаемых результатов, путём добавления новых базовых элементов.

3 Планирование пути с помощью алгоритма A^*

A^* (A-star) – это эвристический метод поиска, разработанный для определения наиболее эффективного маршрута от начальной до конечной вершины графа.

Он объединяет надежные функции поиска пути алгоритма Дейкстры с быстродействующей тактикой жадных алгоритмов. В отличие от подхода Дейкстры, который отдаёт приоритет узлам, ближайшим к началу координат, для расширения, метод A^* использует эвристический компонент, полученный из алгоритма поиска наилучшего первого варианта (Best First Search, BFS). Этот компонент стратегически направляет поисковые усилия к желаемой конечной точке, минимизируя исследование ненужных узлов и сокращая количество бесцельных поисков. Производительность и точность алгоритма во многом зависят от выбора эвристической функции. На практике алгоритм A^* широко используется в различных задачах планирования пути, таких как навигация роботов, сервисы определения местоположения на картах и поиск пути ИИ в играх. Путем соответствующего выбора и настройки эвристической функции алгоритм A^* может эффективно находить кратчайший путь в сложных условиях, сохраняя при этом вычислительную осуществимость и эффективность.

Алгоритм A* использует следующую функцию оценки:

$$f(n) = g(n) + h(n), \tag{21}$$

где функция $g(n)$ представляет собой накопленную стоимость пути от начальной точки до вершины n и имеет вид

$$g(n) = \begin{cases} \operatorname{dist}(S, n) \\ g(n_{pre}) + \operatorname{dist}(n, n_{pre}) \end{cases}, \quad n_{pre} = S, \tag{22}$$

где S – стартовая точка. Если родительский узел узла n, т. е. n_{pre}, соответствует начальному узлу S, то значение $g(n)$ будет определяться расстоянием между узлом n и S. В случае, если родительский узел n не является S, то значение $g(n)$ будет определяться суммированием $g(n_{pre})$ и расстояния между n и n_{pre}.

Эвристическая функция $h(n)$ предсказывает минимальную стоимость пути от вершины n до конечной, полученную с помощью заранее разработанной эвристики, отражающей конкретные условия окружающей среды. В идеале $h(n)$ не должна переоценивать фактическую стоимость, и в этом случае алгоритм A^* гарантирует нахождение пути с минимальной стоимостью. Эвристическую функцию можно задать различными способами, например

- Манхеттенское расстояние

$$h(n) = |n_x - \mathrm{goal}_x| + \left|n_y - \mathrm{goal}_y\right|; \tag{23}$$

- Евклидово расстояние

$$h(n) = \sqrt{(n_x - \mathrm{goal}_x)^2 + \left(n_y - \mathrm{goal}_y\right)^2}. \tag{24}$$

Используя выражение (21), алгоритм A^* оценивает f-значение для каждого узла, где $f(n)$ обозначает совокупную предполагаемую стоимость для каждого узла поиска. Эта оценка, по сути, разветвляется на $g(n)$, как показано в (22), конкретные затраты на переход от начального узла к узлу n, и $h(n)$, которое можно рассчитать с помощью выражения (23) или (24), прогнозируемую минимальную стоимость перехода от узла n к конечному узлу. Алгоритм использует жадную тактику, отдавая приоритет узлу с минимальным f-значением для последующего исследования до тех пор, пока не будет определен пункт назначения. При этом $g(n)$ количественно определяет затраты, понесенные от начала до текущего узла, обычно измеряемые количеством необходимых шагов, в то время как $h(n)$ прогнозирует затраты от текущего местоположения до пункта назначения, часто измеряемые расстоянием до конечной точки.

Для реалиации алгоритма A^* в Matlab используется функция a_star, а для визуализации резултата работы алгоритма – функция a_star_plot. Функция a_star принимает на вход логическую двумерную матрицу, представляющую карту. Значение TRUE указывает на доступную для посещения ячейку карты, а значение FALSE – на невозможность посещения ячейки. Начальная и конечная ячеёки указываются через их номер, нумерация начинается с 1 в левом верхнем углу карты (первый элемент массива) и идет вниз по столбцу. Алгоритм находит кратчайший путь по карте. Для каждой ячейки карты может быть указана стоимость, которая накладывает штраф за посещение этой ячейки. Стоимость может использоваться для обозначения того, что данная ячейка находится дальше других, или её посещение занимает больше времени и т. д.

Листинг 1: Пример реализации алгоритма A^*

```
map = [  ...
     1, 1, 1, 1, 1, 1 ; ...
     1, 0, 0, 0, 1, 1 ; ...
     1, 1, 0, 1, 1, 1 ; ...
     0, 0, 0, 1, 0, 1 ; ...
     1, 1, 1, 1, 1, 1 ; ...
     1, 1, 1, 1, 1, 1 ; ...
     ];
start = 18;
goal = 9;
costs = ones(size(map));
final = a_star(logical(map), costs, start, goal);
a_star_plot(logical(map), costs, final)
```

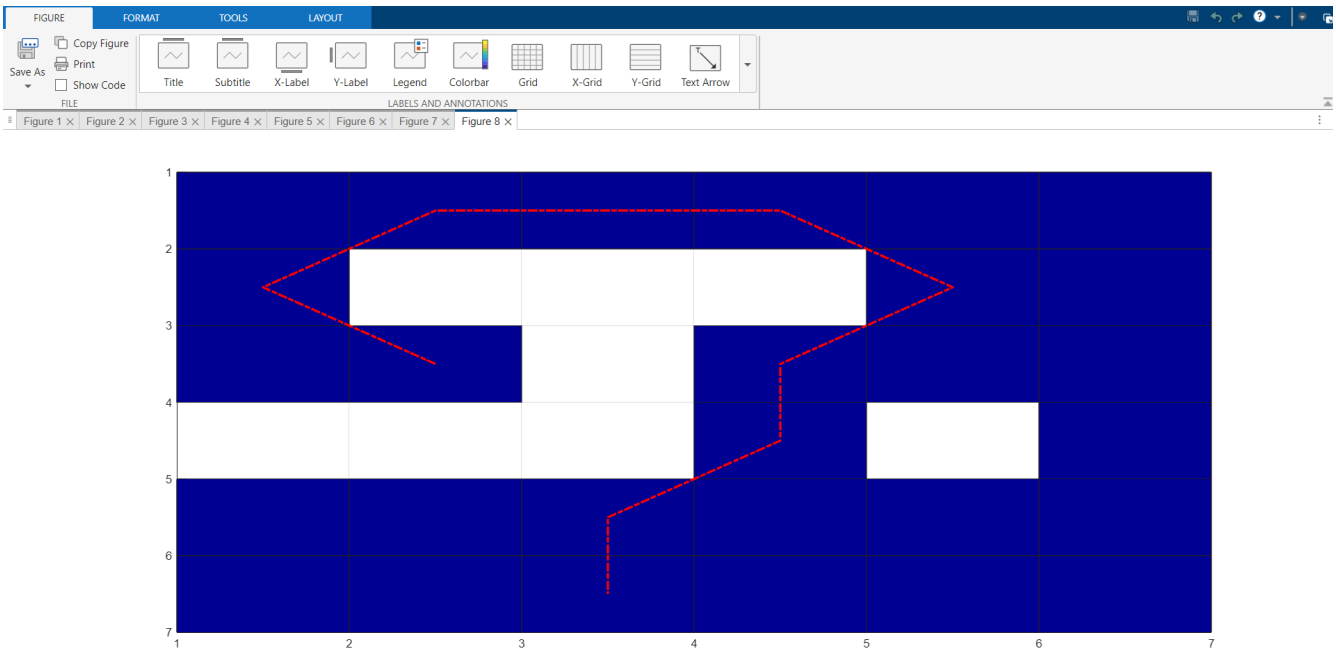

Рис. 13: Пример реализации алгоритма A^8 для Листинга 1.

4 Сглаживание траектории на основе B-сплайнов

Чтобы кривые, генерируемые алгоритмом A^*, лучше соответствовали движению отображаемых объектов, в данной диссертации будет использоваться алгоритм B-сплайна для сглаживания траекторий. Кривые B-сплайна являются производными от кривых Безье и не только эффективно сглаживают кривые, но и решают проблему сложного сшивания кривых.

B-сплайн – это математический инструмент, используемый для построения кривых и графического дизайна, предоставляющий широко используемый метод в компьютерной графике и численном анализе для создания и управления кривыми и поверхностями. B-сплайны определяются серией контрольных точек и степенью (или порядком), и с помощью этих контрольных точек и степени можно строить гладкие кривые или поверхности [20].

- Контрольные точки: группа точек, определяющих форму кривой. B-сплайновые кривые не обязательно проходят через все контрольные точки, в отличие от кривых Безье.

- Степень: степень B-сплайна определяет гладкость кривой. Чем выше степень, тем более гладкой становится кривая. B-сплайновая кривая степени n состоит из полиномов степени $n + 1$, называемых сегментами полиномов.

- Вектор узлов: неубывающая последовательность, используемая для опреде-

ления разбиения пространства параметров кривых или поверхностей, влияющая на форму и сегментацию кривой. Значения в векторе узлов называются узлами.

B-сплайновые кривые строятся путем линейной комбинации набора базисных функций (базисных функций B-сплайна) и контрольных точек. Форма базисных функций зависит от вектора узла, а выбор вектора узла влияет на форму базисных функций и, следовательно, на форму всей кривой.

Определение базисной функции B-сплайна $N_{i,k}(t)$ рекурсивно: Для $k = 1$:

$$N_{i,1}(t) = \begin{cases} 1, & t_i \leq t < t_{i+1} \\ 0 \end{cases}$$

Для $k > 1$:

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i,k+1} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t),$$

где t – параметр, а t_i – i-й узел в векторе узлов.

Первую точку $CP(t)$ на кривой можно рассчитать как взвешенную сумму всех контрольных точек CP_i и соответствующих им нечетных функций $N_{i,k}(t)$ по следующей формуле

$$CP(t) = \sum_{i=0}^{n} CP_i N_{i,k+1}(t). \tag{25}$$

где n – количество контрольных точек минус одна, а k – степень B-сплайна.

Алгоритм построения B-сплайна для n контрольных точек (CP) и степени k, который дает на выходе сглаженную кривую:

1. Задать равномерно распределённый вектор узлов u.

2. Инициализировать пустое множество точек кривой.

3. Рассчитать значение параметра t, равномерно распределённое в диапазоне от 0 до $n - k$.

4. Для значений t, находящихся в диапазоне от начального до конечного параметра кривой, выполнить следующие действия:

   - Вычислить базисную функцию $N_{i,k}(t)$ для всех i по следующим формулам
     - если $k = 0$ то при $u_i \leq t < u_{i+1}$ $N_{i,0}(t) = 1$, в противном случае $N_{i,0}(t) = 0$.

– если $k \neq 0$, то

$$N_{i,k}(t) = \frac{t - u_i}{u_{i+k} - u_i} N_{i,k-1}(t) + \frac{u_{i+k+1} - t}{u_{i+k+1} - u_{i+1}} N_{i+1,k-1}(t).$$

- Вычислить точку кривой

$$C(t) = \sum_{i=0}^{n} N_{i,k}(n) \cdot CP_i.$$

- Добавить $C(t)$ к точкам кривой.

Например, зададим следующие параметры алгоритма генерации B-сплайна:

- В качестве контрольных точек используем набор точек на плоскости с координатами $(0,0)$, $(0,1)$, $(1,1)$, $(1,2)$, $(2,2)$, $(2,1)$, $(3,1)$, $(3,2)$, $(3,3)$, $(4,3)$, $(5,3)$, $(5,2)$, $(4,2)$, $(4,1)$, $(4,0)$ (см. рисунок 14).

- Чтобы сбалансировать гладкость траектории и поиск кратчайшего пути, зададим в качестве параметра $k = 3$.

- Узловые векторы определяют разбиение пространства параметров кривой, влияя на её форму и область влияния контрольных точек. Будем использовать однородные узловые векторы и применим ограничение на обоих концах, чтобы гарантировать прохождение кривой через контрольные точки в начале и конце.

- Для построения точек на кривой необходимо задать количество равномерно распределённых по ней выборок. Зададим его в 10 раз больше числа контрольных точек.

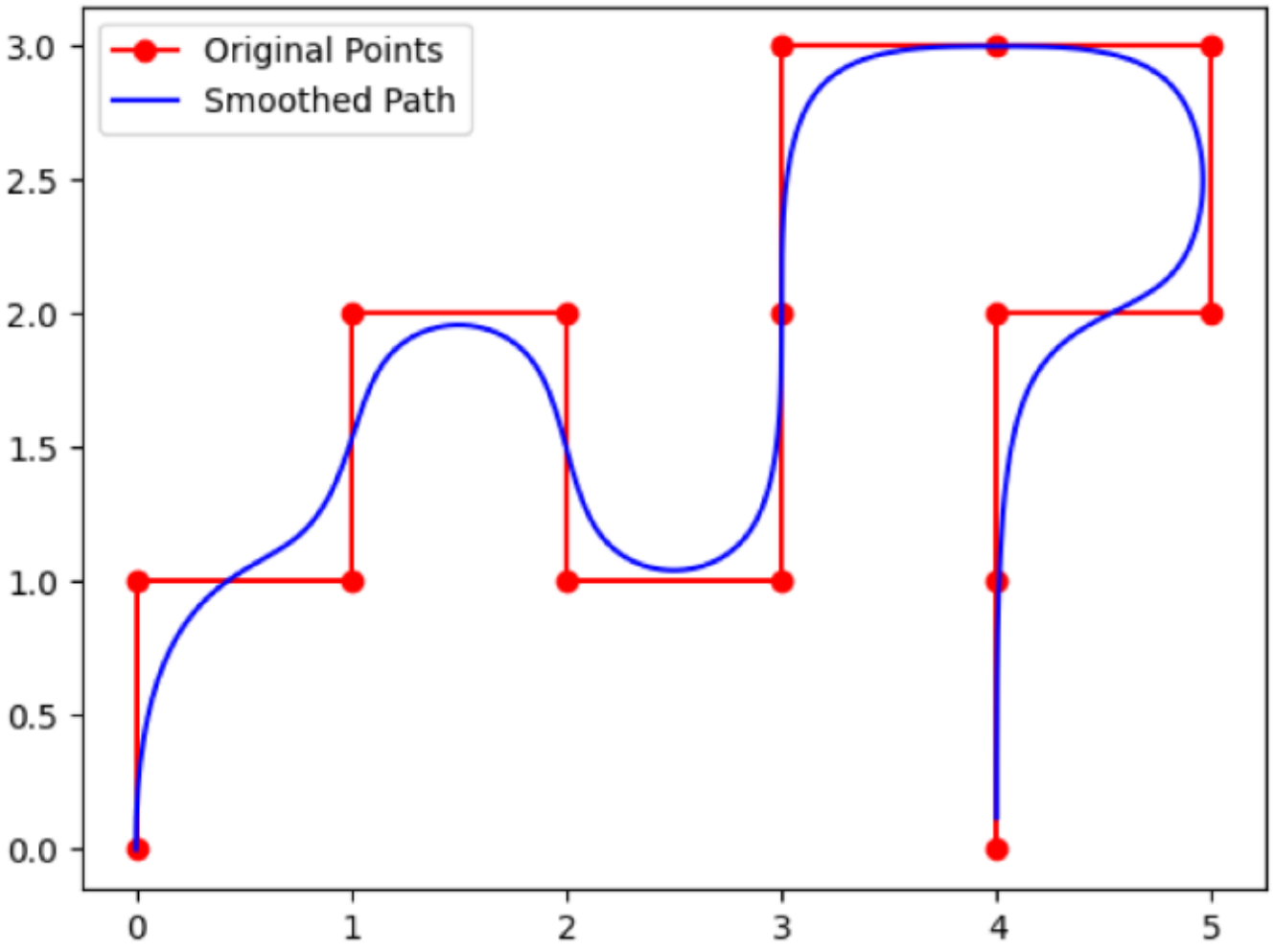

Рис. 14: Пример реализации алгоритма сглаживания траектории при помощи B-сплайна.