ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Квазидискретные дифференциальные игры преследования отличаются своим многообразием. Постановка задач в играх преследования берется из реальных ситуаций. Допустим, на алгоритмы квазидискретных и непрерывных игр преследования таких, как «катер – торпеда» или «ракета – самолет», оказывало влияние техническое состояние вооруженных сил различных стран. Но с течением времени техническое состояние, технологии не стоят на месте. Соответственно в алгоритмы существующих моделей квазидискретных игр преследования требуется внести существенные поправки. Если не создавать модель и алгоритм решения заново.

В настоящее время каждую из задач, относящихся к области дифференциальных игр, можно смоделировать на компьютере в режиме реального времени. Мы в своих исследованиях старались для каждой из задач, представленных в монографии, создать свою компьютерную динамическую модель.

Мы в своих исследованиях старались произвести квазидискретное моделирование классических задач из школ Р. Айзекса, Л.С. Понтрягина, Н.Н. Красовского, А.И. Субботина, J.V. Breakwell, W.H. Fleming, M.G. Crandall, P.L. Lions и др.

Развитие современных информационных технологий позволило вывести классические задачи преследования на уровень компьютерного моделирования. Такие отрасли индустрии, как производство беспилотных летательных аппаратов, конструирование робототехнических комплексов с элементами искусственного интеллекта сделали актуальными задачи разработки моделей поведения для задач преследования.

Класс задач, связанных с задачей преследования, необычайно широк, и входят в отдельный раздел теории управления, как дифференциальные игры.

Но мы в своей исследовательской работе будем больше опираться на методы вычислительной геометрии. Поскольку разрабатываемые

математические модели преследования и уклонения создаются для реальных поверхностей, заданных точечным базисом.

С появлением пакетов компьютерной математики стал шире класс задач преследования, с помощью которых можно не только выполнить математическое моделирование, но и произвести динамическую визуализацию, согласованную с реальным временем.

1. МОДЕЛЬ ПАРАЛЛЕЛЬНОГО СБЛИЖЕНИЯ

Метод параллельного сближения является оптимальным методом преследования на плоскости в том, что он обеспечивает минимальное время достижения цели преследователем. Это показано в работах Петросяна Л.А. Дифференциальные игры преследования. Л.: Изд-во Ленингр. ун-та, 1997, Дифференциальные игры на выживание со многими участниками // Доклады АН СССР. 1965. Т. 161. № 2. С. 285–287.

Метод параллельного сближения используется при наведении летальных аппаратов на цель. В этом случае линия, соединяющая преследователя и цель (линия визирования), при перемещении всегда сохраняет параллельность своему первоначальному положению.

Как и на плоскости, так и в пространстве, направления скоростей преследователя и цели пересекаются в точке на окружности Аполлония.

То есть направление скорости преследователя однозначно определяется расположением преследователя и цели, направлением и величиной скорости цели, а также величиной скорости преследователя.

В рамках настоящей главы нами предложена модификация метода параллельного сближения в случае произвольного направления вектора скорости преследователя.

1.1 МЕТОД ПАРАЛЛЕЛЬНОГО СБЛИЖЕНИЯ НА ПЛОСКОСТИ

В данном разделе представлена квазидискретная геометрическая модель задачи преследования с простым движением на плоскости методом параллельного сближения. Строится окружность Аполлония и связанные с ней характеристические линии для каждого момента времени. В данной геометрической модели для предопределенной траектории цели находится оптимальная траектория преследователя. Моделирование производилось в системе компьютерной математики MathCAD. По результатам моделирования был изготовлен анимационный ролик, где можно просмотреть перемещение и преобразование окружности Аполлония и связанных с ней характеристических точек и линий.

Окружности Аполлония применяются при решении задач простого преследования на плоскости, используя стратегии параллельного сближения. Простым движением в задачах преследования точки называется такое

движение, при котором пройденное расстояние S является линейной функцией времени: $S(t) = v \cdot t$, где $v = Const$ – модуль скорости точки.

Окружность Аполлония – это геометрическое место точек плоскости, когда $|KP|/|KT| = Const$ (Рисунок 1.1).

Применительно к задачам преследования окружность Аполлония содержит следующую идею. Если преследователь и цель в определенный момент времени имеют положения на плоскости P и T, и значения скоростей, равные по модулю V_P и V_T, соответственно.

Тогда геометрическое множество точек K, как место возможных встреч преследователя P с целью T, является окружностью радиуса $|QK|$ с центром в точке Q [1], [2].

Направления скоростей преследователя и цели являются взаимосвязанными. То есть направление скорости цели диктует направление скорости преследователя или наоборот, направление преследователя определяет направление цели, чтобы обеспечить встречу в точках, принадлежащих окружности Аполлония.

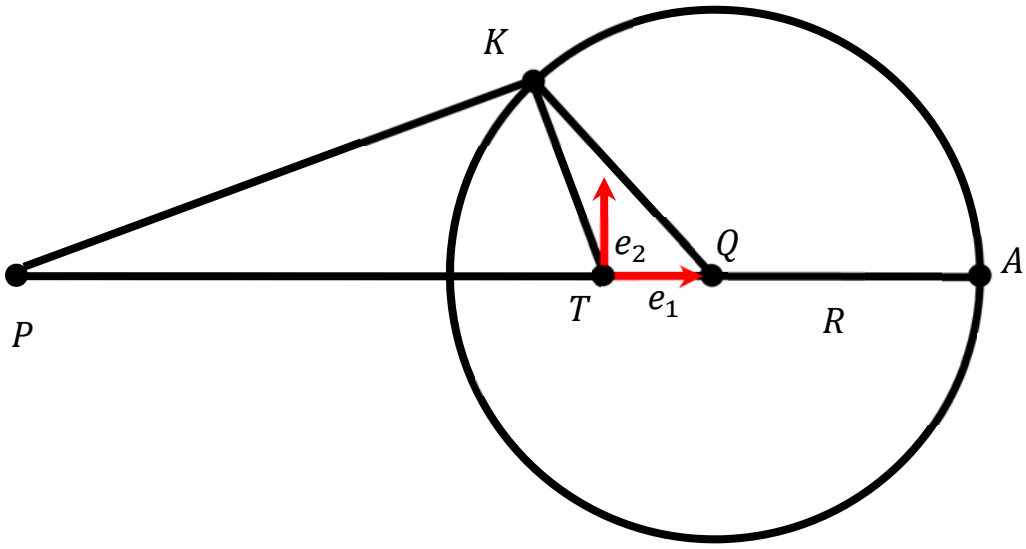


Рисунок 1.1. Окружность Аполлония

Целью данной статьи является формализация и алгоритмизация метода параллельного сближения преследователя и цели.

То, что данное множество точек K является окружностью, было известно еще древним математикам, но мы приведем выкладки расчета окружности и ее центра.

Введем ортонормированную систему координат (e_1, e_2) с центром в точке T (Рисунок 1), вектор e_1 сонаправлен вектору PT. Пусть $K = \begin{bmatrix} x \\ y \end{bmatrix}$, а $P = \begin{bmatrix} -a \\ 0 \end{bmatrix}$, где $a = |PT|$. Тогда $|TK| = \sqrt{x^2 + y^2}$, а $|PK| = |K - P| = \sqrt{(x+a)^2 + y^2}$. Из условия того, что преследователь и цель приходят в точку K одновременно, имеем следующее: $\frac{|PK|}{V_P} = \frac{|TK|}{V_T}$. Откуда следует, что $V_T \cdot \sqrt{(x+a)^2 + y^2} = V_P \cdot \sqrt{x^2 + y^2}$. После возведения в квадрат и раскрытия скобок получаем следующее уравнение:

$$\left(x - \frac{V_T^2}{V_P^2 - V_T^2} \cdot a\right)^2 + y^2 = \left(\frac{V_P \cdot V_T}{V_P^2 - V_T^2} \cdot a\right)^2.$$

Полученное уравнение в системе (e_1, e_2) с центром в точке T описывает окружность радиуса R и с центром в точке Q:

$$Q = \begin{bmatrix} \frac{V_T^2}{V_P^2 - V_T^2} \cdot a \\ 0 \end{bmatrix}, R = \frac{V_P \cdot V_T}{V_P^2 - V_T^2} \cdot a, a = |PT|.$$

Отметим одну характеристическую точку, называемой точкой Аполлония:

$$A = \begin{bmatrix} \frac{V_T^2}{V_P^2 - V_T^2} \cdot a + \frac{V_P \cdot V_T}{V_P^2 - V_T^2} \cdot a \\ 0 \end{bmatrix}.$$

Факт того, что стратегия преследователя в задаче преследования при помощи метода параллельного сближения, является оптимальной в плане минимизации времени поимки цели доказан в работах Петросяна Л.О. [1-8].

Будем считать, что для нашего итерационного процесса известны начальные данные P_0 и T_0. Скорости преследователя и цели постоянны и равны по модулю V_P и V_T, соответственно.

Траектория цели в нашей модели является предопределенной, поэтому мы сможем рассчитать массив точек $\{T_i\}$, где дистанция между точками T_i и T_{i+1} равна:

$$|T_{i+1} - T_i| = V_T \cdot \Delta T, \qquad \Delta T - \text{период дискретизации по времени.}$$

Итерационная схема расчета координат преследователя, координат центров окружностей Аполлония, радиусов окружностей Аполлония, характеристических точек представлена на рисунке 1.2.

Координаты преследователя на i - ом шаге итераций будут выглядеть следующим образом:

$$P_i = P_{i-1} + V_P \cdot \Delta T \cdot \frac{K_{i-1} - P_{i-1}}{|K_{i-1} - P_{i-1}|}.$$

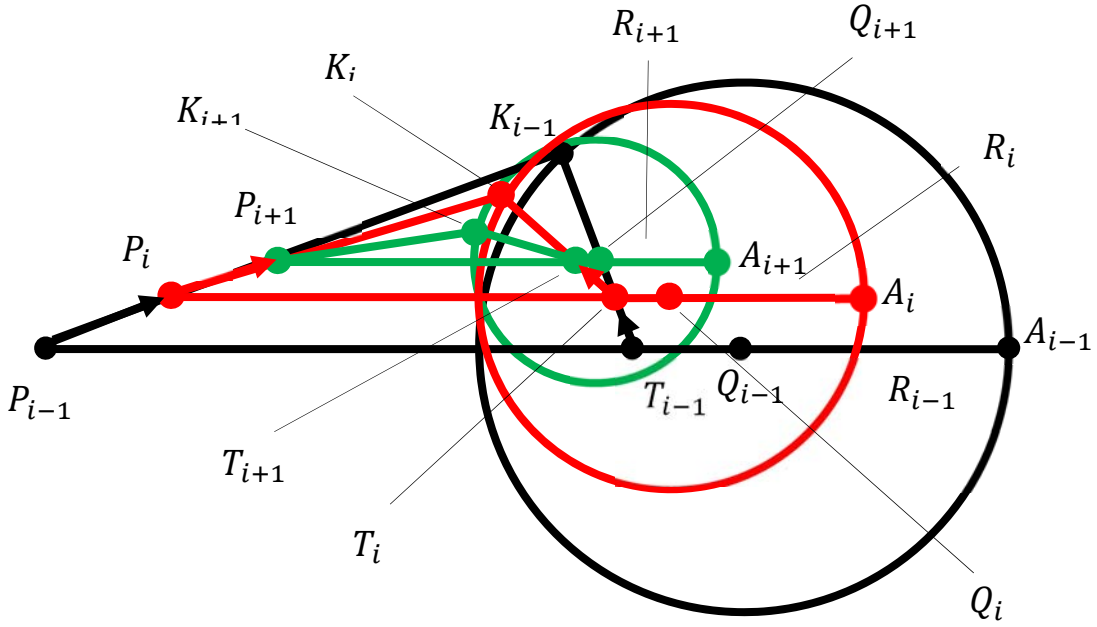


Рисунок 1.2. Итерационная схема

Радиус окружности Аполлония:

$$R_i = \frac{V_P \cdot V_T}{V_P^2 - V_T^2} \cdot |T_i - P_i|.$$

Центр окружности Аполлония рассчитывается таким образом:

$$Q_i = T_i + \frac{V_T^2}{V_P^2 - V_T^2} \cdot (T_i - P_i).$$

Координаты точки K_i есть продукт решения системы уравнений относительно непрерывного параметра t:

$$\begin{cases} (K_i - Q_i)^2 = R_i^2 \\ K_i = T_i + V_T \cdot \dfrac{T_{i+1} - T_i}{|T_{i+1} - T_i|} \cdot t \end{cases}.$$

Разрешенная относительно параметра t, вышеуказанная система представляет собой корни квадратного уравнения, вывод которых в данной статье мы приводить не будем из-за громоздких выражений.

В тестовой программе, написанной по материалам статьи, решение квадратного уравнения написано в виде отдельной процедуры – функции. С текстом тестовой программы можно ознакомиться здесь [18].

То, что отрезок $[P_i, T_i]$ останется параллельным отрезку $[P_0, T_0]$, не вызывает сомнений. Рассмотрим первый отрезок $[P_1, T_1]$.

Координаты точек P_1 и T_1 равны (Рисунок 1.3):

$$P_1 = P_0 + V_P \cdot \frac{P_0K_0}{|P_0K_0|} \cdot \Delta T$$

$$T_1 = T_0 + V_T \cdot \frac{T_0K_0}{|T_0K_0|} \cdot \Delta T$$

Исходя из того, что преследователь и цель должны придти в точку K на окружности Аполлония одновременно, мы вправе сделать вывод, что:

$$\frac{V_P}{|P_0K_0|} \cdot \Delta T = \frac{V_T}{|T_0K_0|} \cdot \Delta T = \varepsilon.$$

Далее:

$$P_1T_1 = T_1 - P_1 = (T_0 - P_0) + \varepsilon \cdot T_0K_0 - \varepsilon \cdot P_0K_0 = (1 - \varepsilon) \cdot (T_0 - P_0).$$

Другими словами, вектор P_1T_1 сонаправлен вектору P_0T_0 и перпендикулярен вектору нормали N (Рисунок 1.3).

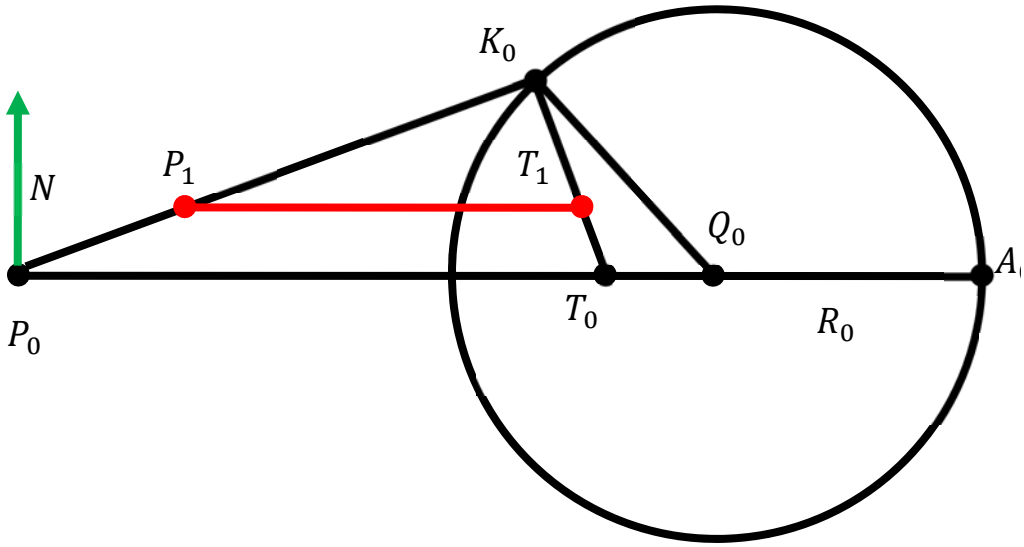


Рисунок 1.3. Параллельное сближение преследователя и цели

Нами была разработана тестовая программа, скриншот результатов работы которой, показан на рисунке 1.4. На рисунке 1.4 видно, что отрезки $[P_i, T_i]$ образуют однопараметрическую последовательность параллельных $[P_0, T_0]$ линий.

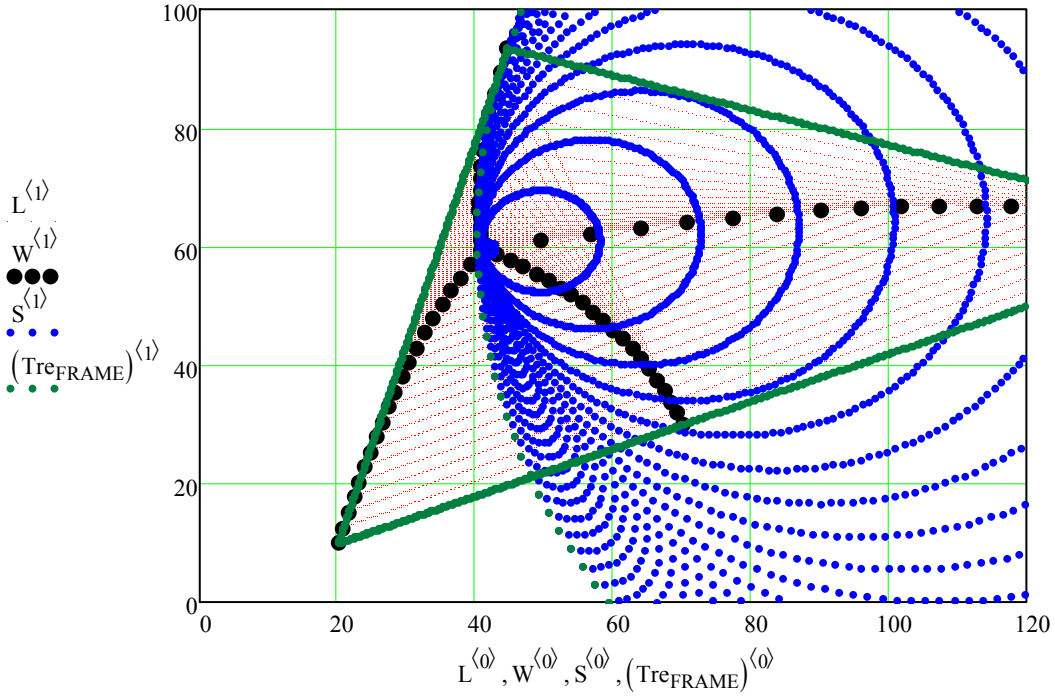


Рисунок 1.4. Перехват цели

Также на рисунке 1.4 показано, точки P_i, T_i, Q_i, A_i принадлежат одной прямой. Показан подвижный треугольник P_i, Q_i, K_i, который сходится к точке встречи преследователя и цели.

Рассматривая рисунок 1.4, а именно на однопараметрическое множество окружностей Аполлония, сходящееся к точке встречи, может возникнуть обманчивое впечатление, все множество окружностей все касается в точке встречи преследователя и цели. В следующей модели с иной траекторией цели мы покажем, что это не так.

Рисунок 1.4 дополнен ссылкой на анимированное изображение [43], где можно посмотреть как изменяется во времени расположение преследователя, цели, точек на окружности Аполлония.

Итак, ситуацию, представленную на рисунке 1.4, можно интерпретировать как моделирование перехвата преследователем цели.

Ситуацию, представленную на рисунке 1.5, можно интерпретировать как моделирование процесса убегания цели от преследователя [44].

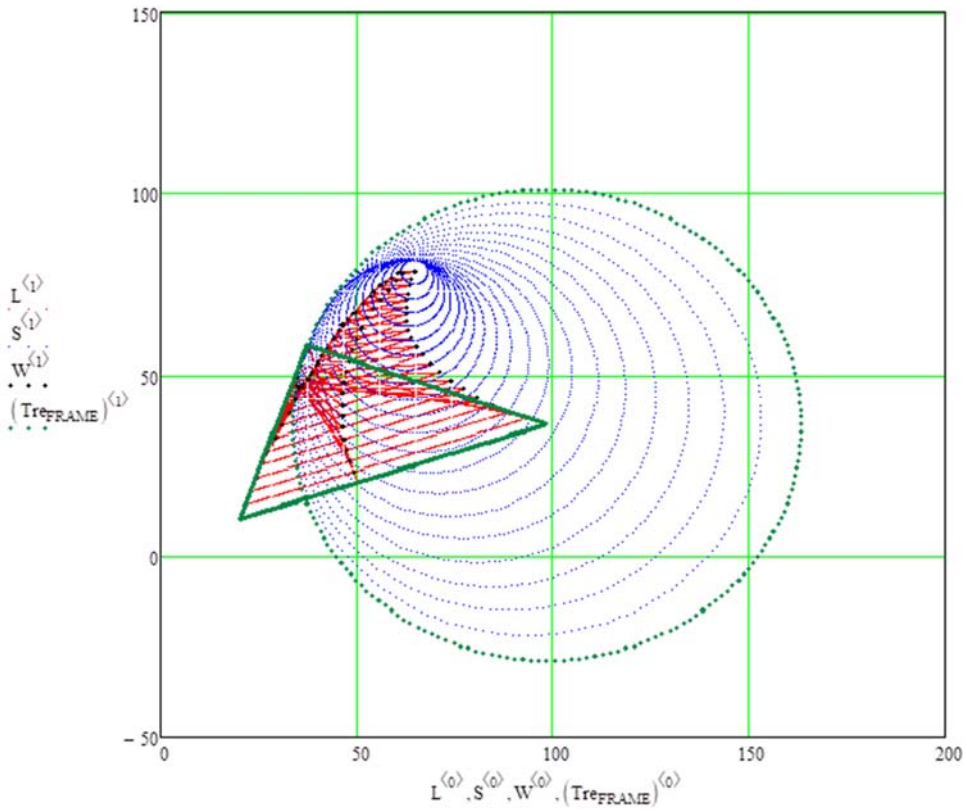


Рисунок 1.5. Моделирование убегания цели от преследователя

На рисунке 1.5 также показан подвижный треугольник P_i, Q_i, K_i, точки P_i, T_i, Q_i, A_i и окружности Аполлония. Обратим внимание на поведение точки K_i, оно похоже на поведение точки возврата второго рода. Здесь как раз наблюдается схождение окружностей Аполлония к точке встречи преследователя и цели, но множество окружностей не является касательными в одной точке [44].

Рисунок 1.5 также дополнен ссылкой на анимированное изображение.

Целью данной статьи являлось показать движение всех характеристических линий и точек в реализации метода параллельного сближения в задачах простого преследования на плоскости. Показано, движение окружности Аполлония, ее схождение к точке встречи преследователя и цели. Продемонстрировано, как сходится отрезок, соединяющий преследователя и цель, будучи параллелен самому себе.

При геометрическом моделировании методом параллельного сближения не обязательно вычислять параметры окружности Аполлония. Достаточно

будет строить однопараметрическое множество параллельных линий, соединяющих преследователя и цель, окружность, с радиусом равным шагу преследователя, чтобы найти точку следующего положения преследователя.

Тексты программ выложены на ресурсе автора [18]. Ссылки на анимированное изображение, изготовленных по результатам работы программ доступны на ресурсах [22], [23]. При написании статьи приняты во внимание результаты, полученные в следующих источниках [9-17], [19-21].

1.2 КИНЕМАТИЧЕСКАЯ МОДЕЛЬ МЕТОДА ПАРАЛЛЕЛЬНОГО СБЛИЖЕНИЯ НА ПЛОСКОСТИ

В данном разделе рассматривается модель задачи преследования методом параллельного сближения. Целью данной статьи является модификация метода параллельного сближения для того, чтобы учитывать случай, когда в момент начала преследования вектор скорости преследователя направлен не на цель. Кроме того, в рассматриваемой в статье модели, преследователь не может мгновенно изменять направление движения. То есть происходит наложение условия, что радиус кривизны траектории движения преследователя не может быть меньше определенной величины. Предлагаемый метод основан на том, преследователь, выбирая шаг на этапе итераций, будет стараться следовать прогнозируемым траекториям. По материалам статьи написана тестовая программа, которая рассчитывает траекторию преследователя, учитывая изложенные условия. Выполненное анимированное изображение визуализирует изменение координат преследователя, цели и прогнозируемых траекторий от времени.

В описании задачи преследования методом параллельного сближения в трудах Петросяна Л. О. [1],[2], [3] направление вектора скорости преследователя P и направление вектора скорости цели T пересекаются в одной точке K, принадлежащей окружности Аполлония (Рисунок 1.1).

Для точек P и T точка K окружности Аполлония характерна тем, отношение длин $\frac{|PK|}{|QK|} = \frac{|V_P|}{|V_T|}$ есть отношение модулей скоростей преследователя и цели.

При квазидискретном моделировании точек траектории преследователя $\{P_i\}$ можно предложить следующую итерационную схему (Рисунок 1.2):

$$P_i = P_{i-1} + V_P \cdot \Delta T \cdot \frac{K_{i-1} - P_{i-1}}{|K_{i-1} - P_{i-1}|}.$$

Радиус окружностей Аполлония будут такими:

$$R_i = \frac{V_T^2}{V_P^2 - V_T^2} \cdot |T_i - P_i|.$$

Центры окружностей Аполлония рассчитываются так:

$$Q_i = T_i + \frac{V_T^2}{V_P^2 - V_T^2} \cdot (T_i - P_i).$$

Координаты точки K_i есть продукт решения системы уравнений относительно непрерывного параметра t:

$$\begin{cases} (K_i - Q_i)^2 = R_i^2 \\ K_i = T_i + V_T \cdot \dfrac{T_{i+1} - T_i}{|T_{i+1} - T_i|} \cdot t \end{cases}.$$

Такова одна из квазидискретных моделей построения траектории преследователя. Она требует, чтобы направления векторов движения преследователя и цели пересекались в точках, принадлежащим окружностям Аполлония.

Если рассмотреть итерационную схему, представленную на рисунке 2, то мы видим что на каждом шаге итераций линии, соединяющие преследователя и цель $(P_i T_i)$, всегда параллельны между собой. В источниках [2], [3] приводится доказательство этого факта. Но мы приведем его здесь.

Рассмотрим отрезок $[P_i, T_i]$. Координаты точек P_1 и T_1 равны (Рисунок 1.2):

$$P_i = P_{i-1} + V_P \cdot \frac{P_{i-1}K_{i-1}}{|P_{i-1}K_{i-1}|} \cdot \Delta T$$
$$T_i = T_{i-1} + V_T \cdot \frac{T_{i-1}K_{i-1}}{|T_{i-1}K_{i-1}|} \cdot \Delta T$$

Исходя из того, что преследователь и цель должны придти в точку K_{i-1}на окружности Аполлония одновременно, мы вправе сделать вывод, что:

$$\frac{V_P}{|P_{i-1}K_{i-1}|} \cdot \Delta T = \frac{V_T}{|T_{i-1}K_{i-1}|} \cdot \Delta T = \varepsilon.$$

Далее:

$$P_i T_i = T_i - P_i = (T_{i-1} - P_{i-1}) + \varepsilon \cdot T_{i-1}K_{i-1} - \varepsilon \cdot P_{i-1}K_{i-1}$$
$$= (1 - \varepsilon) \cdot (T_{i-1} - P_{i-1}).$$

Другими словами, вектор P_iT_i сонаправлен вектору $P_{i-1}T_{i-1}$ (Рисунок 1.2).

Данная модель нахождения траектории преследователя не позволяет моделировать, если точка пересечения направлений векторов движения не принадлежит окружности Аполлония для заданной пары точек преследователя и цели. Целью данной статьи является разработка метода решения именно такой задачи.

Итерационную схему, представленную на рисунке 1.2, можно интерпретировать иначе.

На рисунке 1.6 представлен итерационный процесс, определяющий координаты точки P_i при известных координатах точек P_{i-1}, T_{i-1}, T_i и скоростей преследователя и цели, V_P и V_T, соответственно.

Другими словами, линия визирования, соединяющая преследователя и цель, в процессе перемещения остается параллельной своему первоначальному положению.

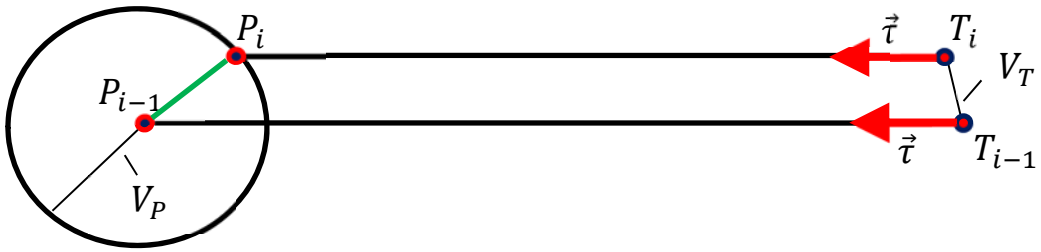


Рисунок 1.6. Интерпретация итерационной схемы

Сначала определяется единичный вектор:

$$\vec{\tau} = \frac{T_{i-1} - P_{i-1}}{|T_{i-1} - P_{i-1}|}.$$

Координаты точки $T_i = T_{i-1} + \vec{V}_T \cdot \Delta T, \Delta T - \text{период дискретизации}$, предопределены поведением цели. Тогда прямую линию, которая будет соединять точки P_i и T_i, можно представить в виде $L(\mu) = T_i + \mu \cdot \vec{\tau}$. Тогда координаты P_i следующего шага итераций траектории преследователя можно

интерпретировать как точку окружности радиуса $V_P \cdot \Delta T$ с центром в точке P_{i-1} и прямой линии $L(\mu)$:

$$\begin{cases} (L(\mu) - P_{i-1})^2 = (V_P \cdot \Delta T)^2 \\ L(\mu) = T_i + \mu \cdot \vec{\tau} \end{cases}.$$

Решение вышеприведенной системы уравнений относительно параметра μ даст такое значение параметра, при котором будет выполняться следующее: $P_i = T_i + \mu \cdot \vec{\tau}$.

Такая интерпретация итерационной схемы параллельного сближения позволяет перейти к расчету траектории преследователя, когда в момент начала преследования скорость преследователя направлена не на точку на окружности Аполлония.

Другими словами, не так как показано на рисунке 1.1.

Итерационную схему параллельного сближения мы предлагаем модифицировать следующим образом. Пусть в момент начала сближения вектор скорости преследователя P_{i-1} направлен произвольным образом, но не в точку на соответствующей паре точек $\{P_{i-1}, T_{i-1}\}$ окружности Аполлония (Рисунок 1.7).

В силу инертности преследователя, минимальный радиус кривизны траектории не может быть меньше определенного значения r_c. Точке преследователя P_{i-1} соответствуют вектор скорости $\vec{V}_{P\,i-1}$ и вектор единичной нормали $\vec{n}_{i-1}$, $\vec{V}_{P\,i-1} \cdot \vec{n}_{i-1} = 0$.

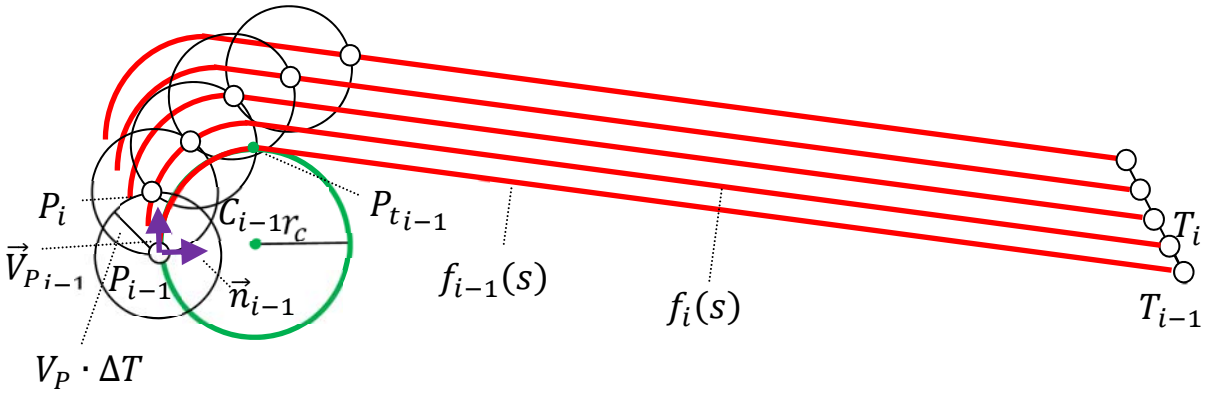


Рисунок 1.7. Квазидискретная модель параллельного сближения

Далее, находится центр окружности радиуса r_c: $C_{i-1} = P_{i-1} + V_P \cdot \Delta T \cdot \vec{n}_{i-1}$. К построенной окружности строится касательная из точки T_{i-1} для нахождения точки $P_{t_{i-1}}$ сопряжения прямой и окружности (Рисунок 1.7).

Дугу окружности $\overset{\frown}{P_{i-1}P_{t_{i-1}}}$ и отрезок $\left[P_{t_{i-1}}T_{i-1}\right]$ будем считать одной составной кривой линией $f_{i-1}(s)$, где параметром s служит длина дуги нашей параметрической кривой. Выбор в качестве параметра длины дуги вполне обоснован потому, что сегментами составной кривой могут служить не только отрезок прямой и дуги окружности, но и, к примеру, кривые Безье или кубические параболы.

Следует отметить, что в нашей тестовой программе, написанной по материалам статьи, отсчет длины дуги начинается от точки T_{i-1}, $f_{i-1}(0) = T_{i-1}$.

Производим параллельный перенос линии $f_{i-1}(s)$ на вектор $T_i - T_{i-1}$. Положение точки T_i известно и полностью определяется поведением цели. В рамках решения нашей задачи будем считать поведение цели полностью детерминированным.

Уравнение линии параллельной линии $f_i(s) = f_{i-1}(s) + T_i - T_{i-1}$ будем считать известным и для нахождения точки P_i следующего шага преследователя, необходимо решение следующей системы уравнений и неравенств относительно параметра s:

$$\begin{cases} (f_i(s) - P_{i-1})^2 = (V_P \cdot \Delta T)^2 \\ f_i(s) = f_{i-1}(s) + T_i - T_{i-1} \\ 0 \le s < s_{i-1} \end{cases}.$$

Где s_{i-1} - это значение параметра s, соответствующее точке P_{i-1}.

При реализации кинематической модели задачи преследования методом параллельного сближения был выбран пакет компьютерной математики MathCAD 15. Отметим некоторые особенности нашей тестовой программы.

Первое, что мы сделали в нашей программе, мы для составной кривой в момент начала преследования, состоящей из дуги и отрезка, выполнили параметризацию от длины дуги. Для этого нам необходимо было получить упорядоченный набор точек $\{x_i, y_i\}$. По каждой координате встроенными средствами MathCAD, выполнили кубическую сплайн-интерполяцию от формального параметра δ и получили функции $X(\delta), Y(\delta), i \in [0, N-1], \delta_i \leq \delta \leq \delta_{i+1}$, где $\delta_i = i$, а $N -$ количество элементов массивов $\{x_i, y_i\}$.

Далее, был составлен Якобиан для передачи во встроенные решатели обыкновенных дифференциальных уравнений системы MathCAD:

$$D(s, \delta) = \frac{1}{\sqrt{\frac{dX}{d\delta}^2 + \frac{dY}{d\delta}^2}}.$$

Полученное решение методом Рунге-Кутты четвертого порядка выражает зависимость массивов $\{x_i, y_i\}$ от параметра длины дуги s. Таким образом, мы считаем, что уравнение базовой кривой, с которой мы будем совершать параллельный перенос, получено (Рисунок 1.8).

Далее, нам предстоит создать вычислительный цикл, в котором решалась система уравнений и неравенств. Данная задача сводится к численному решению уравнения поиском нулей функции методом секущей в заданном диапазоне.

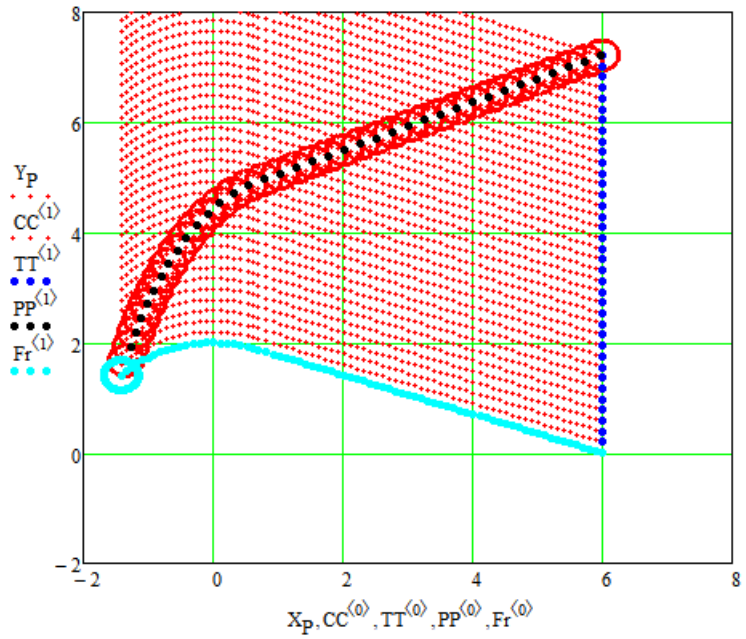


Рисунок 1.8. Кинематическая модель параллельного сближения

Встроенные средства численного решения уравнений системы MathCAD позволяют это решить при помощи процедуры *root*. Процедурой *root* решается в вычислительном цикле уравнение:

$(f_{i-1}(s) + T_i - T_{i-1} - P_{i-1})^2 - (V_P \cdot \Delta T)^2 = 0$, в диапазоне $s \in [0, s_{i-1}]$.

На рисунке 1.8 представлены результаты моделирования тестовой программы. Рисунок 1.8 дополнен ссылкой на анимированное изображение [45], где в динамике можно будет посмотреть процесс методом параллельного сближения.

В нашей тестовой программе мы намеренно выбрали траекторию движения цели в виде прямой линии. В результатах этих экспериментов мы выяснили следующее. Если в начале движения, скорость преследователя направлена на точку на окружности Аполлония, то время достижения цели будет всегда меньше времени, если бы преследование выполнялось бы методом погони с теми же параметрами. В нашем случае, описываемый нами метод, не является оптимальным, но перспективность в плане группового преследования с разными скоростями из разных точек, но с одновременным достижением цели, несомненна.

В данном разделе рассматривается кинематическая модель задачи преследования на плоскости методом погони, когда в момент начала преследования скорость преследователя направлена не на цель.

Данный метод возможен для использования при разработке геометрической модели группового преследования с одновременным достижением цели или целей. Также возможно использование при разработке моделей, когда преследователь методом параллельного сближения достигает цели под заданными углами.

Данный метод моделирования задач преследования методом параллельного сближения может быть востребован при проектировании БПЛА с автономным управлением.

По предложенным моделям и алгоритмам написаны тестовые программы расчета траекторий в системе компьютерной математики MathCAD. Тексты программ доступны на ресурсе автора [147]. Ссылки на анимированное изображение, изготовленных по результатам работы программ доступны на ресурсе [45].

При написании главы за основу приняты теоретические результаты, полученные в следующих источниках [87, 98, 99, 115]. Также приняты во внимание результаты работ [81, 111, 113,114, 135, 136].

1.3 КИНЕМАТИЧЕСКАЯ МОДЕЛЬ МЕТОДА ПАРАЛЛЕЛЬНОГО СБЛИЖЕНИЯ НА ПОВЕРХНОСТИ

В данном разделе рассматривается кинематическая модель задачи преследования, перенесенная с плоскости на поверхность, методом параллельного сближения. Моделирование итерационных процессов в задачах преследования является актуальным с развитием автономных беспилотных объектов. Целью статьи является разработка модели, в которой траектория преследователя есть результат следования прогнозируемым маршрутам в каждый дискретный момент времени. Исследования проводились в системе компьютерной математики MathCAD на поверхности, заданной точечным

базисом. Моделировались ситуации с различными исходными состояниями. С подробными результатами в виде анимированных изображений по материалам статьи, программных кодов можно ознакомиться на сайте и канале автора.

Ранее, в работах Петросяна Л. О. [91-98] рассматривался метод параллельного сближения на плоскости, в которых было показано, что данный метод являет оптимальным для достижения преследователем маневрирующей цели.

Задачу преследования на плоскости методом параллельного сближения можно интерпретировать так, как показано на рисунке 1.6:

$$\begin{cases} \vec{\tau} = \dfrac{Target_{i-1} - Pers_{i-1}}{|Target_{i-1} - Pers_{i-1}|} \\ L(\mu) = Target_i + \mu \cdot \vec{\tau} \\ (L(\mu) - Pers_{i-1})^2 = (V_P \cdot \Delta T)^2 \end{cases}.$$

Если цель совершает шаг: $Target_i = Target_{i-1} + \vec{V}_T \cdot \Delta T$, то из точки $Target_i$ проводится линия $L(u)$ и ищется ее точка пересечения с окружностью радиуса $V_P \cdot \Delta T$ с центром в точке $Pers_{i-1}$.

В этой статье предлагается модель подобной итерационной схемы, но уже в пространстве на поверхности.

В данной статье рассматривается итерационная модель последовательного расчета точек траектории преследователя методом, который в проекции на горизонтальную плоскость аналогичен методу параллельного сближения.

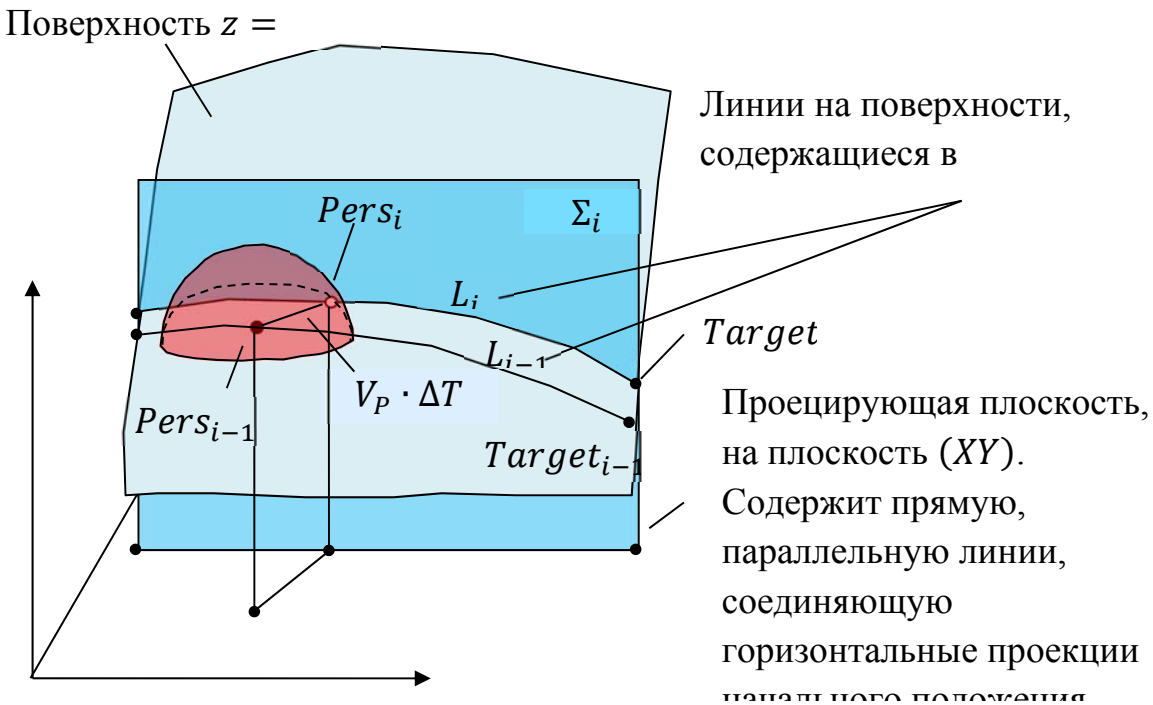


Рисунок 1.9. Расчет следующего шага итерационного процесса

Для расчета следующего шага итераций в точке нахождения преследователя $Pers_{i-1}$ на поверхности $z = f(x, y)$ строится сфера S_{i-1} радиуса $V_P \cdot \Delta T$ (Рисунок 1.9). Затем, ищется точка $Pers_i$ пересечения сферы S_{i-1} с линией L_i, Которая и является искомой точкой следующего шага. В точке нахождения $Target_i$ цели строится проецирующая плоскость Σ_i, которая является параллельной линии, соединяющей горизонтальные проекции начальных положений точек $Pers_0$ и $Target_0$, преследователя и цели. Линия L_i есть продукт пересечения поверхности $z = f(x, y)$ и плоскости Σ_i.

Будем считать, что поверхность передвижения преследователя и цели, задана в явном виде $z = f(x, y)$. Вообще, предлагаемая к рассмотрению геометрическая модель, будет работать, когда преследователь и цель будут оказывать взаимное влияние на поведение друг друга. В написанной по материалам статьи программе траектория цели предопределена и задается проекцией на плоскость (XY) в виде функций $x_t(t)$ и $y_t(t)$. Где t - формальный параметр. После обработки проекции получаем уравнение траектории цели:

$$Target(s) = \begin{bmatrix} x_t(t(s)) \\ y_t(t(s)) \\ f\left(x_t(t(s)), y_t(t(s))\right) \end{bmatrix}, \text{где } s - \text{параметр длины дуги.}$$

Полный дифференциал длины дуги будет таким: $ds^2 = {dx_t}^2 + {dy_t}^2 + \left(\frac{\partial f}{\partial x_t} \cdot dx_t + \frac{\partial f}{\partial y_t} \cdot dy_t\right)^2$. Откуда мы приходим к дифференциальному уравнению:

$$\frac{dt}{ds} = \frac{1}{\sqrt{\frac{{dx_t}^2}{dt} + \frac{{dy_t}^2}{dt} + \left(\frac{\partial f}{\partial x_t} \cdot \frac{dx_t}{dt} + \frac{\partial f}{\partial y_t} \cdot \frac{dy_t}{dt}\right)^2}}.$$

Данное уравнение в тестовой программе мы решаем методом Рунге-Кутты четвертого порядка с начальными условиями $t(0) = 0$. В результате решения мы получили функциональную зависимость $t = t(s)$. Если скорость движения цели по поверхности постоянна и равна по модулю V_T, то можно перейти к параметру времени: $s = V_T \cdot T$.

Шаг цели в нашей итерационной модели будет таким:

$$Target_i = Target_{i-1} + \frac{dTarget(s)}{ds} \cdot V_T \cdot \Delta T.$$

Ход решения данного итерационного процесса определятся начальным положением точек преследователя и цели. Рассмотрим начальное положение точек преследователя и цели, $Pers_0$ и $Target_0$ (Рисунок 1.10). Их проекции на плоскость (XY) будут $Pers_{XY\,0}$ и $Target_{XY\,0}$.

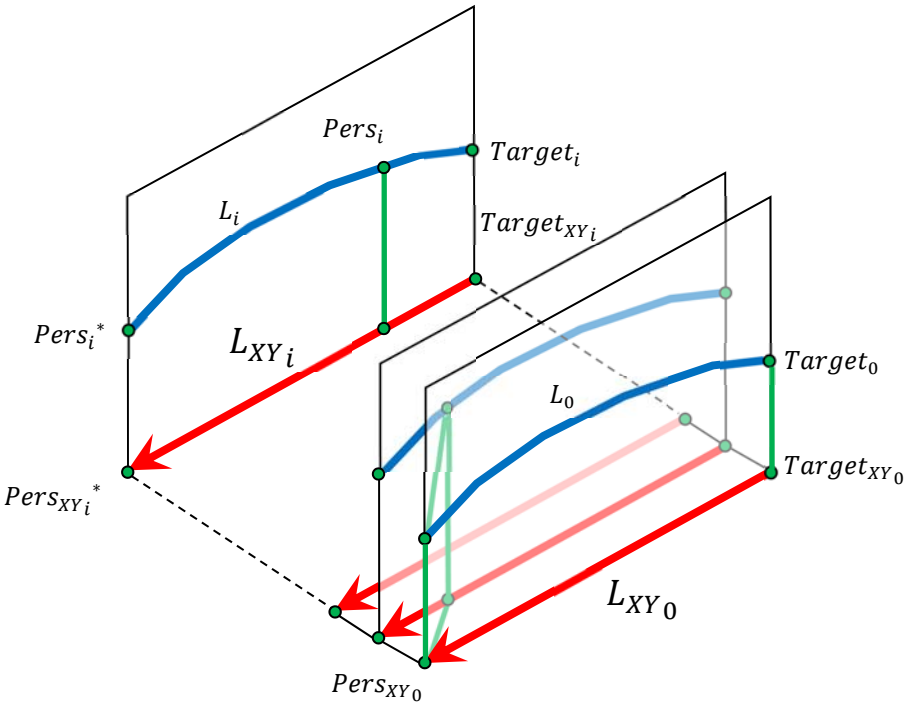


Рисунок 1.10. Множество проецирующих плоскостей

На плоскости (XY) сформируем вектор $\tau = Pers_{XY_0} - Target_{XY_0}$. Где $Pers_{XY_0}$ и $Target_{XY_0}$ горизонтальные проекции точек $Pers_0$ и $Target_0$ начального положения. Если координаты точки $Target_i$ известны, то известна и ее горизонтальная проекция $Target_{XY_i}$. От каждой проекции $Target_{XY_i}$ отложим вектор τ для получения точки ${Pers_{XY_i}}^*$. Далее, на плоскости (XY) формируется параметрическая прямая $L_{XY_i}(\gamma) = (1-\gamma) \cdot Target_{XY_i} + \gamma \cdot {Pers_{XY_i}}^*$. Прямую $L_{XY_i}(\gamma)$ на плоскости (XY) можно разложить по координатам:

$$L_{XY_i}(\gamma) = \begin{bmatrix} L_{X_i}(\gamma) \\ L_{Y_i}(\gamma) \end{bmatrix}.$$

Откуда мы получим параметрическое уравнение линии

$$L_i(\gamma) = \begin{bmatrix} L_{X_i}(\gamma) \\ L_{Y_i}(\gamma) \\ f\left(L_{X_i}(\gamma), L_{Y_i}(\gamma)\right) \end{bmatrix}$$

на поверхности $z = f(x, y)$.

Чтобы получить координаты следующего шага итерации $Pers_i$, необходимо в ранее вычисленной точке $Pers_{i-1}$ построить сферу с радиусом $V_P \cdot \Delta T$. Где V_P модуль скорости, ΔT - период дискретизации по времени.

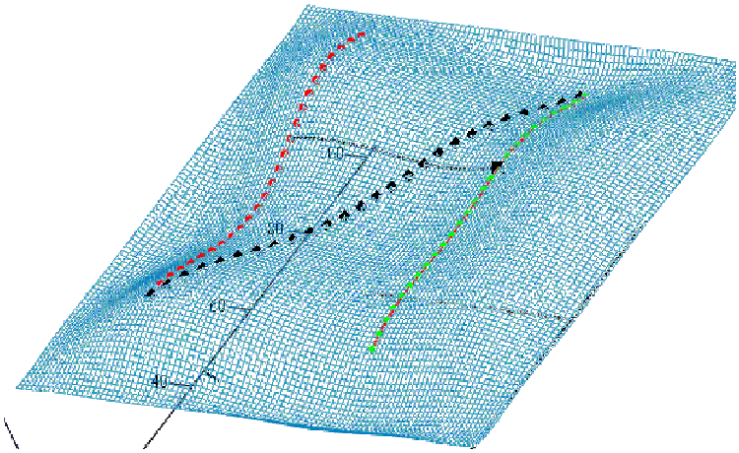


Рисунок 1.11. Проекция прямой линии на поверхность

Для решения уравнения $|L_i(\gamma) - Pers_{i-1}| = V_P \cdot \Delta T$ относительно параметра γ используются встроенные средства систем компьютерной математики. В нашей тестовой программе для решения использовался метод Мюллера. Найденное значение γ подставляем в уравнение линии $L_i(\gamma)$, $Pers_i = L_i(\gamma)$. Это значение и будет искомой точкой следующего шага нашего итерационного процесса.

По материалам главы была написана тестовая программа, которая последовательно рассчитывает точки траектории движения преследователя (Рисунок 1.11). Полный текст программы с комментариями расположен на ресурсе [4].

На рисунке 1.11 показаны множество точек траектории цели $\{Target_i\}$, множество точек $\{Pers_i^*\}$ и одна из линий, соединяющих точки $\{Target_i\}$ и $\{Pers_i^*\}$.

Рисунок 1.11 дополнен ссылкой на анимированное изображение [46].

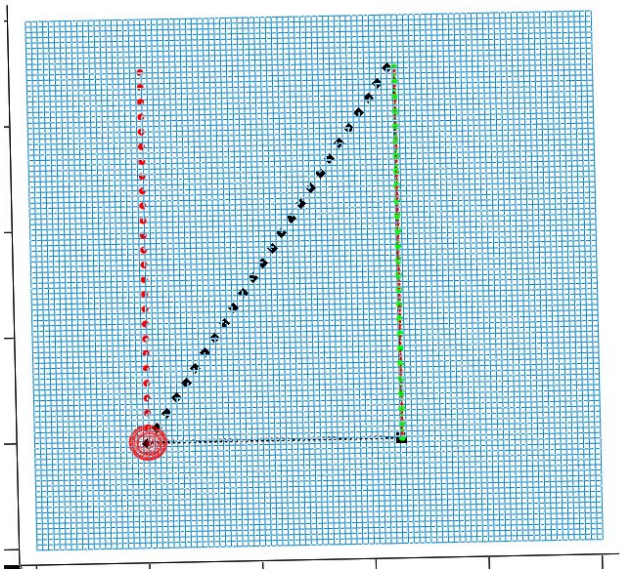


Рисунок 1.12. Сфера на поверхности

Каждая из точек траектории преследователя есть продукт пересечения сферы радиуса $V_P \cdot \Delta T$ с центром в месте предыдущего положения преследователя. На рисунке 1.12 изображена одна сфера. Из рисунка видно, сфера проходит через точку следующего шага траектории. Рисунок 1.12 дополнен ссылкой на анимированное изображение [47].

Метод параллельного сближения в задачах преследования на плоскости и пространстве имеет множество практических реализаций. Хотелось бы реализовать аналог метода преследования на поверхностях. Отметим, что цель и преследователь могут находиться на разных поверхностях. Поверхность преследователя может являться эквидистантой к поверхности цели или близкой к этому. При таком преследовании встают вопросы о прямой видимости цели, об ее возможности скрыться в «складках местности». Нашей целью являлась разработка модели, которая бы пригодилась разработчикам автономных робототехнических комплексов с элементами искусственного интеллекта.

1.4 КОММЕНТАРИИ К ПРОГРАММЕ ВИЗУАЛИЗАЦИИ ОКРУЖНОСТЕЙ АПОЛЛОНИЯ В ЗАДАЧЕ ПРЕСЛЕДОВАНИЯ НА ПЛОСКОСТИ

В данном разделе мы подробно разберем листинг тестовой программы, которая была написана, чтобы произвести визуализацию всей динамической картины при выполнении преследования одной цели методом параллельного сближения.

Преследование происходит на плоскости размерами 100×100 метров.

Каждая линия, которая будет визуализирована в данной задачи, будет построена при помощи данного количества точек

$N := 200$

Число расчетных тактов в задаче преследования

$m := 20$

Начальная позиция преследователя в момент начала преследования

$P := \begin{pmatrix} 20 \\ 10 \end{pmatrix}$

Начальная позиция цели в момент начала преследования

$T := \begin{pmatrix} 70 \\ 30 \end{pmatrix}$

Модуль скорости преследователя ($\frac{m}{c}$)

$V_p := 14$

Модуль скорости цели ($\frac{m}{c}$)

$V_T := 11$

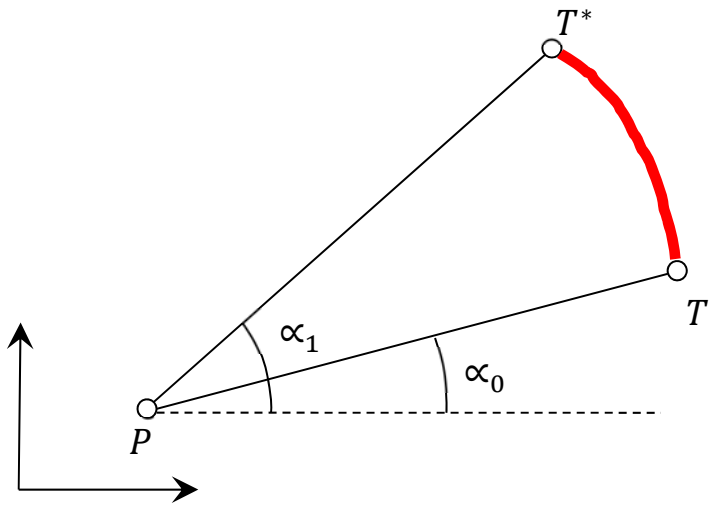


Рисунок 1.13. Предопределенная траектория цели

Цель T будет двигаться по предопределенной траектории, а именно по дуге окружности радиуса $|T - P|$ с центром в точке P. Начинается движение с позиции с углом α_0 и до позиции с углом $\propto_1$ (Рисунок 1.13).

Начальный угол траектории цели

$$\alpha_0 := \operatorname{acos}\left[\frac{(T - P)_0}{|T - P|}\right] = 0.381$$

Конечный угол траектории цели

$$\alpha_1 := \alpha_0 + 0.75 \cdot \frac{\pi}{3}$$

Создается ранжированная переменная i от 0 до m.

Ранжированная переменная

$i := 0..m$

Рассчитывается массив углов траектории цели в зависимости от ранжированной переменной i.

$$\alpha_i := \left(1 - \frac{i}{m}\right) \cdot \alpha_0 + \frac{i}{m} \cdot \alpha_1$$

Массив углов траектории цели

Рассчитываются массив точек предопределенной траектории цели в зависимости от ранжированной переменной i.

Массив точек траектории цели

$$x_{T_i} := |T - P| \cdot \cos(\alpha_i) + P_0 \qquad y_{T_i} := |T - P| \cdot \sin(\alpha_i) + P_1$$

Массив радиус-векторов точек цели T представляем в виде двухстолбцовой матрицы.

$$T_{arr} := \text{augment}(x_T, y_T)$$

1-ый столбец значения абсцисс, 2-ой столбец – значения ординат

Далее, составляется функция, в которой входными данными служат: радиус-вектор положения преследователя P, радиус-вектор положения цели T, вектор скорости преследователя V_P, вектор скорости цели V_T и число точек N, из которых будет состоять визуализируемая окружность Аполлония.

$$\begin{aligned}
&\text{Apollo}(p, t, v_p, v_t, n) := \\
&\quad q \leftarrow \begin{pmatrix} t_0 \\ t_1 \end{pmatrix} + \frac{v_t^2}{v_p^2 - v_t^2} \cdot \left[\begin{pmatrix} t_0 \\ t_1 \end{pmatrix} - \begin{pmatrix} p_0 \\ p_1 \end{pmatrix}\right] \\
&\quad r \leftarrow \frac{v_t \cdot v_p}{v_p^2 - v_t^2} \cdot |t - p| \\
&\quad \text{for } i \in 0..n \\
&\quad\quad x_i \leftarrow \left(1 - \frac{i}{n}\right) \cdot p_0 + \frac{i}{n} \cdot t_0 \\
&\quad\quad y_i \leftarrow \left(1 - \frac{i}{n}\right) \cdot p_1 + \frac{i}{n} \cdot t_1 \\
&\quad\quad l \leftarrow \text{augment}(x, y) \\
&\quad\quad xc_i \leftarrow r \cdot \cos\left(\frac{i}{n} \cdot 2\pi\right) + q_0 \\
&\quad\quad yc_i \leftarrow r \cdot \sin\left(\frac{i}{n} \cdot 2\pi\right) + q_1 \\
&\quad\quad lc \leftarrow \text{augment}(xc, yc) \\
&\quad sit \leftarrow \text{stack}(1, q^T, lc) \\
&\quad (sit \;\; q \;\; r)
\end{aligned}$$

Выходными данными являются координаты центра Q, радиус r окружности Аполлония и массив из N точек, принадлежащих окружности Аполлония.

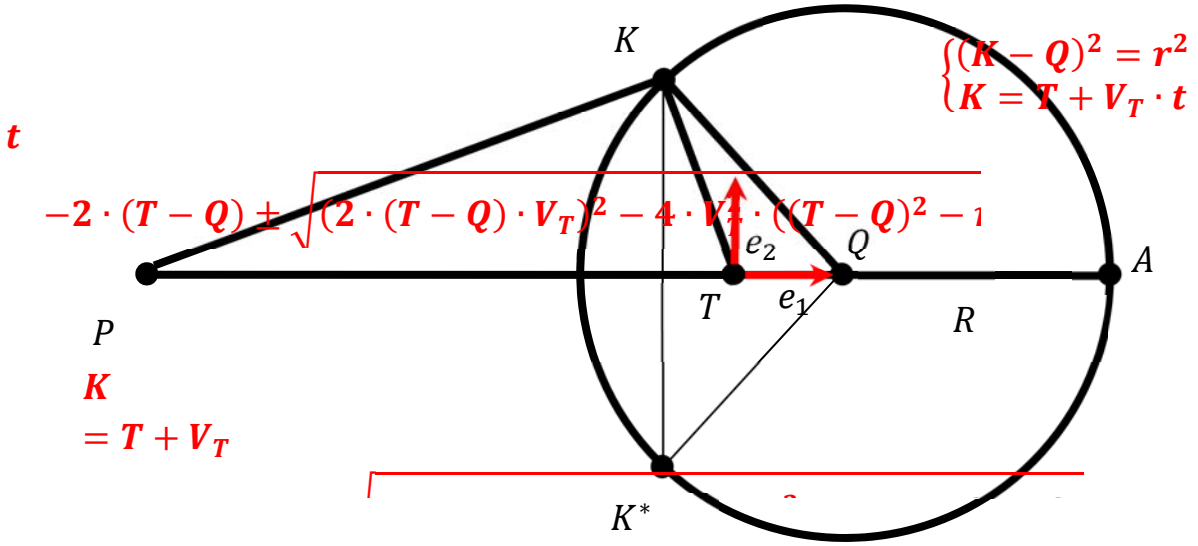


Рисунок 1.14. Вычисление точки *K*

Особенностью этой функции является то, что начало координат помещено в точку положения цели T (Рисунок 1.14). Соответственно, координаты точки Q центра и точек окружности Аполлония выдаются нам относительно системы координат (e_1, T, e_2).

$$
\begin{aligned}
&\text{Point}_{\text{Apollo}}(t,q,v,r) := \\
&\quad A \leftarrow \begin{pmatrix} v_0 & v_1 \end{pmatrix} \cdot \begin{pmatrix} v_0 & v_1 \end{pmatrix}^T \\
&\quad B \leftarrow 2 \cdot \begin{pmatrix} v_0 & v_1 \end{pmatrix} \cdot \left[\begin{pmatrix} t_0 \\ t_1 \end{pmatrix} - \begin{pmatrix} q_0 \\ q_1 \end{pmatrix} \right] \\
&\quad C \leftarrow (t-q)^T \cdot (t-q) - r^2 \\
&\quad D \leftarrow B^2 - 4 \cdot A \cdot C \\
&\quad \tau_1 \leftarrow \frac{-B+\sqrt{D}}{2 \cdot A} \\
&\quad \tau_2 \leftarrow \frac{-B-\sqrt{D}}{2 \cdot A} \\
&\quad k_1 \leftarrow t + v \cdot \tau_1 \\
&\quad k_2 \leftarrow t + v \cdot \tau_2 \\
&\quad \text{augment}(k_1, k_2)
\end{aligned}
$$

Далее, составляется функция, результатом работы которой будут координаты точки K (Рисунок 1.14). На рисунке 1.14 показано, что положений точки K - два. Входными данными нашей функции являются координаты цели T, координаты центра окружности Аполлония Q, вектор скорости цели V_T и

радиус окружности Аполлония R (Рисунок 1.14). На рисунке 1.14 приведены формулы расчета точек K.

В процедуре расчета точки K, как сама точка K, так и координаты входных параметров T, Q представлены в локальной системе координат (e_1, T, e_2) (Рисунок 1.14).

Две процедуры-функции, рассчитывающие параметры окружностей Аполлония, как видно, из нашей тестовой программы, являются зависимыми друг от друга, но в теле наших функций не используются глобальные переменные. Поэтому, теоретически представляется возможным объединить в одну процедуру и вынести в отдельную DLL – библиотеку.

Далее определяем фиксированный промежуток времени или период дискретизации ΔT.

$$\Delta T := \frac{|T - P| \cdot (\alpha_1 - \alpha_0)}{m \cdot V_T}$$

Далее, составляется процедура – функция, где входными данными являются: радиус-вектор начального положения преследователя, массив радиус-векторов точек предопределенной траектории цели T, вектор скорости цели, число точек визуализации каждой линии N, число точек m предопределенной траектории цели и период дискретизации ΔT.

Выходными параметрами служат упорядоченный массив радиус-векторов точек положения преследователя P_i, упорядоченный массив радиус-векторов точек положения цели P_i (избыточная информация, но для определенности мы сделали это), упорядоченный массив радиус-векторов точек положения центров окружностей Аполлония Q_i, упорядоченный массив радиус-векторов точек на окружности Аполлония K_i, упорядоченный массив радиусов окружностей Аполлония R_i. Для пояснения можно посмотреть рисунки 1.1, 1.2, 1.3, 1.14.

$$\text{Triangle}(p, t, v_p, v_t, nn, mm, \Delta t) := \left| \begin{array}{l} tt \leftarrow t^T \\ pp^{\langle 0 \rangle} \leftarrow p \\ qq^{\langle 0 \rangle} \leftarrow \begin{bmatrix} (tt^{\langle 0 \rangle})_0 \\ (tt^{\langle 0 \rangle})_1 \end{bmatrix} + \frac{v_t^2}{v_p^2 - v_t^2} \cdot \left[\begin{bmatrix} (tt^{\langle 0 \rangle})_0 \\ (tt^{\langle 0 \rangle})_1 \end{bmatrix} - \begin{bmatrix} (pp^{\langle 0 \rangle})_0 \\ (pp^{\langle 0 \rangle})_1 \end{bmatrix} \right] \\ rr_0 \leftarrow \frac{v_t \cdot v_p}{v_p^2 - v_t^2} \cdot \left| tt^{\langle 0 \rangle} - pp^{\langle 0 \rangle} \right| \\ kk^{\langle 0 \rangle} \leftarrow \text{Point}_{\text{Apollo}} \left[tt^{\langle 0 \rangle}, qq^{\langle 0 \rangle}, \begin{bmatrix} -(tt^{\langle 0 \rangle} - pp^{\langle 0 \rangle})_1 \\ (tt^{\langle 0 \rangle} - pp^{\langle 0 \rangle})_0 \end{bmatrix}, rr_0 \right]^{\langle 0 \rangle} \end{array} \right.$$

Массив $\{T_i\}$ уже определен, как говорилось выше, а расчетные итерационные формулы для массивов $\{P_i\}, \{Q_i\}, \{K_i\}, \{R_i\}$ выглядят так:

$$P_i = P_{i-1} + V_P \cdot \Delta T \cdot \frac{K_{i-1} - P_{i-1}}{|K_{i-1} - P_{i-1}|},\ Q_i = T_i + \frac{V_T^2}{V_P^2 - V_T^2} \cdot (T_i - P_i),$$

$$K_i = T_i + V_T \cdot \frac{T_{i+1} - T_i}{|T_{i+1} - T_i|} \cdot t, \qquad R_i = \frac{V_T^2}{V_P^2 - V_T^2} \cdot |T_i - P_i|.$$

$$\left| \begin{array}{l} \text{for } i \in 1..mm \\ \left| \begin{array}{l} pp^{\langle i \rangle} \leftarrow pp^{\langle i-1 \rangle} + v_p \cdot \Delta t \cdot \frac{kk^{\langle i-1 \rangle} - pp^{\langle i-1 \rangle}}{\left| kk^{\langle i-1 \rangle} - pp^{\langle i-1 \rangle} \right|} \\ qq^{\langle i \rangle} \leftarrow \begin{bmatrix} (tt^{\langle i \rangle})_0 \\ (tt^{\langle i \rangle})_1 \end{bmatrix} + \frac{v_t^2}{v_p^2 - v_t^2} \cdot \left[\begin{bmatrix} (tt^{\langle i \rangle})_0 \\ (tt^{\langle i \rangle})_1 \end{bmatrix} - \begin{bmatrix} (pp^{\langle i \rangle})_0 \\ (pp^{\langle i \rangle})_1 \end{bmatrix} \right] \\ rr_i \leftarrow \frac{v_t \cdot v_p}{v_p^2 - v_t^2} \cdot \left| tt^{\langle i \rangle} - pp^{\langle i \rangle} \right| \\ kk^{\langle i \rangle} \leftarrow \text{Point}_{\text{Apollo}} \left[tt^{\langle i \rangle}, qq^{\langle i \rangle}, \begin{bmatrix} -(tt^{\langle i \rangle} - pp^{\langle 0 \rangle})_1 \\ (tt^{\langle i \rangle} - pp^{\langle 0 \rangle})_0 \end{bmatrix}, rr_i \right]^{\langle 0 \rangle} \end{array} \right. \end{array} \right.$$

Выходные данные нами записаны в структурированную переменную в виде вектор-строки из пяти элементов $[\{P_i\}\ \{T_i\}\ \{Q_i\}\ \{K_i\}\ \{R_i\}]$. Если пятый элемент такой структурированной переменной $\{R_i\}$ это вектор-столбец из скалярных значений радиусов из $m + 1$ элементов, то $\{P_i\}$, $\{T_i\}$, $\{Q_i\}$, $\{K_i\}$

это вектор-столбцы из $m+1$ значений векторов размерности $[2 \times 1]$. Допустим, к примеру $T_i = \begin{bmatrix} X_{T\,i} \\ Y_{T\,i} \end{bmatrix}$.

```
(PP  TT  QQ  KK) ← (pp^⟨0⟩^T  tt^⟨0⟩^T  qq^⟨0⟩^T  kk^⟨0⟩^T)
for i ∈ 1..mm
    PP ← stack(PP, pp^⟨i⟩^T)
    TT ← stack(TT, tt^⟨i⟩^T)
    QQ ← stack(QQ, qq^⟨i⟩^T)
    KK ← stack(KK, kk^⟨i⟩^T)
(PP  TT  QQ  KK  rr)
```

В следующей процедуре вычисляется массивы точек прямых линий, проходящих от точек $\{P_i\}$ до точек $\{T_i\}$.

```
offcut(pp, tt) :=
    for i ∈ 0..m
        p_i ← (pp_{i,0}  pp_{i,1})^T
        t_i ← (tt_{i,0}  tt_{i,1})^T
        for j ∈ 0..N
            pt_{i,j} ← (1 − j/N) · p_i + (j/N) · t_i
        L_i ← pt_{i,0}^T
        for j ∈ 1..N
            L_i ← stack(L_i, pt_{i,j}^T)
    LL ← L_0
    for i ∈ 1..m
        LL ← stack(LL, L_i)
    LL
```

Входными данными для этой процедуры являются структурированные массивы точек $\{P_i\}$, $\{T_i\}$. Размерность массивов $\{P_i\}$, $\{T_i\}$ есть $[(m+1) \times 1]$. На выходе выдаются точки однопараметрического множества прямых L_i, соединяющих точки P_i и T_i. Поскольку, структурированная переменная L хранит точки не одной прямой, а однопараметрического множества прямых, то размерность массива $\{L_i\}$ будет $\left[\left((m+1) \cdot (N+1)\right) \times 1\right]$. В этой процедуре

переменные N и m являются глобальными. Хотя, если их сделать локальными суть нашей программы не поменяется. Каждый программист может реализовать предложенный алгоритм по-своему.

Далее, создается процедура, рассчитывающая точки однопараметрического множества окружностей.

```
cir(Q_t, r_t) := | for i ∈ 0..m
                 |   q_i ← (Q_t[i,0]  Q_t[i,1])^T
                 |   for j ∈ 0..N
                 |     s[i,j] ← (r_t)_i · ( cos(j/N · 2π) ) + q_i
                 |                         ( sin(j/N · 2 · π) )
                 |   L_i ← s[i,0]^T
                 |   for j ∈ 1..N
                 |     L_i ← stack(L_i, s[i,j]^T)
                 | LL ← L_0
                 | for i ∈ 1..m
                 |   LL ← stack(LL, L_i)
                 | LL
```

Формат входных данных является следующим: одномерный массив $[(m+1)\times 1]$ точек $\{Q_i\}$, являющихся центрами окружности Аполлония, каждая точка Q_i есть вектор-столбец из ее координат, одномерный массив $\{R_i\}$ из радиусов окружностей Аполлония.

Выходными данными является объединенный массив L множества точек, принадлежащих однопараметрическому множеству окружностей Аполлония. Размерность массива L будет таким $\left[\left((m+1)\cdot(N+1)\right)\times 1\right]$.

Далее следует обращение к описанной выше процедуре $Triangle$ для получения массивов всех характеристических точек.

```
Pl := Triangle(P, T_arr, V_p, V_T, N, m, ΔT)_0,0
```
Массив точек преследователя

```
Tl := Triangle(P, T_arr, V_p, V_T, N, m, ΔT)_0,1
```
Массив точек цели (отметим, что данная информация избыточна)

```
Ql := Triangle(P, T_arr, V_p, V_T, N, m, ΔT)_0,2
```
Массив центров окружностей Аполлония

$Kl := Triangle(P, T_{arr}, V_p, V_T, N, m, \Delta T)_{0,3}$ **Массив характеристических точек на окружности Аполлония**

$rl := Triangle(P, T_{arr}, V_p, V_T, N, m, \Delta T)_{0,4}$ **Массив радиусов окружностей Аполлония**

Далее, последует блок операторов, необходимых для динамического описания нашей задачи преследования на плоскости методом параллельного сближения с визуализацией окружностей Аполлония и некоторых характеристических линий и точек.

Создаем две ранжированные переменные $i = 0..m$ и $j = 0..N$, что является аналогом организации двойного цикла.

$i := 0..m \qquad j := 0..N$

Выполняем построение характеристических линий. Для примера рассмотрим рисунок 1.14

$$pt_{xl_{i,j}} := \left(1 - \frac{j}{N}\right) \cdot Pl_{i,0} + \frac{j}{N} \cdot Tl_{i,0}$$
$$pt_{yl_{i,j}} := \left(1 - \frac{j}{N}\right) \cdot Pl_{i,1} + \frac{j}{N} \cdot Tl_{i,1}$$

Построение отрезка $[P_iT_i]$

$$cl_{x_{i,j}} := rl_i \cdot \cos\left(\frac{j}{N} \cdot 2\pi\right) + Ql_{i,0}$$
$$cl_{y_{i,j}} := rl_i \cdot \sin\left(\frac{j}{N} \cdot 2\pi\right) + Ql_{i,1}$$

Построение окружности радиуса R_i с центром в точке Q_i

$$tq_{xl_{i,j}} := \left(1 - \frac{j}{N}\right) \cdot Tl_{i,0} + \frac{j}{N} \cdot Ql_{i,0}$$
$$tq_{yl_{i,j}} := \left(1 - \frac{j}{N}\right) \cdot Tl_{i,1} + \frac{j}{N} \cdot Ql_{i,1}$$

Построение отрезка $[Q_iT_i]$

$$pk_{xl_{i,j}} := \left(1 - \frac{j}{N}\right) \cdot Pl_{i,0} + \frac{j}{N} \cdot Kl_{i,0}$$
$$pk_{yl_{i,j}} := \left(1 - \frac{j}{N}\right) \cdot Pl_{i,1} + \frac{j}{N} \cdot Kl_{i,1}$$

Построение отрезка $[P_iK_i]$

$$qk_{xl_{i,j}} := \left(1 - \frac{j}{N}\right) \cdot Ql_{i,0} + \frac{j}{N} \cdot Kl_{i,0}$$
$$qk_{yl_{i,j}} := \left(1 - \frac{j}{N}\right) \cdot Ql_{i,1} + \frac{j}{N} \cdot Kl_{i,1}$$

Построение отрезка $[Q_iK_i]$

Формируем матрицы $[(m+1) \times 2]$ вышеперечисленных характеристических линий.

$i := 0..m$

$pt_i := augment\left[\left(pt_{xl}^T\right)^{\langle i \rangle}, \left(pt_{yl}^T\right)^{\langle i \rangle}\right]$

$$tq_i := augment\left[\left(tq_{xl}^{T}\right)^{\langle i \rangle}, \left(tq_{yl}^{T}\right)^{\langle i \rangle}\right]$$

$$pk_i := augment\left[\left(pk_{xl}^{T}\right)^{\langle i \rangle}, \left(pk_{yl}^{T}\right)^{\langle i \rangle}\right]$$

$$qk_i := augment\left[\left(qk_{xl}^{T}\right)^{\langle i \rangle}, \left(qk_{yl}^{T}\right)^{\langle i \rangle}\right]$$

$$cl_i := augment\left[\left(cl_x^{T}\right)^{\langle i \rangle}, \left(cl_y^{T}\right)^{\langle i \rangle}\right]$$

$$Tre_i := stack\left(pk_i, qk_i, tq_i, pt_i, cl_i\right)$$

Формируем матрицу вывода на экран всех указанных характеристических линий размерности $[5 \cdot (m+1) \times 2]$.

$$Tre_i := stack\left(pk_i, qk_i, tq_i, pt_i, cl_i\right)$$

Ситуация, соответствующая определенному дискретному промежутку времени, описывается переменной Tre_{FRAME} , где $FRAME$ - это системная переменная анимации.

В приведенных ниже операторах подготовлены все точки, которые будут выведены экран, независимо от времени.

$$P_t := Triangle\left(P, T_{arr}, V_p, V_T, N, m, \Delta T\right)_{0,0}$$

$$r_t := Triangle\left(P, T_{arr}, V_p, V_T, N, m, \Delta T\right)_{0,4}$$

$$K_t := Triangle\left(P, T_{arr}, V_p, V_T, N, m, \Delta T\right)_{0,3}$$

$$Q_t := Triangle\left(P, T_{arr}, V_p, V_T, N, m, \Delta T\right)_{0,2}$$

$$T_t := Triangle\left(P, T_{arr}, V_p, V_T, N, m, \Delta T\right)_{0,1}$$

В приведенных ниже операторах подготовлены все линии, которые будут выведены экран, независимо от времени.

$$PT := offcut\left(P_t, T_t\right)$$

$$TQ := offcut\left(T_t, Q_t\right)$$

$$PK := offcut\left(P_t, K_t\right)$$

$$QK := offcut\left(Q_t, K_t\right)$$

$$TK := offcut\left(T_t, K_t\right)$$

$$S := cir\left(Q_t, r_t\right)$$

Объединение всех выводимых точек в единый массив.

$$L := stack(PT, TQ, PK, QK, TK)$$

Объединение всех точек прямолинейных участков в один единый массив.

$W := \text{stack}(P_t, T_t, Q_t, K_t)$

Результат работы нашей тестовой программы изображен на рисунке 1.15. Если в настройках анимации программы MathCAD указать число кадров m, то с частотой кадров в секунду, равной 5 мы сможем получить такое анимированное изображение: https://youtu.be/rsMGA1ICo7M [48] .

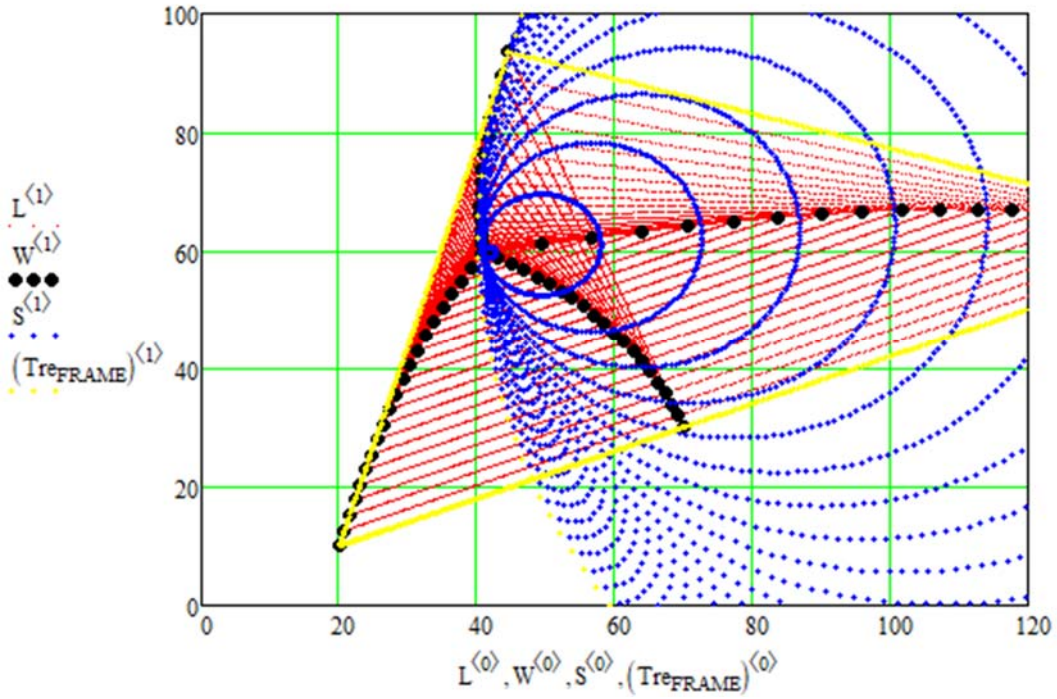


Рисунок 1.15. Визуализация окружностей Аполлония

1.5 КОММЕНТАРИИ К ПРОГРАММЕ ПРОГРАММЫ МОДЕЛИРОВАНИЯ МЕТОДА ПАРАЛЛЕЛЬНОГО СБЛИЖЕНИЯ НА ПЛОСКОСТИ С ЗАДАННЫМИ ОГРАНИЧЕНИЯМИ НА КРИВИЗНУ

В данном разделе мы подробно разберем листинг тестовой программы, которая выполняет реализацию модели задачи преследования на плоскости методом, приближенным к методу параллельного сближения, с заданными ограничениями по кривизне траекторий.

Преследование также происходит на плоскости размерами 100×100 метров.

Для того, чтобы построить одну линию из однопараметрического множества линий, показанных на рисунке 1.7, рассмотрим локальный базис (e_1, C, e_2) с центром в точке C центре окружности радиуса Rad.

Центр окружности расположен таким образом, что отрезок $[PC]$ перпендикулярен вектору скорости преследователя V_P (Рисунок 1.16). Ось абсцисс e_1 системы координат (e_1, C, e_2) направлена от точки C до точки T (Рисунок 1.16). Ось e_2 , перпендикулярна оси e_1, и направление ее, из возможных направлений, выбрано такое, как показано на рисунке 1.16.

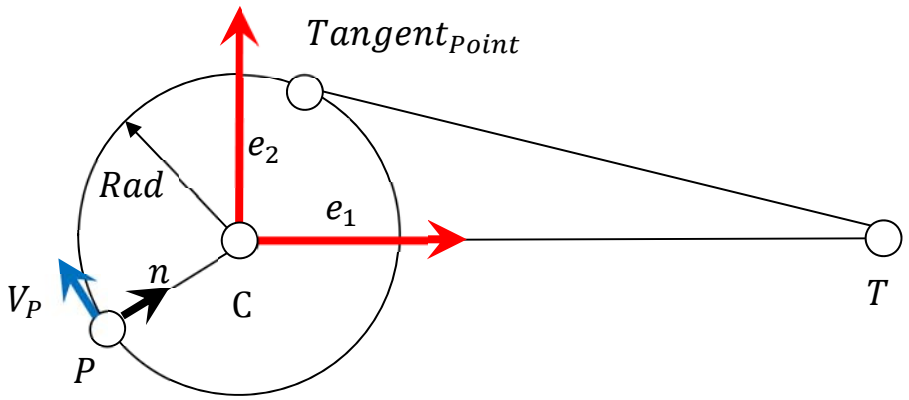


Рисунок 1.16. Определение точки касания в локальном базисе

Нашей задачей на данном этапе является моделирование составной кривой, состоящей из сегмента дуги $\left(P, \widetilde{Tangent}_{Point}\right)$ и прямолинейного отрезка $[Tangent_{Point}, T]$. Положение точки C в мировой системе координат (h_1, O, h_2) формируется следующим образом: $C = P + n \cdot Rad$, где n - единичный вектор, перпендикулярный вектору скорости V_p преследователя (Рисунок 1.16),

$$n = \frac{\begin{bmatrix} V_{p_y} \\ -V_{p_x} \end{bmatrix}}{|V_p|}.$$

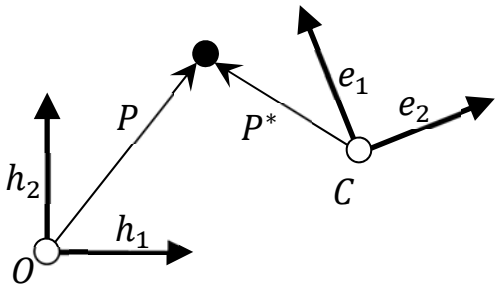


Рисунок 1.17 Преобразование базиса

Координаты точки T в базисе (e_1, C, e_2) будут $\begin{bmatrix} |CT| \\ 0 \end{bmatrix}$. Координаты точки P в базисе (e_1, C, e_2) будут такими $P^* = \begin{bmatrix} (P-C)\cdot e_1 \\ (P-C)\cdot e_2 \end{bmatrix}$, где значения векторов P, C, e_1, e_2 выражены в мировой системе координат (h_1, O, h_2) (Рисунок 1.17).

В рассматриваемом программном коде принято $|CT| = Dist_0$.

$\text{Rad} := 2$ **Ограничение по кривизне прогнозируемых траекторий движения**

$\text{Dist}_0 := 6$ **Расстояние от центра окружности до цели ($|CT|$)**

Начальные координаты цели T базисе (e_1, C, e_2).

$$\text{Target}_0 := \begin{pmatrix} \text{Dist}_0 \\ 0 \end{pmatrix}$$

Координаты точки касания $Tangent_{Point}$ в базисе (e_1, C, e_2).

$$\text{Tangent}_{\text{Point}} := \begin{pmatrix} \frac{\text{Rad}^2}{\text{Dist}_0} \\ \frac{\text{Rad}}{\text{Dist}_0}\cdot\sqrt{\text{Dist}_0^{\,2} - \text{Rad}^2} \end{pmatrix} = \begin{pmatrix} 0.667 \\ 1.886 \end{pmatrix}$$

Координаты точки $Tangent_{Point}$ есть следствие решения системы уравнений:

$$\begin{cases} (Target_0 - Tangent_{Point})\cdot Tangent_{Point} = 0 \\ (Target_0 - Tangent_{Point})^2 + Tangent_{Point}^{\,2} = Rad^2 \end{cases}.$$

Параметры множества точек на окружности.

$$\text{Circle}(R, t) := R\begin{pmatrix} \cos(-t) \\ \sin(-t) \end{pmatrix}$$ **Параметр множества точек окружности $t \in [0, 2\pi]$**

Множество точек на отрезке $[Tangent_{Point}, Target_0]$, соединяющем цель и точку касания на окружности.

$\text{Line}(\text{Target}_0, \text{Tangent}_{\text{Point}}, t) := (1 - t)\cdot\text{Tangent}_{\text{Point}} + t\cdot\text{Target}_0$ **Параметр множества точек** $t \in [0, 1]$

Стартовые конфигурации множеств точек окружности и отрезка.

$\text{Circle}_0(t) := \text{Circle}(\text{Rad}, t)$ **Точки окружности в системе координат** (e_1, C, e_2)

$\text{Line}_0(t) := \text{Line}(\text{Target}_0, \text{Tangent}_{\text{Point}}, t)$ **Точки отрезка в системе координат** (e_1, C, e_2)

Вывод на экран стартовой конфигурации множеств точек окружности и касательного отрезка $[Tangent_{Point}, Target_0]$ в системе координат (e_1, C, e_2).

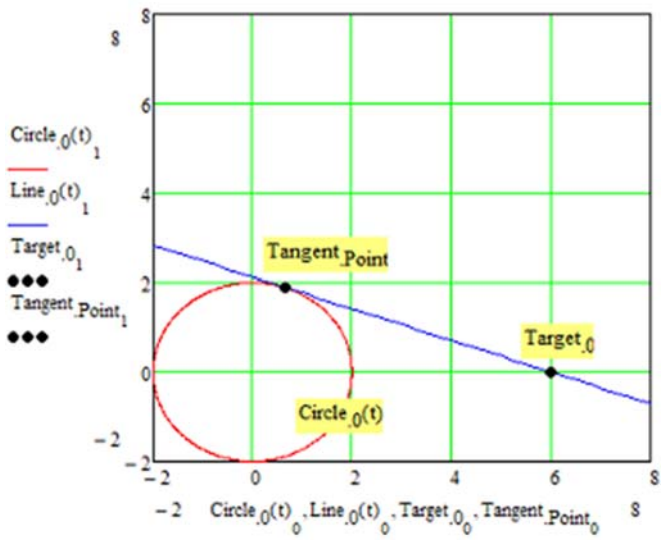


Рисунок 1.18. Вывод на экран начальной конфигурации задачи преследования в системе координат (e_1, C, e_2)

Задание значений скоростей преследователя и цели.

$V_P := 18$ **Скорость преследователя 18 м/с, что соответствует 64.8 км/час**

$V_T := 13$ **Скорость целия 13 м/с, что соответствует 46.8 км/час**

Скорости движения объектов в нашей тестовой программе, иллюстрирующую задачу преследования на плоскости, мы выбрали сравнимые со скоростями движения автомобилей.

$\text{Vector}_{\text{Target}} := \dfrac{\begin{pmatrix} 0.0 \\ 1 \end{pmatrix}}{\sqrt{1 + 0.}}$ **Единичный вектор направления движения цели**

$\Delta T := 0.015$ **Задаем временной промежуток итерационного процесса**

Задаем единичный касательный вектор к окружности $Circle_0(t)$.

$$\text{Tangent}_{\text{Vector}}(t) := \frac{\frac{d}{dt}\text{Circle}_0(t)}{\sqrt{\frac{d}{dt}\text{Circle}_0(t)^{T} \cdot \frac{d}{dt}\text{Circle}_0(t)}} \rightarrow \begin{pmatrix} -\frac{\sin(t)}{\sqrt{\cos(t)^2 + \sin(t)^2}} \\ -\frac{\cos(t)}{\sqrt{\cos(t)^2 + \sin(t)^2}} \end{pmatrix}$$

В этой функции рассчитывается массив точек от точки, соответствующей значению параметра $t = \alpha_s$ до точки K на окружности $Circle_0(t)$.

$$\begin{aligned}
&\text{Arc}_{\text{Segment}}(K, \alpha_s) := \\
&\quad n \leftarrow 20 \\
&\quad \alpha_f \leftarrow 2\pi - \text{acos}\left(\frac{K_0}{|K|}\right) \\
&\quad \text{for } i \in 0..n \\
&\quad\quad x_i \leftarrow \text{Circle}_0\left[\left(1 - \frac{i}{n}\right)\cdot\alpha_s + \frac{i}{n}\cdot\alpha_f\right]_0 \\
&\quad\quad y_i \leftarrow \text{Circle}_0\left[\left(1 - \frac{i}{n}\right)\cdot\alpha_s + \frac{i}{n}\cdot\alpha_f\right]_1 \\
&\quad P \leftarrow \text{augment}(x, y) \\
&\quad P
\end{aligned}$$

Получение данного массива от точки, соответствующей параметру $t = \frac{5}{4}\pi$, до точки $Tangent_{Point}$, состоящего из 20 значений.

$$\text{Arc}_{\text{Segment}.0} := \text{Arc}_{\text{Segment}}\left(\text{Tangent}_{\text{Point}}, \frac{5}{4}\pi\right)$$

Расчет массива из 20 точек прямолинейного отрезка $[Tangent_{Point}, Target_0]$. Не забываем, что выше мы определили функцию прямой $Line_0(t)$, проходящую через точки $Tangent_{Point}$ и $Target_0$.

$$\begin{aligned}
&\text{Line}_{\text{Segment}} := \\
&\quad n \leftarrow 20 \\
&\quad \text{for } i \in 0..20 \\
&\quad\quad x_i \leftarrow \text{Line}_0\left(\frac{i}{n}\right)_0 \\
&\quad\quad y_i \leftarrow \text{Line}_0\left(\frac{i}{n}\right)_1 \\
&\quad P \leftarrow \text{augment}(x, y) \\
&\quad P
\end{aligned}$$

$$\text{Line}_{\text{Segment}.0} := \text{Line}_{\text{Segment}}$$

Формирование объединенного упорядоченного массива из точек дуги $\widetilde{PTangent_{Point}}$ и прямолинейного отрезка $[Tangent_{Point}, Target_0]$ (Рисунок 1.16). Объединенный массив точек начинается с точки $Target_0$.

$$Trajectory_{Point.0} := reverse\left(stack\left(Arc_{Segment.0}, submatrix\left(Line_{Segment.0}, 1, rows\left(Line_{Segment.0}\right) - 1, 0, 1\right)\right)\right)$$

Все эти процедуры требуются для того, чтобы на основе объединенного массива $Trajectory_{Point.0}$ гладкую кривую, после сплайн-интерполяции.

Для этого вводится ранжированная переменная i_{fp}, число элементов которой равно числу элементов массива $Trajectory_{Point.0}$, с индексацией от 0.

$$i_{fp} := 0 .. rows\left(Trajectory_{Point.0}\right) - 1$$

$$Formal_{Parameter_{i_{fp}}} := i_{fp}$$

Далее вводится формальный параметр, который совпадает с ранее введенной переменной, но мы его ввели, чтобы путаницы и недопонимания.

В следующих процедурах на основе массива $Trajectory_{Point.0}$ производится расчет коэффициентов кубического сплайна при помощи встроенных процедур системы MathCAD. Массив $Trajectory_{Point.0}$ представляет собой двухстолбцовую матрицу. В первом столбце хранятся X – координаты, во втором Y - координаты.

$$Coef_x := cspline\left(Formal_{Parameter}, Trajectory_{Point.0}^{\langle 0 \rangle}\right)$$

$$Coef_y := cspline\left(Formal_{Parameter}, Trajectory_{Point.0}^{\langle 1 \rangle}\right)$$

Далее, следуют функции построения интерполяции от непрерывного параметра f_p, узловыми точками которого являются элементы массива $Trajectory_{Point.0}$.

$$f_{x.formal}\left(f_p\right) := interp\left(Coef_x, Formal_{Parameter}, Trajectory_{Point.0}^{\langle 0 \rangle}, f_p\right)$$

$$f_{y.formal}\left(f_p\right) := interp\left(Coef_y, Formal_{Parameter}, Trajectory_{Point.0}^{\langle 1 \rangle}, f_p\right)$$

В итоге построена векторная функция от непрерывного параметра f_p.

$$Predator_{formal}\left(f_p\right) := \begin{pmatrix} f_{x.formal}\left(f_p\right) \\ f_{y.formal}\left(f_p\right) \end{pmatrix}$$

Для того, чтобы перейти в построенной векторной функции от формального параметра f_p к параметру длины дуги, нам необходимо сначала построить массив узловых производных первого порядка. Для этого мы использовали двухточечную итерационную схему:

$$\frac{dX_i}{dt} = \frac{X_{i+1} - X_{i-1}}{2 \cdot \Delta t}.$$

Конечный результат объединяется в одну двустолбцовую матрицу.

$$\begin{aligned}
&\mathrm{Diff_{Point}} := \left|\begin{array}{l}
\mathrm{hT} \leftarrow \mathrm{Formal_{Parameter}}_1 - \mathrm{Formal_{Parameter}}_0 \\
\mathrm{D} \leftarrow (0 \;\; 0) \\
\text{for } i \in 0..\mathrm{rows}(\mathrm{Formal_{Parameter}}) - 1 \\
\quad \left|\begin{array}{l}
\text{if } i = 0 \\
\quad \left|\begin{array}{l}
\mathrm{Dx} \leftarrow \dfrac{\mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_{i+1}\right)_0 - \mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_i\right)_0}{\mathrm{hT}} \\
\mathrm{Dy} \leftarrow \dfrac{\mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_{i+1}\right)_1 - \mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_i\right)_1}{\mathrm{hT}}
\end{array}\right. \\
\text{if } i = \mathrm{rows}(\mathrm{Formal_{Parameter}}) - 1 \\
\quad \left|\begin{array}{l}
\mathrm{Dx} \leftarrow \dfrac{\mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_i\right)_0 - \mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_{i-1}\right)_0}{\mathrm{hT}} \\
\mathrm{Dy} \leftarrow \dfrac{\mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_i\right)_1 - \mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_{i-1}\right)_1}{\mathrm{hT}}
\end{array}\right. \\
\text{if } (i \neq 0) \wedge \left(i \neq \mathrm{rows}(\mathrm{Formal_{Parameter}}) - 1\right) \\
\quad \left|\begin{array}{l}
\mathrm{Dx} \leftarrow \dfrac{\mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_{i+1}\right)_0 - \mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_{i-1}\right)_0}{2\mathrm{hT}} \\
\mathrm{Dy} \leftarrow \dfrac{\mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_{i+1}\right)_1 - \mathrm{Predator_{formal}}\left(\mathrm{Formal_{Parameter}}_{i-1}\right)_1}{2\mathrm{hT}}
\end{array}\right. \\
\mathrm{D} \leftarrow \mathrm{stack}[\mathrm{D}, (\mathrm{Dx} \;\; \mathrm{Dy})]
\end{array}\right. \\
\mathrm{D} \leftarrow \mathrm{submatrix}(\mathrm{D}, 1, \mathrm{rows}(\mathrm{Formal_{Parameter}}), 0, 1)
\end{array}\right.
\end{aligned}$$

Далее, по полученным массивам первых производных от формального параметра проводится кубическая сплайн-интерполяция, чтобы получить непрерывные координатные функции первой производной нашей рассчитываемой линии от непрерывного параметра f_p.

$$\mathrm{CoeffDiff_X} := \mathrm{cspline}\left(\mathrm{Formal_{Parameter}}, \mathrm{Diff_{Point}}^{\langle 0 \rangle}\right)$$

$$CoeffDiff_Y := cspline\left(Formal_{Parameter}, Diff_{Point}^{\langle 1 \rangle}\right)$$

$$d_X(f_p) := interp\left(CoeffDiff_X, Formal_{Parameter}, Diff_{Point}^{\langle 0 \rangle}, f_p\right)$$

$$d_Y(f_p) := interp\left(CoeffDiff_Y, Formal_{Parameter}, Diff_{Point}^{\langle 1 \rangle}, f_p\right)$$

Для того, чтобы получить производную длины дуги от формального параметра:

$$\frac{dt}{ds} = \frac{1}{\sqrt{\frac{dx}{dt}^2 + \frac{dy}{dt}^2}},$$

Мы составляем следующее дифференциальное уравнение первого порядка для передачи во встроенные решатели системы MathCAD.

$$Jc\left(Arc_{length}, f_p\right) := \frac{1}{\sqrt{\left(d_X(f_p)\right)^2 + \left(d_Y(f_p)\right)^2}}$$

Для решения задачи Коши нам понадобятся начальные условия.

$Start_0 := 0$ **Этот оператор при обращении к встроенному решателю будет использоваться как условие $f_p = 0$, то есть отсчет интегрирования начинается от точки $Target_0$**

Задаем число отрезков на промежутке интегрирования.

$NumberOfKnot := 200$ **Отрезок интегрирования по параметру длины дуги разбивается на 200 равных интервалов**

Для решения задачи Коши методом Рунге – Кутты четвертого порядка примем условие, что $s(0) = 0$. Что выбрать конечный предел интегрирования по s, введем приближенный метод расчета длины дуги, а именно расчет методом накопленных хорд. В тексте программы это переменная $ChordLength$.

$$ChordLength := \left| \begin{array}{l} Temp \leftarrow 0 \\ \text{for } i \in 0..rows\left(Trajectory_{Point.0}\right) - 2 \\ \quad Temp \leftarrow Temp + \sqrt{\left[\left(Trajectory_{Point.0}^{\langle 0 \rangle}\right)_{i+1} - \left(Trajectory_{Point.0}^{\langle 0 \rangle}\right)_i\right]^2} \\ \qquad + \left[\left(Trajectory_{Point.0}^{\langle 1 \rangle}\right)_{i+1} - \left(Trajectory_{Point.0}^{\langle 1 \rangle}\right)_i\right]^2 \\ Temp \end{array} \right.$$

Производится суммирование длин отрезков нашей составной кривой

Производится обращение к встроенному решателю системы MathCAD.

$$Formal_{Arc} := rkfixed(Start_0, 0, ChordLength, NumberOfKnot, Jc)$$

Результат обращения хранится в двухстолбцовой матрице $Formal_{Arc}$. В первом столбце хранится разбитый на 200 участков отрезок $[0, ChordLength]$ параметра длины дуги, а во втором столбце хранится соответствующий по значениям формальный параметр f_p. Другими словами, отрезок формального параметра f_p от 0 до последнего элемента массива $Formal_{Parameter}$ взаимно-однозначно отображается на отрезок параметра длины дуги $[0, ChordLength]$.

Для нахождения зависимости формального параметра f_p от длины дуги, в нашем коде это переменная $dist$, мы произвели кубическую интерполяцию на основе массива $Formal_{Arc}$.

$$Coeff_{Formal\_Arc} := cspline(Formal_{Arc}^{\langle 0 \rangle}, Formal_{Arc}^{\langle 1 \rangle})$$

$$Form(dist) := interp(Coeff_{Formal\_Arc}, Formal_{Arc}^{\langle 0 \rangle}, Formal_{Arc}^{\langle 1 \rangle}, dist)$$

В результате нами получена непрерывная функция $Form(dist)$ непрерывного формального параметра f_p от непрерывного параметра длины дуги $dist$. Получим координатные функции прогнозируемой траектории преследователя от длины дуги на начальном этапе, в момент начала преследования в системе координат (e_1, C, e_2) (Рисунок 1.19).

$$Predator_x(dist) := Predator_{formal}(Form(dist))_0$$

$$Predator_y(dist) := Predator_{formal}(Form(dist))_1$$

Координатные функции прогнозируемой траектории преследователя.

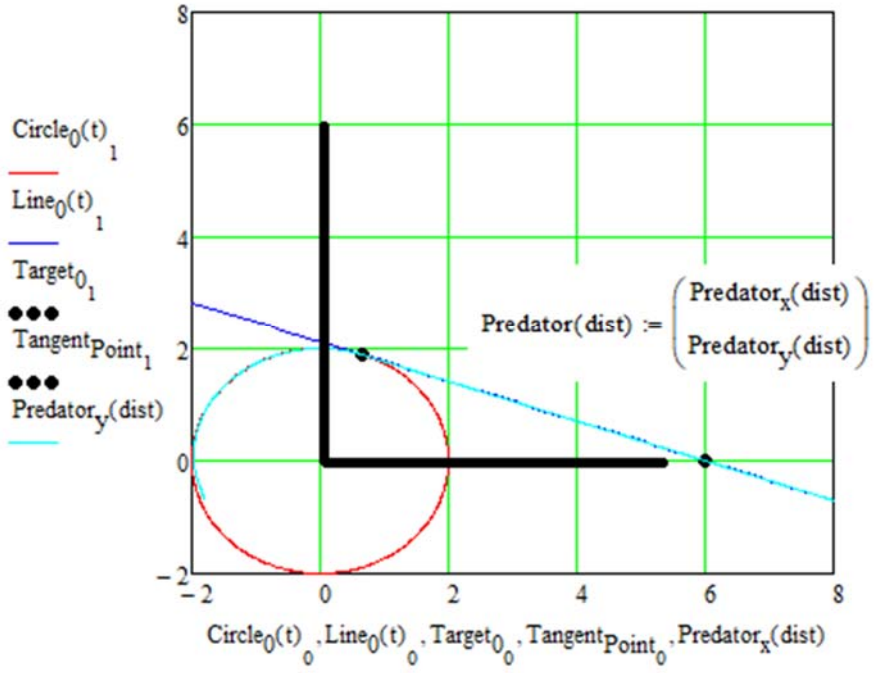


Рисунок 1.19. Прогнозируемая траектория преследователя

Векторная функция прогнозируемой траектории преследователя в начальный момент времени. В процессе построения точек траектории преследователя данная прогнозируемая траектория будет сдвигаться параллельной самой себе, в зависимости от шага цели.

$$Predator(dist) := \begin{pmatrix} Predator_x(dist) \\ Predator_y(dist) \end{pmatrix}$$

Еще раз повторимся, определим отрезок, на котором параметризуется начальная прогнозируемая траектория преследователя.

$Dist_{start} := 0$

Значение длины дуги в точке цели

$Dist_{final} := root\left(Predator_x(q) - Circle_0\left(\frac{5}{4}\pi\right)_0, q, 0, 10\right) = 7.906$

Значение длины дуги в точке преследователя

$Predator\left(Dist_{start}\right) = \begin{pmatrix} 6 \\ 0 \end{pmatrix}$

Проверочные значения прогнозируемой траектории

$Predator\left(Dist_{final}\right) = \begin{pmatrix} -1.414 \\ 1.414 \end{pmatrix}$

Зададим траекторию цели, как прямую линию в направлении вектора $Vector_{Target}$ со скоростью V_T. Данные величины заданы выше.

$Target_{line}(t) := Predator\left(Dist_{start}\right) + t \cdot V_T \cdot Vector_{Target}$

Преобразуем линию начальной прогнозируемой траектории в массивы точек.

$N_{point} := 75$ **Количество точек**

$i := 0..N_{point}$ **Ранжированная переменная**

$dist_i := Dist_{start} + \frac{i}{N_{point}} \cdot (Dist_{final} - Dist_{start})$ **Массив длин дуг**

$Tr_{point.X_i} := Predator(dist_i)_0$ **Массив точек по координате *X***

$Tr_{point.Y_i} := Predator(dist_i)_1$ **Массив точек по координате *Y***

Формирование однопараметрического множества параллельных линий путем параллельного сдвига начальной прогнозируемой траектории.

$$Pred(t, nn) := Predator(t) + (Target_{line}(nn) - Predator(Dist_{start}))$$

Сначала формируется вектор сдвига, зависящий от параметра nn положения цели $Target_{line}(nn)$, равный величине $Target_{line}(nn) - Predator(Dist_{startl})$. И путем прибавления к точке $Predator(Dist)$, формируется однопараметрическое множество параллельных линий $Pred(nn, Dist)$, что проиллюстрировано на рисунке 1.20.

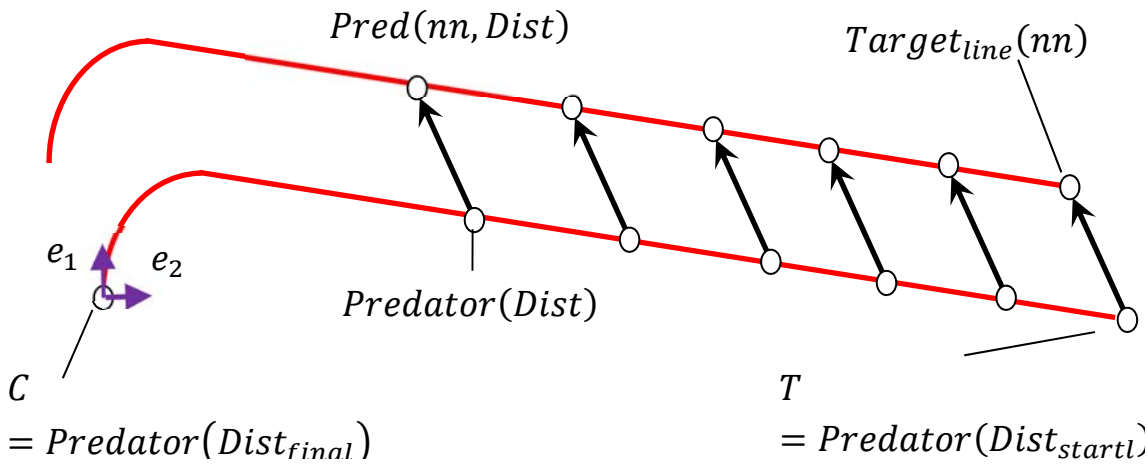


Рисунок 1.20. Формирование однопараметрического множества параллельных линий

Далее, вводим число кадров анимации.

$N_F := 37$

Число кадров анимации, конечно же, можно было рассчитывать автоматически, но мы не стали этого делать, поскольку у нас

демонстрационная программа. Число кадров анимации введено эмпирически. Критерием выбора значения числа кадров анимации служит достижение преследователем цели.

По числу кадров анимации сформируем однопараметрическое множество прогнозируемых траекторий движения преследователя.

$Tt_0 := 0$ — **Ввод начального значения временного промежутка**

$i := 0..N_F$ — **Формирование ранжированной переменной по числу кадров**

$Tt_{i+1} := Tt_i + \Delta T$ — **Формирование дискретных значений времени, распределенных равномерно по временному промежутку** $[\mathbf{0}, \boldsymbol{N_F \cdot \Delta T}]$

$j := 0..N_{point}$ — **Формирование ранжированной переменной по числу точек на отдельной прогнозируемой траектории преследователя**

$dist_j := Dist_{start} + \frac{j}{N_{point}} \cdot (Dist_{final} - Dist_{start})$ — **Формирование массива длин дуг для прогнозируемых траекторий по числу точек на траектории**

$Pr_{X_{i,j}} := Pred(dist_j, Tt_i)_0$ — **Формирование координатного массива абсцисс прогнозируемых траекторий в виде матрицы** $[\boldsymbol{N_F \times N_{Point}}]$

$Pr_{Y_{i,j}} := Pred(dist_j, Tt_i)_1$ — **Формирование координатного массива ординат прогнозируемых траекторий в виде матрицы** $[(\boldsymbol{N_F + 1}) \times (\boldsymbol{N_{Point} + 1})]$

```
M2Column(M) := | m ← cols(M) − 1
               | P ← M^⟨0⟩
               | for i ∈ 1..m
               |   P ← stack(P, M^⟨i⟩)
               | P
```

Преобразование матрицы $[(\boldsymbol{N_F + 1}) \times (\boldsymbol{N_{Point} + 1})]$ **в вектор столбец** $[(\boldsymbol{N_F + 1}) \cdot (\boldsymbol{N_{Point} + 1}) \times \mathbf{1}]$

$X_P := M2Column(Pr_X)$ — **Преобразование матрицы абсцисс в вектор столбец**

$Y_P := M2Column(Pr_Y)$ — **Преобразование матрицы ординат в вектор столбец**

Основной расчетный цикл заключается в итерационном расчете точек траектории движения преследователя.

```
PP := | P_0 ← Predator(Dist_final)
      | d_0 ← Dist_final
      | for i ∈ 1..N_F
      |   | d_i ← root(|Pred(dd, Tt_i) − P_{i−1}| − V_P·ΔT, dd, 0, d_{i−1})
      |   | P_i ← Pred(d_i, Tt_i)
      | P_x ← (P_0)_0
      | P_y ← (P_0)_1
      | for i ∈ 1..N_F
      |   | P_x ← stack[P_x, (P_i)_0]
      |   | P_y ← stack[P_y, (P_i)_1]
      | PP ← augment(P_x, P_y)
```

Суть основного расчетного цикла заключается в следующем: пусть координаты точек P_{i-1}, T_{i-1}, T_i известны (Рисунок 1.21). Также являются известными параметрические уравнения линий из однопараметрического множества прогнозируемых траекторий $L_{i-1}(s)$ и $L_i(s)$. Тогда точка следующего шага итерации преследователя P_i будет являться точкой пересечения окружности радиуса $V_P \cdot \Delta T$ и линии $L_i(s)$, проходящей через точку T_i.

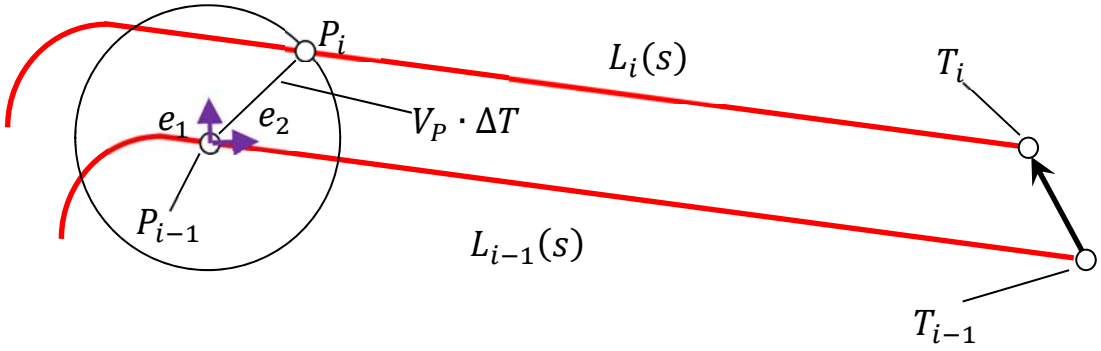


Рисунок 1.21. Расчет следующего шага траектории движения преследователя

Выводятся на экран массив точек цели, который хранится в матрице вида $[(N_F + 1) \times 2]$.

```
TT := | T_x ← Target_line(Tt_0)_0
      | T_y ← Target_line(Tt_0)_1
      | for i ∈ 1..N_F
      |   | T_x ← stack(T_x, Target_line(Tt_i)_0)
      |   | T_y ← stack(T_y, Target_line(Tt_i)_1)
      | T ← augment(T_x, T_y)
      | T
```

Выводим на экран окружности радиуса $V_P \cdot \Delta T$ в каждой точке нахождения преследователя. Каждая окружность представляет собой массив из 61 точек. Полностью массив окружностей хранится в двухстолбцовой матрице размерностью $[61 \cdot (N_F + 1) \times 2]$.

```
CC := | n ← 60
      | for j ∈ 0..n
      |   | x_j ← V_P·ΔT·cos(j·2π/n) + (PP^⟨0⟩)_0
      |   | y_j ← V_P·ΔT·sin(j·2π/n) + (PP^⟨1⟩)_0
      | C ← augment(x,y)
```

```
| for i ∈ 1..N_F
|   | for j ∈ 0..n
|   |   | x_j ← V_P·ΔT·cos(j·2π/n) + (PP^⟨0⟩)_i
|   |   | y_j ← V_P·ΔT·sin(j·2π/n) + (PP^⟨1⟩)_i
|   | xy ← augment(x,y)
|   | C ← stack(C,xy)
```

Производим формирование того, какие переменные будут изменяться при смене кадра.

```
Fr := | q ← augment[(PP^⟨0⟩)_FRAME, (PP^⟨1⟩)_FRAME]
      | T ← augment[(TT^⟨0⟩)_FRAME, (TT^⟨1⟩)_FRAME]
      | F ← stack(q, T)
      | n ← 60
      | for j ∈ 0..n
      |   x_j ← V_P·ΔT·cos(j·2π/n) + (PP^⟨0⟩)_FRAME
      |   y_j ← V_P·ΔT·sin(j·2π/n) + (PP^⟨1⟩)_FRAME
      | C ← augment(x, y)
      | F ← stack(F, C)
      | for i ∈ 0..N_point
      |   X_i ← Pred(dist_i, Tt_FRAME)0
      |   Y_i ← Pred(dist_i, Tt_FRAME)1
      | W ← augment(X, Y)
      | F ← stack(F, W)
      | F
```

В один массив объединены координаты точек положения преследователя и цели в текущий момент времени, координаты точек окружностей радиуса $V_P \cdot \Delta T$ с центром в точке положения преследователя в текущий момент времени и линия, соединяющая преследователя и цель, являющаяся в текущий момент времени прогнозируемой траекторией движения преследователя.

На рисунке 1.22 представлен первый кадр анимированного изображения, на котором показано движение прогнозируемой траектории движения, точек положений преследователя и цели, окружностей с центром в точке положения преследователя. Полностью сам ролик выложен на ресурсе: https://youtu.be/qNXdykK21Z8 [49].

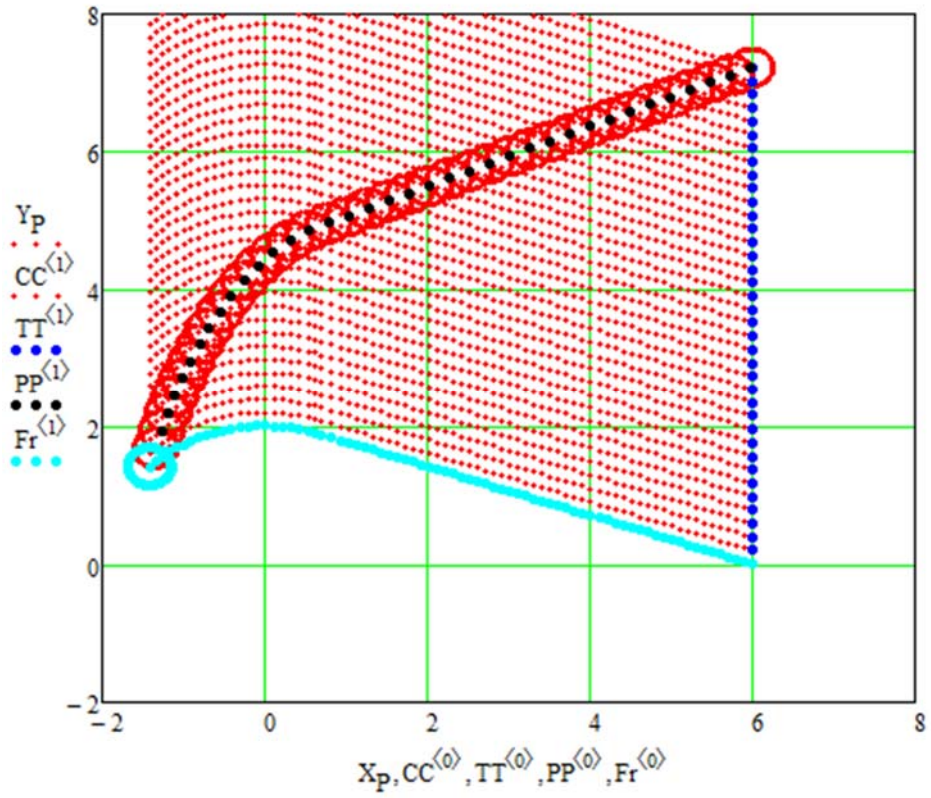


Рисунок 1.22. Визуализация задачи преследования методом параллельного сближения на плоскости

1.6 КОММЕНТАРИИ К ПРОГРАММЕ МОДЕЛИРОВАНИЯ МЕТОДА ПАРАЛЛЕЛЬНОГО СБЛИЖЕНИЯ НА ПОВЕРХНОСТИ

В данном разделе мы подробно разберем листинг тестовой программы, которая выполняет реализацию модели задачи преследования на поверхности методом, приближенным к методу параллельного сближения. Ограничения по кривизне принимаются такие, как и в задаче предыдущего параграфа. То есть радиус кривизны пространственной кривой не может быть меньше наперед заданной величины.

Преследование также происходит на плоскости размерами 100×100 метров. По высоте окно вывода ограничено от 0 до 6 метров.

Мы предварительно в системе AutoCAD поверхность в виде линий уровня. Затем встроенными операторами был импорт линий уровня в систему MathCAD.

```
A1 := READPRN("C:\WorkMathCAD\land2\l1.txt")
```

```
A4 := READPRN("C:\WorkMathCAD\land2\l4.txt")
A2 := READPRN("C:\WorkMathCAD\land2\l2.txt")
A5 := READPRN("C:\WorkMathCAD\land2\l5.txt")
A3 := READPRN("C:\WorkMathCAD\land2\l3.txt")
```

Импорт текстовых файлов со значениями линий уровня

К введенным из системы MathCAD массивам точек в таком виде будет невозможно применить процедуру сплайн-интерполяции без предварительного упорядочивания. Поэтому нами была выбрана для дальнейшей работы процедура полиномиальной регрессии.

Поскольку, программа является демонстрационной, мы позволили себе ввести дополнительный массив точек по краям квадраты с нулевой аппликатой, чтобы избежать осцилляций итоговой поверхности после полиномиальной регрессии.

$i := 0..100$

$Xkr1_i := i \quad Ykr1_i := 0 \quad Zkr1_i := 0$
$Xkr2_i := i \quad Ykr2_i := 100 \quad Zkr2_i := 0$
$Xkr3_i := 0 \quad Ykr3_i := i \quad Zkr3_i := 0$
$Xkr4_i := 100 \quad Ykr4_i := i \quad Zkr4_i := 0$

Ввод точек по краям квадрата

Сформируем объединенную трехстолбцовую матрицу из полученных массивов точек.

$LandPointX := stack(A1^{\langle 0 \rangle}, A2^{\langle 0 \rangle}, A3^{\langle 0 \rangle}, A4^{\langle 0 \rangle}, A5^{\langle 0 \rangle}, Xkr1, Xkr2, Xkr3, Xkr4)$
$LandPointY := stack(A1^{\langle 1 \rangle}, A2^{\langle 1 \rangle}, A3^{\langle 1 \rangle}, A4^{\langle 1 \rangle}, A5^{\langle 1 \rangle}, Ykr1, Ykr2, Ykr3, Ykr4)$
$LandPointZ := stack(A1^{\langle 2 \rangle}, A2^{\langle 2 \rangle}, A3^{\langle 2 \rangle}, A4^{\langle 2 \rangle}, A5^{\langle 2 \rangle}, Zkr1, Zkr2, Zkr3, Zkr4)$

Покоординатное формирование объединенных вектор-столбцов из полученных массивов точек

$LandPoint := (LandPointX \;\; LandPointY \;\; LandPointZ)^T$

Формирование итоговой матрицы

Для процедуры полиномиальной регрессии все предварительные операции проведены.

$CoeffRegressionXY := augment(LandPointX, LandPointY)$

Массивы по абсциссам и ординатам объединяются в одну матрицу

Рассчитываем коэффициенты регрессии.

$RegressionXY := regress(CoeffRegressionXY, LandPointZ, 8)$

Коэффициенты регрессии

$ArrayRangeLandPointX := cols(LandPointX) - 1$

Вспомогательная переменная, отображающая общее количество точек

С полученными коэффициентами регрессии применяется процедура интерполяции. В результате получаем параметрическую поверхность.

$$\text{Surface}(u,v) := \text{interp}\left[\text{RegressionXY}, \text{CoeffRegressionXY}, \text{LandPointZ}, \begin{pmatrix} u \\ v \end{pmatrix}\right]$$

В следующем операторе мы подготавливаем матрицу для вывода на экран.

$$\text{SurfacePicture} := \text{CreateMesh}(\text{Surface}, 0, 100, 0, 100, 200, 200)$$

Нам для дальнейшей работы, необходимо сформировать равномерную сетку на плоскости (XY) в квадрате $[0; 100] \times [0; 100]$.

$$i := 0..100 \qquad j := 0..100$$

$$\text{LandX}_i := i \qquad \text{LandY}_j := j$$

$$\text{LandZ}_{i,j} := \text{Surface}(\text{LandX}_i, \text{LandY}_j)$$

Формирование ранжированных переменных для организации двойного цикла

Определение массивов абсцисс и ординат

Расчет двумерного массива аппликат в узлах сетки

Численный расчет первых частных производных в узлах равномерной сетки. Если считать, что поверхность задана в виде: $Z = Z(X, Y)$, то мы рассчитываем частные производные по схеме:

$$\frac{\partial Z_{i,j}}{\partial x} = \frac{Z_{i+1,j} - Z_{i-1,j}}{X_{i+1} - X_{i-1}}, \frac{\partial Z_{i,j}}{\partial y} = \frac{Z_{i,j+1} - Z_{i,j-1}}{Y_{j+1} - Y_{j-1}}.$$

$$\begin{aligned}
&\text{LandDiffZfromXpoint} := \left|\begin{array}{l}
hX \leftarrow \text{LandX}_1 - \text{LandX}_0 \\
\text{for } i \in 0..100 \\
\quad \text{for } j \in 0..100 \\
\qquad \left|\begin{array}{l}
\text{DxZZZ}_{i,j} \leftarrow \dfrac{\text{LandZ}_{i+1,j} - \text{LandZ}_{i,j}}{\text{LandX}_{i+1} - \text{LandX}_i} \quad \text{if } i = 0 \\
\text{DxZZZ}_{i,j} \leftarrow \dfrac{\text{LandZ}_{i,j} - \text{LandZ}_{i-1,j}}{\text{LandX}_i - \text{LandX}_{i-1}} \quad \text{if } i = 100 \\
\text{DxZZZ}_{i,j} \leftarrow \dfrac{\text{LandZ}_{i+1,j} - \text{LandZ}_{i-1,j}}{2 \cdot hX} \quad \text{if } (i \neq 0) \wedge (i \neq 100)
\end{array}\right. \\
\text{DxZZZ}
\end{array}\right.
\end{aligned}$$

С краевыми условиями такого вида:

$$\frac{\partial Z_{0,j}}{\partial x} = \frac{Z_{1,j} - Z_{0,j}}{X_1 - X_0}, \frac{\partial Z_{i,0}}{\partial y} = \frac{Z_{i,1} - Z_{i,0}}{Y_1 - Y_0}.$$

$$\frac{\partial Z_{N,j}}{\partial x} = \frac{Z_{N,j} - Z_{N-1,j}}{X_N - X_{N-1}}, \frac{\partial Z_{i,N}}{\partial y} = \frac{Z_{i,N} - Z_{i,N-1}}{Y_N - Y_{N-1}}.$$

Мы учитываем то, что сетка является равномерной и квадратной.

$$\text{LandDiffZfromYpoint} := \left| \begin{array}{l} hY \leftarrow LandY_1 - LandY_0 \\ \text{for } i \in 0..100 \\ \quad \text{for } j \in 0..100 \\ \quad\quad DyZZZ_{i,j} \leftarrow \dfrac{LandZ_{i,j+1} - LandZ_{i,j}}{LandY_{j+1} - LandY_j} \text{ if } j = 0 \\ \quad\quad DyZZZ_{i,j} \leftarrow \dfrac{LandZ_{i,j} - LandZ_{i,j-1}}{LandY_j - LandY_{j-1}} \text{ if } j = 100 \\ \quad\quad DyZZZ_{i,j} \leftarrow \dfrac{LandZ_{i,j+1} - LandZ_{i,j-1}}{2 \cdot hY} \text{ if } (j \neq 0) \wedge (j \neq 100) \\ DyZZZ \end{array} \right.$$

Далее, следует сплайн-интерполяция для получения непрерывных функций частных производных.

$$LandXY := augment(LandX, LandY)$$

Для двумерной интерполяции объединяем массивы абсцисс и ординат в одну двухстолбцовую матрицу

$$CoeffLandSplineZX := cspline(LandXY, LandDiffZfromXpoint)$$

$$CoeffLandSplineZY := cspline(LandXY, LandDiffZfromYpoint)$$

Расчет коэффициентов сплайновой поверхности

$$DiffZX(u,v) := interp\left[CoeffLandSplineZX, LandXY, LandDiffZfromXpoint, \begin{pmatrix} u \\ v \end{pmatrix}\right]$$

Интерполяция. Любая система программирования требует осторожного обращения с переменными. Фактически, эти операторы формируют непрерывные функции частных производных $\frac{\partial Z}{\partial x}$ и $\frac{\partial Z}{\partial y}$

$$DiffZY(u,v) := interp\left[CoeffLandSplineZY, LandXY, LandDiffZfromYpoint, \begin{pmatrix} u \\ v \end{pmatrix}\right]$$

Очень важно, чтобы в демонстрационной программе, какой является наша, поверхность и окно вывода выглядело, на наш субъективный взгляд, более или менее приемлемо. Поэтому поднимем поверхность.

$$F_T(x,y) := Surface(x,y) + 2 \qquad F_x(x,y) := DiffZX(x,y) \qquad F_y(x,y) := DiffZY(x,y)$$

Частные производные остаются неизменными.

Переменная для вывода на экран.

$$F_{T.p} := \text{CreateMesh}(F_T, 0, 100, 0, 100, 100, 100)$$

Траектория цели задается горизонтальной проекцией на плоскость XY в параметрическом виде от формального параметра t.

$v_{t.x} := 0 \quad v_{t.y} := 0.7$

Задание горизонтальной проекции направления движения цели

$x_t(t) := v_{t.x} \cdot t + 65$

$y_t(t) := v_{t.y} \cdot t + 20$

Координатные функции горизонтальной проекции движения цели

$$Tar_g(t) := \begin{pmatrix} x_t(t) \\ y_t(t) \\ 0 \end{pmatrix}$$

Векторная функция горизонтальной проекции движения цели в трехмерном пространстве

$tar_{g.p} := \text{CreateSpace}(Tar_g, 0, 100, 100)$

Вывод на экран линии горизонтальной проекции

$$Tar_{g.F}(t) := \begin{pmatrix} x_t(t) \\ y_t(t) \\ F_T(x_t(t), y_t(t)) \end{pmatrix}$$

Проекция линии на горизонтальной плоскости на поверхность

$tar_{F.p} := \text{CreateSpace}(Tar_{g.F}, 0, 100, 100)$

Вывод на экран линии передвижения цели на поверхности от формального параметра

Затем, линия на плоскости XY проецируется на поверхность. Линия передвижения цели на данном этапе зависит от формального параметра, для того, чтобы перейти к параметризации от времени, следует сначала перейти к параметризации от длины дуги.

$$D_J(ss, tt) := \frac{1}{\sqrt{\left(\frac{d}{dtt}x_t(tt)\right)^2 + \left(\frac{d}{dtt}y_t(tt)\right)^2 + \left(F_x(x_t(tt), y_t(tt)) \cdot \frac{d}{dtt}x_t(tt) + F_y(x_t(tt), y_t(tt)) \cdot \frac{d}{dtt}y_t(tt)\right)^2}}$$

В данном операторе составлен якобиан для линии на поверхности $z = F(x, y)$, заданной координатными функциями x_t и y_t, где мы устанавливаем зависимость формального параметра t от длины дуги s:

$$\frac{dt}{ds} = \frac{1}{\sqrt{\frac{dx_t}{dt}^2 + \frac{dy_t}{dt}^2 + \left(\frac{\partial F}{\partial x} \cdot \frac{dx_t}{dt} + \frac{\partial F}{\partial y} \cdot \frac{dy_t}{dt}\right)^2}}.$$

$Jcb(S_j, T_j) := D_J(S_j, T_{j_0})$

Подготовка дифференциального уравнения во встроенные решатели

Далее, задаем начальные условия для задачи Коши.

В момент начала движения цели значение параметра длины дуги примем равным 0: $s = 0$. Значение формального параметра t при начальном значении длины дуги также примем равным 0. В программном коде это будет оператор ниже.

$$TS_0 := 0$$

Обращение к встроенному решателю системы MathCAD происходит в следующем формате: начальное значение формального параметра при начальном значении длины дуги, диапазон значений длины дуги, на котором будет решаться задача Коши. В программном коде это отрезок $[0; 70]$. Поделим данный отрезок на 100 фрагментов.

$$Tar_{Tr} := rkfixed(TS_0, 0, 70, 100, Jcb)$$

Решатель по методу Рунге-Кутты 4 порядка с фиксированным шагом

На выходе возвращается двухстолбцовая матрица из 101 строк. Первый столбец матрицы, это массив длин дуг из диапазона $[0; 70]$, а второй – соответственные значения формального параметра.

Чтобы получить непрерывную формального параметра от длины дуги произведем сплайн-интерполяцию.

$$C_{tr} := cspline(Tar_{Tr}^{\langle 0 \rangle}, Tar_{Tr}^{\langle 1 \rangle})$$

Коэффициенты кубического сплайна

$$t_f(s) := interp(C_{tr}, Tar_{Tr}^{\langle 0 \rangle}, Tar_{Tr}^{\langle 1 \rangle}, s)$$

Оператор интерполяции

Уравнение векторной функции траектории цели на поверхности.

$$Tar(s) := \begin{pmatrix} x_t(t_f(s)) \\ y_t(t_f(s)) \\ F_T(x_t(t_f(s)), y_t(t_f(s))) \end{pmatrix}$$

Вывод на экран траектории движения цели.

$$Tar_{s.p} := CreateSpace(Tar, 0, 70, 50)$$

Введем постоянную скорость движения по поверхности.

$$V_{Target} := 10$$

Введем отрезок дискретизации по параметру длины дуги.

$$\Delta s := (Tar_{Tr}^{\langle 0 \rangle})_1 - (Tar_{Tr}^{\langle 0 \rangle})_0 = 0.7$$

Это будет нам необходимо при численном дифференцировании в узлах траектории движения цели. Узлы расположены равномерно на отрезке [0; 70].

Определение ранжированной переменной по числу точек траектории цели

$$i := 0 .. \mathrm{rows}\left(Tar_{Tr}^{\langle 0 \rangle}\right) - 1$$

Массив точек траектории цели

$$Tar_{point_i} := Tar\left[\left(Tar_{Tr}^{\langle 0 \rangle}\right)_i\right]$$

Вектор производной в начальной точке траектории

$$DTar_0 := \frac{Tar_{point_1} - Tar_{point_0}}{\Delta s}$$

Вектор производной в конечной точке траектории

$$DTar_{\mathrm{rows}\left(Tar_{Tr}^{\langle 0 \rangle}\right)-1} := \frac{Tar_{point_{\mathrm{rows}\left(Tar_{Tr}^{\langle 0 \rangle}\right)-1}} - Tar_{point_{\mathrm{rows}\left(Tar_{Tr}^{\langle 0 \rangle}\right)-2}}}{\Delta s}$$

Вектор производной в точках траектории

$$i := 1 .. \mathrm{rows}\left(Tar_{Tr}^{\langle 0 \rangle}\right) - 2$$

$$DTar_i := \frac{Tar_{point_{i+1}} - Tar_{point_{i-1}}}{2 \cdot \Delta s}$$

Теперь от векторов производных в точках траектории движения цели перейдем к определению производных в узлах по координатам.

Массив по координате *X*

$$DTar_X := \left| \begin{array}{l} DT_X \leftarrow \left(DTar_0\right)_0 \\ \text{for } i \in 1 .. \mathrm{rows}\left(Tar_{Tr}^{\langle 0 \rangle}\right) - 1 \\ \quad DT_X \leftarrow \mathrm{stack}\left[DT_X, \left(DTar_i\right)_0\right] \\ DT_X \end{array} \right.$$

Массив по координате *Y*

$$DTar_Y := \left| \begin{array}{l} DT_Y \leftarrow \left(DTar_0\right)_1 \\ \text{for } i \in 1 .. \mathrm{rows}\left(Tar_{Tr}^{\langle 0 \rangle}\right) - 1 \\ \quad DT_Y \leftarrow \mathrm{stack}\left[DT_Y, \left(DTar_i\right)_1\right] \\ DT_Y \end{array} \right.$$

Массив по координате *Z*

$$DTar_Z := \left| \begin{array}{l} DT_Z \leftarrow \left(DTar_0\right)_2 \\ \text{for } i \in 1 .. \mathrm{rows}\left(Tar_{Tr}^{\langle 0 \rangle}\right) - 1 \\ \quad DT_Z \leftarrow \mathrm{stack}\left[DT_Z, \left(DTar_i\right)_2\right] \\ DT_Z \end{array} \right.$$

Для дальнейшей интерполяции получаем коэффициенты кубических сплайнов.

$$C_{DTar.X} := cspline\left(Tar_{Tr}^{\langle 0 \rangle}, DTar_X\right)$$

$$C_{DTar.Y} := cspline\left(Tar_{Tr}^{\langle 0 \rangle}, DTar_Y\right)$$

$$C_{DTar.Z} := cspline\left(Tar_{Tr}^{\langle 0 \rangle}, DTar_Z\right)$$

Непрерывная векторная функция первой производной траектории движения цели по поверхности от параметра длины дуги, как продукт интерполяции.

$$dTar(s) := \begin{pmatrix} interp\left(C_{DTar.X}, Tar_{Tr}^{\langle 0 \rangle}, DTar_X, s\right) \\ interp\left(C_{DTar.Y}, Tar_{Tr}^{\langle 0 \rangle}, DTar_Y, s\right) \\ interp\left(C_{DTar.Z}, Tar_{Tr}^{\langle 0 \rangle}, DTar_Z, s\right) \end{pmatrix}$$

Начинаем вводить такой параметр как реальное время. В момент начала движения пусть параметр времени начинает отсчет с 0.

$$T_{time_0} := 0$$

Зададим временной промежуток.

$$\Delta T_{time} := \frac{\Delta s}{V_{Target}}$$

Положение цели в момент начала движения.

$$Target_0 := Tar(0)$$

Сформируем положение точек цели через равные временные равномерные промежутки.

$$i := 1 .. rows\left(Tar_{Tr}^{\langle 0 \rangle}\right) - 1$$

Ранжированная переменная по числу узловых точек цели

$$T_{time_i} := T_{time_{i-1}} + \Delta T_{time}$$

Равные временные промежутки

$$Target_i := Target_{i-1} + dTar\left(V_{Target} \cdot T_{time_i}\right) \cdot V_{Target} \cdot \Delta T_{time}$$

Реализация итерационной схемы

Массивы положения цели по координатам. Упорядоченные по времени массивы точек, распределенных в зависимости от времени.

$$X_t := \left| \begin{array}{l} DT_X \leftarrow (Target_0)_0 \\ \text{for } i \in 1..\text{rows}\left(Tar_{Tr}^{\langle 0 \rangle}\right) - 1 \\ \quad DT_X \leftarrow \text{stack}\left[DT_X, (Target_i)_0\right] \\ DT_X \end{array} \right.$$

$$Y_t := \left| \begin{array}{l} DT_Y \leftarrow (Target_0)_1 \\ \text{for } i \in 1..\text{rows}\left(Tar_{Tr}^{\langle 0 \rangle}\right) - 1 \\ \quad DT_Y \leftarrow \text{stack}\left[DT_Y, (Target_i)_1\right] \\ DT_Y \end{array} \right.$$

$$Z_t := \left| \begin{array}{l} DT_Z \leftarrow (Target_0)_2 \\ \text{for } i \in 1..\text{rows}\left(Tar_{Tr}^{\langle 0 \rangle}\right) - 1 \\ \quad DT_Z \leftarrow \text{stack}\left[DT_Z, (Target_i)_2\right] \\ DT_Z \end{array} \right.$$

Определяем предел по времени итерационного процесса. Эмпирически мы подсчитали время, когда преследователь настигнет цель. Определили также число кадров анимации, через которое завершится итерационный процесс.

$T_{final} := 7$ — **Время достижения преследователем цели**

$N_{frame} := 25$ — **Число кадров**

$i := 0..N_{frame}$ — **Ранжированная переменная по числу кадров**

$Time_i := \frac{i}{N_{frame}} \cdot T_{final}$ — **Временные промежутки**

$\Delta T := Time_1 - Time_0 = 0.28$ — **Временной интервал**

$X_{T_{0,0}} := Tar\left(V_{Target} \cdot Time_{FRAME}\right)_0$

$Y_{T_{0,0}} := Tar\left(V_{Target} \cdot Time_{FRAME}\right)_1$

$Z_{T_{0,0}} := Tar\left(V_{Target} \cdot Time_{FRAME}\right)_2$

Точки траектории цели, динамически выводимые на экран, в процессе анимации. Очень важно, в этих операторах формируется вывод по кадрам положения цели

Далее, рассмотрим рисунок 1.23. Пусть положение преследователя P задается координатами $P = \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix}$, а положение цели задается так $T = \begin{bmatrix} T_0 \\ T_1 \\ T_2 \end{bmatrix}$.

Линия, соединяющей точки P и T, являющейся вертикальной проекцией на поверхность $Z = F_T(X, Y)$ прямой линии, соединяющей проекции точек P и T на плоскость XY (Рисунок 1.23):

$$L(P, T, \tau) = \begin{bmatrix} (1-\tau)\cdot P_0 + \tau \cdot T_0 \\ (1-\tau)\cdot P_1 + \tau \cdot T_1 \\ F_T[(1-\tau)\cdot P_0 + \tau \cdot T_0, (1-\tau)\cdot P_1 + \tau \cdot T_1] \end{bmatrix}.$$

Причем, присутствует такая связь: $P_2 = F_T(P_0, P_1)$ и $T_2 = F_T(T_0, T_1)$.

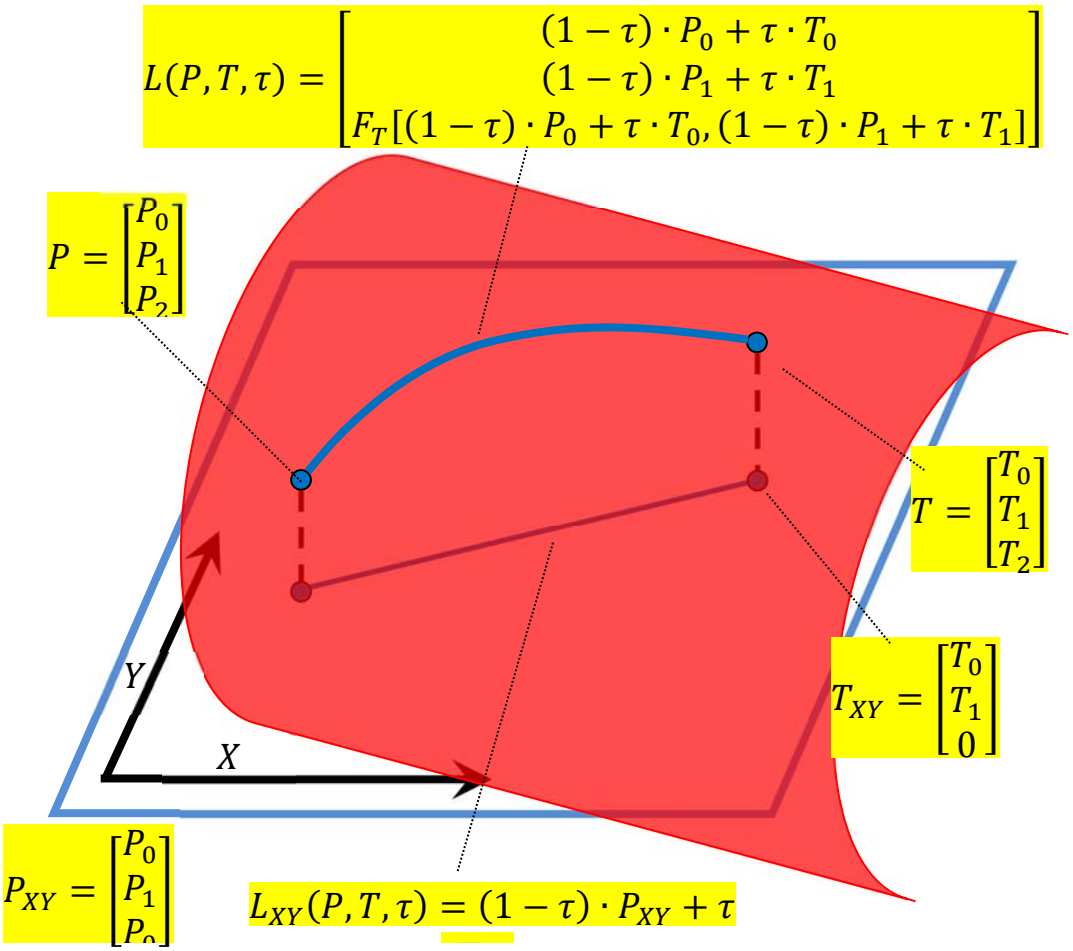


Рисунок 1.23. Проекция линии на поверхность

В коде программы это выглядит так.

$$\text{Line}_{PT}(P, T, \tau) := \begin{bmatrix} (1-\tau)\cdot P_0 + \tau \cdot T_0 \\ (1-\tau)\cdot P_1 + \tau \cdot T_1 \\ F_T[(1-\tau)\cdot P_0 + \tau \cdot T_0, (1-\tau)\cdot P_1 + \tau \cdot T_1] \end{bmatrix}$$

Зададим начальные положения преследователя. Отметим то, что траектория цели определена уже полностью, поэтому ввод начальных координат цели избыточен.

$$P_0 := \begin{pmatrix} 20 \\ 20 \\ F_T(20,20) \end{pmatrix}$$

Определим начальные положения проекций на плоскость XY преследователя и цели.

$$P_H := \begin{pmatrix} P0_0 \\ P0_1 \\ 0 \end{pmatrix} \qquad T_H := \begin{pmatrix} Xt_0 \\ Yt_0 \\ 0 \end{pmatrix}$$

Вывод на экран конца отрезка, соответственного кадровому положению горизонтальной проекции цели (Рисунок 1.24).

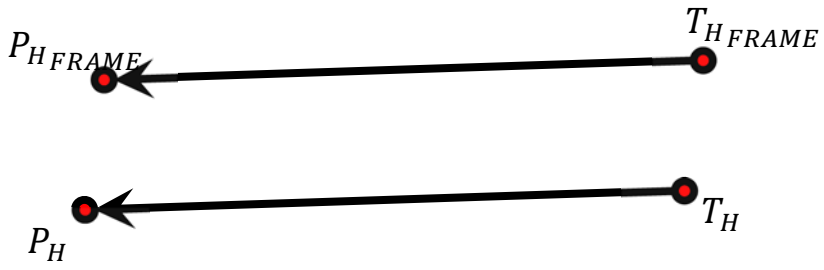


Рисунок 1.24. Определение кадрового положения проекции преследователя

На рисунке 1.24, как формируется кадровое положение горизонтальной проекции преследователя $P_{H_{FRAME}}$ от известного положения горизонтальной проекции цели $T_{H_{FRAME}}$. Начальные положения горизонтальных проекций преследователя и цели, $P_{H_{FRAME}}$ и $T_{H_{FRAME}}$ также определены:

$$P_{H_{FRAME}} = T_{H_{FRAME}} + (P_H - T_H).$$

В программном коде это выглядит так.

$$p_H := \begin{pmatrix} X_{T_{0,0}} \\ Y_{T_{0,0}} \\ 0 \end{pmatrix} + (P_H - T_H)$$

Линия, соединяющая проекции $P_{H_{FRAME}}$ и $T_{H_{FRAME}}$ выглядит так.

$$PT_H(\tau) := (1-\tau)\cdot p_H + \tau\cdot \begin{pmatrix} X_{T_{0,0}} \\ Y_{T_{0,0}} \\ 0 \end{pmatrix}$$

Вывод на экран.

$$PT_{H.p} := CreateSpace(PT_H, 0, 1, 50)$$

Проекция линии, соединяющей точки $P_{H_{FRAME}}$ и $T_{H_{FRAME}}$ на поверхность $Z = F_T(X, Y)$ с выводом на экран.

$$PT_L(\tau) := Line_{PT}(p_H, T_0, \tau) \qquad PT_{L.p} := CreateSpace(PT_L, 0, 1, 50)$$

Еще раз о цели.

$$TARGET_0 := \begin{pmatrix} X_{t_0} \\ Y_{t_0} \\ Z_{t_0} \end{pmatrix}$$

$$X_{TARGET_0} := (TARGET_0)_0 \quad Y_{TARGET_0} := (TARGET_0)_1 \quad Z_{TARGET_0} := (TARGET_0)_2$$

Еще раз о преследователе.

$$Pers_0 := P_0$$

$$X_{Pers_0} := P_{0_0} \quad Y_{Pers_0} := P_{0_1} \quad Z_{Pers_0} := P_{0_2}$$

Итерационная схема для траектории цели в промежутке времени $[0; T_{final}]$ с числом кадров N_{frame}.

$$i := 1..N_{frame}$$

$$TARGET_i := \begin{bmatrix} (TARGET_{i-1})_0 \\ (TARGET_{i-1})_1 \\ F_T[(TARGET_{i-1})_0, (TARGET_{i-1})_1] \end{bmatrix} + dTar(V_{Target}\cdot Time_i)\cdot V_{Target}\cdot \Delta T$$

$$X_{TARGET_i} := (TARGET_i)_0$$

$$Y_{TARGET_i} := (TARGET_i)_1$$

$$Z_{TARGET_i} := F_T(X_{TARGET_i}, Y_{TARGET_i})$$

Данные операторы отражают движение цели с постоянной скоростью вдоль своей траектории.

Затем рассчитываем точки, как показано на рисунке 1.24 и проецируем их на поверхность.

$$X_{Pers_i} := (TARGET_i)_0 + (P_H - T_H)_0 \quad Y_{Pers_i} := (TARGET_i)_1 + (P_H - T_H)_1 \quad Z_{Pers_i} := F_T(X_{Pers_i}, Y_{Pers_i})$$

Введем модуль скорости преследователя.

$$V_P := 11.8$$

Далее, сформируем функцию пересечения пространственной линии и сферы. Пространственная линия $Line_{PT}$ задается точками P и T, принадлежащими поверхности $Z = F_T(X, Y)$ (Рисунок 1.25).

$$f_{int}(pp, tt, cc, w) := \sqrt{\left[Line_{PT}\left[\begin{pmatrix} pp_0 \\ pp_1 \end{pmatrix}, \begin{pmatrix} tt_0 \\ tt_1 \end{pmatrix}, w\right] - \begin{pmatrix} cc_0 \\ cc_1 \\ cc_2 \end{pmatrix}\right]^T \cdot \left[Line_{PT}\left[\begin{pmatrix} pp_0 \\ pp_1 \end{pmatrix}, \begin{pmatrix} tt_0 \\ tt_1 \end{pmatrix}, w\right] - \begin{pmatrix} cc_0 \\ cc_1 \\ cc_2 \end{pmatrix}\right] - (V_P \cdot \Delta T)}$$

На рисунке 1.25 показано, что точки P, T и C принадлежат поверхности $Z = F_T(X, Y)$.

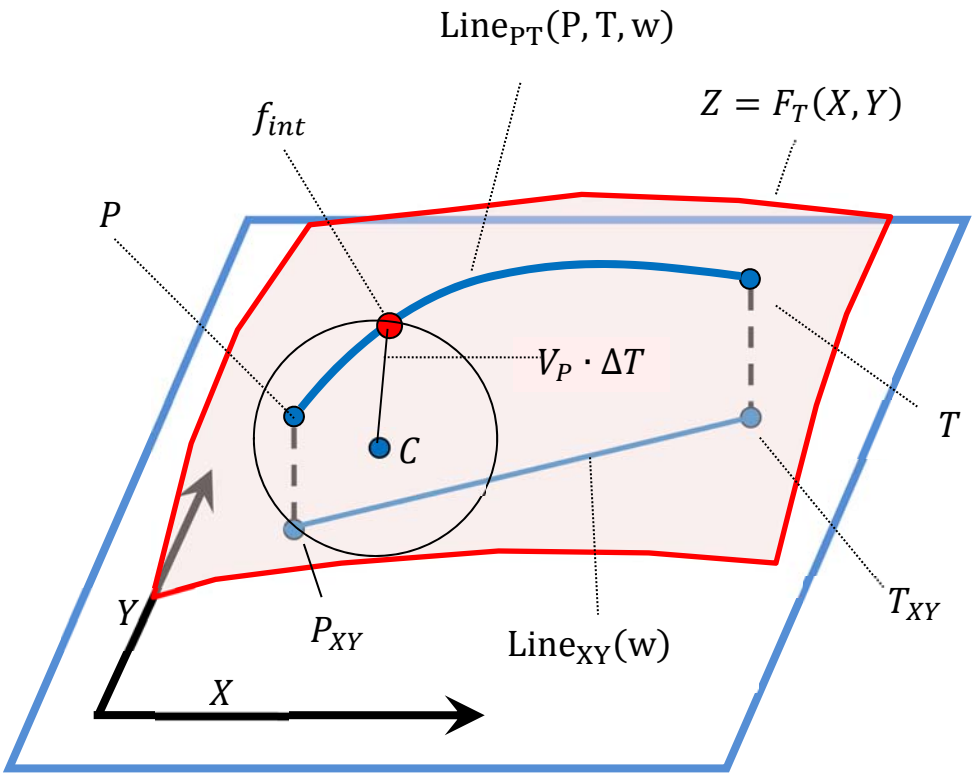


Рисунок 1.25. Точка пересечения сферы с линией на поверхности

Линия $Line_{PT}$ есть проекция прямой линии $Line_{XY}$, соединяющей точки P_{XY} и T_{XY}: $Line_{XY}(w) = (1 - w) \cdot P_{XY} + w \cdot T_{XY}$. Целью данной функции есть

нахождение найти точки f_{int}, являющейся точкой пересечения линии $Line_{PT}$ со сферой с центром в точке C и радиусом $V_P \cdot \Delta T$. Точек пересечения f_{int} сферы с линией $Line_{PT}$ всего две, поэтому необходимо наложить ограничение, чтобы отсечь ненужное решение. В нашем алгоритме, в конечном расчетном цикле, это ограничение связано с принадлежностью параметра w к отрезку $[w_{i-1}: 1]$. Где w_{i-1} – это значение параметра w на предыдущем шаге итераций.

$$
\begin{aligned}
QQ := \;& \begin{pmatrix} X_{PERS_0} \\ Y_{PERS_0} \\ Z_{PERS_0} \end{pmatrix} \leftarrow \begin{pmatrix} X_{Pers_0} \\ Y_{Pers_0} \\ Z_{Pers_0} \end{pmatrix} \\
& ans_0 \leftarrow 0 \\
& \text{for } i \in 1..N_{frame} \\
& \quad ans_i \leftarrow root\left[f_{int}\left[\begin{pmatrix} X_{Pers_i} \\ Y_{Pers_i} \end{pmatrix}, \begin{pmatrix} X_{TARGET_i} \\ Y_{TARGET_i} \end{pmatrix}, \begin{pmatrix} X_{PERS_{i-1}} \\ Y_{PERS_{i-1}} \\ Z_{PERS_{i-1}} \end{pmatrix}, w \right], w, ans_{i-1}, 1 \right] \\
& \quad \begin{pmatrix} X_{PERS_i} \\ Y_{PERS_i} \\ Z_{PERS_i} \end{pmatrix} \leftarrow Line_{PT}\left[\begin{pmatrix} X_{Pers_i} \\ Y_{Pers_i} \end{pmatrix}, \begin{pmatrix} X_{TARGET_i} \\ Y_{TARGET_i} \end{pmatrix}, ans_i \right] \\
& \begin{pmatrix} X_{PERS} \\ Y_{PERS} \\ Z_{PERS} \end{pmatrix}
\end{aligned}
$$

Конечный расчетный цикл последовательно вычисляет точки траектории движения преследователя на основе функции расчета точки пересечения сферы с радиусом равным шагу преследователя с прогнозируемой линией (аналог линии визирования).

$$
sp(\alpha, \varphi) := \begin{bmatrix} (V_P \cdot \Delta T) \cdot \cos(\alpha) \cdot \cos(\varphi) \\ (V_P \cdot \Delta T) \cdot \sin(\alpha) \cdot \cos(\varphi) \\ (V_P \cdot \Delta T) \cdot \sin(\varphi) \end{bmatrix} + \begin{bmatrix} (QQ_0)_{FRAME} \\ (QQ_1)_{FRAME} \\ (QQ_2)_{FRAME} \end{bmatrix}
$$

$$
Sp := CreateMesh\left(sp, 0, 2\pi, \frac{-\pi}{2}, \frac{\pi}{2}, 50, 25 \right)
$$

Данные операторы формируют и выводят на экран сферы, соответствующие шагу преследователя.

На рисунке 1.26 показаны результаты расчета точек траектории движения преследователя. Также рисунок 1.26 дополнен ссылкой на анимированное изображение [46].

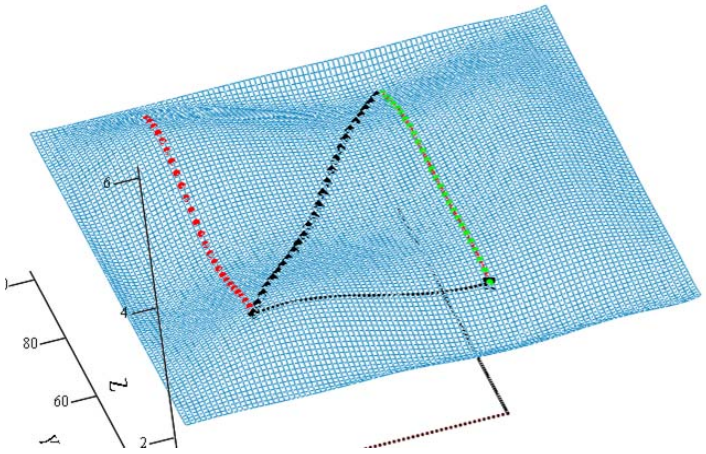


Рисунок 1.26. Расчет точек траектории преследователя

На рисунке 1.27 показаны итоговые результаты расчета траектории преследователя с визуализацией сфер на каждом шаге.

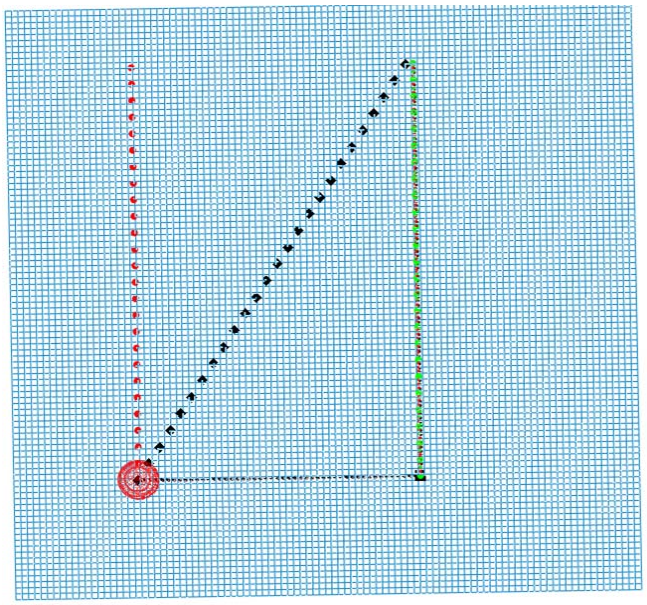


Рисунок. 1.27. Траектория преследователя с визуализацией сфер

Рисунок 1.27 также дополнен ссылкой на анимированное изображение [47].

2. МОДЕЛЬ ПРЕСЛЕДОВАНИЯ МЕТОДОМ ПОГОНИ

Задачу преследования на плоскости, в пространстве и на поверхности можно представить в виде численного решения систем обыкновенных дифференциальных уравнений первого порядка.

Допустим, решение системы уравнений (2.1) дает координаты траектории преследователя $P(t) = \begin{bmatrix} P_x(t) \\ P_y(t) \end{bmatrix}$ на плоскости.

$$\begin{aligned} \frac{dP_x(t)}{dt} &= \frac{P_x(t) - T_x(t)}{\sqrt{\left(P_x(t) - T_x(t)\right)^2 + \left(P_y(t) - T_y(t)\right)^2}} \cdot V_P \\ \frac{dP_y(t)}{dt} &= \frac{P_y(t) - T_y(t)}{\sqrt{\left(P_x(t) - T_x(t)\right)^2 + \left(P_y(t) - T_y(t)\right)^2}} \cdot V_P \end{aligned} \tag{2.1}$$

Точки траектории преследователя $P(t)$ зависят от положения цели $T(t)$ и от модуля скорости движения преследователя V_P. Начальное направление движения не задается, на неявно предполагается, что скорость движения преследователя всегда направлена на цель. Также предполагается, что траектория цели $T(t)$ является заранее известной.

$$\begin{aligned} \frac{dP_x(t)}{dt} &= \frac{P_x(t) - T_x(t)}{\sqrt{\left(P_x(t) - T_x(t)\right)^2 + \left(P_y(t) - T_y(t)\right)^2}} \cdot V_P \\ \frac{dP_y(t)}{dt} &= \frac{P_y(t) - T_y(t)}{\sqrt{\left(P_x(t) - T_x(t)\right)^2 + \left(P_y(t) - T_y(t)\right)^2}} \cdot V_P \\ \frac{dT_x(t)}{dt} &= -\frac{P_y(t) - T_y(t)}{\sqrt{\left(P_x(t) - T_x(t)\right)^2 + \left(P_y(t) - T_y(t)\right)^2}} \cdot V_T \\ \frac{dT_y(t)}{dt} &= -\frac{P_y(t) - T_y(t)}{\sqrt{\left(P_x(t) - T_x(t)\right)^2 + \left(P_y(t) - T_y(t)\right)^2}} \cdot V_T \end{aligned} \tag{2.2}$$

Решение системы (2.2) ($V_T -$ модуль скорости цели) приводит к движению вдоль прямой линии, соединяющей начальные положения преследователя и цели.

Система уравнений (3.3) дает решение для траектории преследователя в трехмерном пространстве.

$$\begin{aligned}
\frac{dP_x(t)}{dt} &= \frac{P_x(t) - T_x(t)}{\sqrt{\left(P_x(t) - T_x(t)\right)^2 + \left(P_y(t) - T_y(t)\right)^2 + \left(P_z(t) - T_z(t)\right)^2}} \cdot V_P \\
\frac{dP_y(t)}{dt} &= \frac{P_y(t) - T_y(t)}{\sqrt{\left(P_x(t) - T_x(t)\right)^2 + \left(P_y(t) - T_y(t)\right)^2 + \left(P_z(t) - T_z(t)\right)^2}} \cdot V_P \\
\frac{dP_z(t)}{dt} &= \frac{P_z(t) - T_z(t)}{\sqrt{\left(P_x(t) - T_x(t)\right)^2 + \left(P_y(t) - T_y(t)\right)^2 + \left(P_z(t) - T_z(t)\right)^2}} \cdot V_P
\end{aligned} \quad (2.3)$$

Рассмотрим случай, когда преследователь $P(t)$ и цель $T(t)$ движутся по некоторой поверхности $z = f(x, y)$.

$$\begin{gathered}
\frac{dP_x}{dt} \cdot \left(T_y - P_y\right) = \frac{dP_y}{dt} \cdot (T_x - P_x) \\
\left(\frac{dP_x}{dt}\right)^2 + \left(\frac{dP_y}{dt}\right)^2 + \left(\frac{dP_z}{dt}\right)^2 = (V_P)^2 \quad . \\
P_z = f\left(P_x, P_y\right)
\end{gathered} \quad (2.4)$$

Первое уравнение системы (2.4) говорит о том, что горизонтальная проекция вектора движения преследователя $P(t)$ направлена на горизонтальную проекцию вектора цели $T(t)$.

Второе уравнение системы (2.4) говорит о постоянной по модулю скорости преследователя.

Третье уравнение системы (2.4) говорит о том, что все точки траектории движения преследователя принадлежат поверхности $z = f(x, y)$.

Продифференцировав третье уравнение системы (2.4), перейдем к системе уравнений (2.5).

$$\frac{dP_x}{dt}\cdot\left(T_y-P_y\right)=\frac{dP_y}{dt}\cdot(T_x-P_x)$$
$$\left(\frac{dP_x}{dt}\right)^2+\left(\frac{dP_y}{dt}\right)^2+\left(\frac{dP_z}{dt}\right)^2=(V_P)^2 \quad . \tag{2.5}$$
$$\frac{dP_z}{dt}=\frac{\partial f}{\partial P_x}\cdot\frac{dP_x}{dt}+\frac{\partial f}{\partial P_y}\cdot\frac{dP_y}{dt}$$

Система уравнений (2.5) имеет решение относительно переменных:

$$\left\{\frac{dP_x}{dt},\frac{dP_y}{dt},\frac{dP_z}{dt}\right\}.$$

В системе дифференциальных уравнений (2.6) представлено это решение.

$$\frac{dP_x}{dt}=\frac{V_P\cdot(T_x-P_x)}{\sqrt{(T_x-P_x)^2+\left(T_y-P_y\right)^2+\left((T_x-P_x)\cdot\frac{\partial f}{\partial P_x}+\left(T_y-P_y\right)\cdot\frac{\partial f}{\partial P_y}\right)^2}}$$
$$\frac{dP_y}{dt}=\frac{V_P\cdot\left(T_y-P_y\right)}{\sqrt{(T_x-P_x)^2+\left(T_y-P_y\right)^2+\left((T_x-P_x)\cdot\frac{\partial f}{\partial P_x}+\left(T_y-P_y\right)\cdot\frac{\partial f}{\partial P_y}\right)^2}}. \tag{2.6}$$
$$\frac{dP_z}{dt}=\frac{\partial f}{\partial P_x}\cdot\frac{dP_x}{dt}+\frac{\partial f}{\partial P_y}\cdot\frac{dP_y}{dt}$$

В этой главе будут рассматриваться квазидискретные задачи преследования, основанные на методах погони.

2.1 МЕТОД ПОГОНИ НА ПЛОСКОСТИ

Рассмотрим такую квазидискретную модель метода погони на плоскости, когда вектор скорости преследователя P постоянно направлен на цель T (Рисунок 2.1).

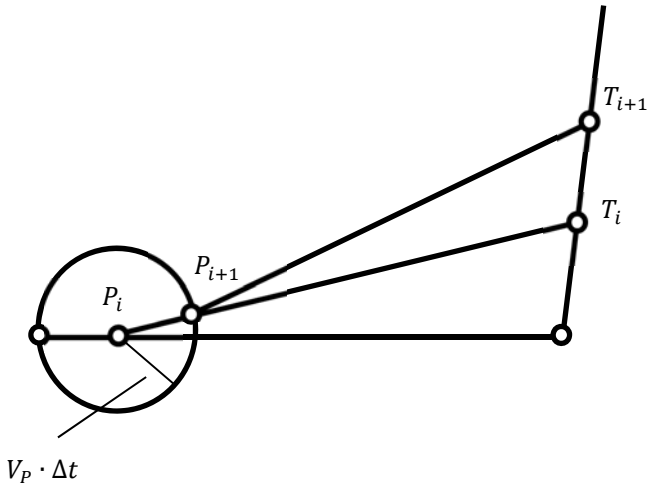


Рисунок 2.1. Скорость преследователя всегда направлена на цель

На рисунке 2.1 представлен итерационный процесс метода погони на плоскости. В момент времени t_i считаются известными координаты преследователя P_i, координаты цели T_i. Через временной промежуток Δt следующий шаг преследователя P_{i+1} рассчитывается по формуле:

$$P_{i+1} = P_i + V_P \cdot \Delta t \cdot (T_i - P_i).$$

Координаты следующего шага преследователя P_{i+1} есть точка пересечения линии визирования $[P_i T_i]$ в момент времени t_i с окружностью радиуса $V_P \cdot \Delta t$ с центром в точке P_i. V_P - это модуль скорости преследователя.

На рисунке 2.2 показан результат моделирования задачи преследования методом погони, согласно итерационной схеме, с нахождением точки пересечения окружности $(P_i, V_P \cdot \Delta t)$ и прямой $(P_i T_i)$, в системе компьютерной математики.

Такой подход к проектированию траектории не позволяет моделировать, когда скорость преследователя в момент начала преследования направлена не на цель.

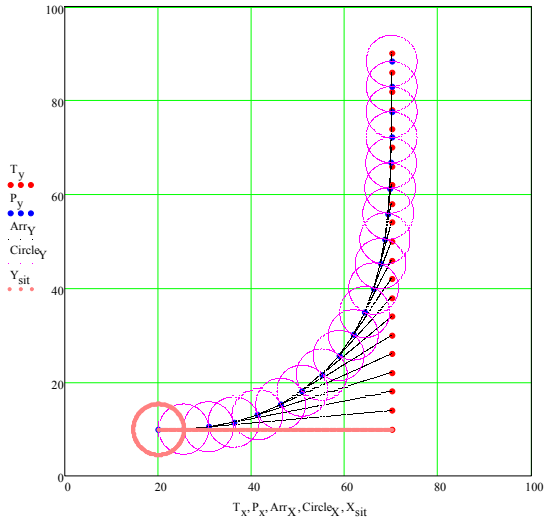


Рисунок 2.2. Результат моделирования траектории преследователя методом погони

2.2 КИНЕМАТИЧЕСКАЯ МОДЕЛЬ МЕТОДА ПОГОНИ НА ПЛОСКОСТИ

Рассмотрим следующую итерационную схему. Будем считать, что в момент времени t_i являются известными положение цели T_i, положение преследователя P_i и векторное уравнение $F_i(s)$ прогнозируемой на данный момент времени траектории движения преследователя (Рисунок 2.3).

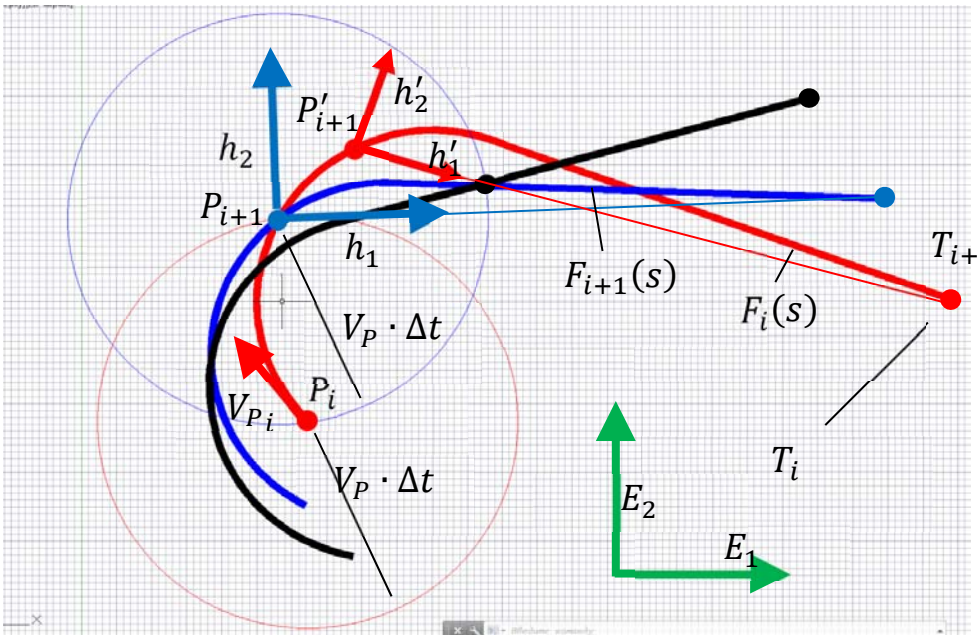


Рисунок 2.3. Моделирование траектории преследователя

В таком итерационном процессе нашей задачей является рассчитать координаты P_{i+1} следующего шага преследователя и выполнить аффинные преобразования векторной функции $F_i(s)$ для того, чтобы найти выражения для функции $F_{i+1}(s)$.

Чтобы найти координаты точки P_{i+1}, необходимо решить уравнение $|F_i(s_{i+1}) - F_i(s_i)| = V_P \cdot \Delta t$ относительно параметра s_{i+1}.

Когда мы разрабатывали модель итерационного процесса, мы задавали начальные координаты T и P, начальный вектор скорости движения преследователя V_P. Также определили векторную функцию $F(s)$ в момент начала преследования.

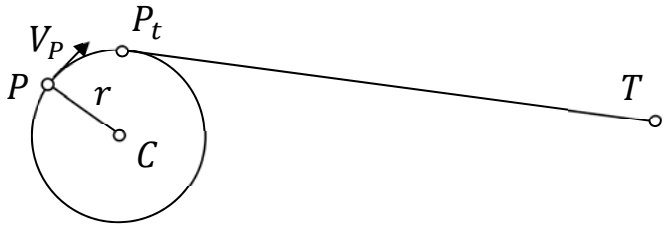


Рисунок 2.4. Моделирование траектории преследователя

На рисунке 2.4 это составная кривая из дуги $\widetilde{PP_t}$ и прямолинейного сегмента $[P_tT]$, где параметром служит длина дуги этой кривой.

На $i-$том итерационном шаге происходит следующее:

1. Строится окружность с центром в точке P_i радиуса $V_P \cdot \Delta t$. Точка пересечения этой окружности и траектории $F_i(s)$ будет точкой следующего шага преследователя P_{i+1}.
2. Потом находим точку пересечения P'_{i+1} параметрической векторной функции $F_i(s)$ с окружностью с центром в точке T_i и радиуса $|T_{i+1} - P_{i+1}|$.
3. Затем, формируем локальный базис (h'_1, h'_2) центром координат в точке P'_{i+1} и пересчитываем функцию $F_i(s)$. В базисе (h'_1, h'_2) она будет выглядеть так $F'_i(s)$. Компоненты базиса (h'_1, h'_2) такие:

$$h_1' = \frac{T_i - P_{i+1}'}{|T_i - P_{i+1}'|}$$
$$h_2' = \begin{bmatrix} -h_{1_y}' \\ h_{1_x}' \end{bmatrix}.$$

Вид функции $F_i'(s)$ будет таким:

$$F_i'(s) = \begin{bmatrix} (F_i(s) - P_{i+1}') \cdot h_1' \\ (F_i(s) - P_{i+1}') \cdot h_2' \end{bmatrix}.$$

4. Сформируем базис (h_1, h_2) с центром координат в точке P_{i+1}.

   Компоненты базиса (h_1, h_2) будут выглядеть так:

$$h_1 = \frac{T_{i+1} - P_{i+1}}{|T_{i+1} - P_{i+1}|}$$
$$h_2 = \begin{bmatrix} -h_{1_y} \\ h_{1_x} \end{bmatrix}.$$

   Отметим очень важный момент, что $|T_{i+1} - P_{i+1}| = |T_i - P_{i+1}'|$. Отсюда, мы можем утверждать, что локальное представление в локальном базисе (h_1, h_2) с центром в точке P_{i+1} кривой $F_{i+1}(s)$ будет совпадать с локальным представлением базиса (h_1', h_2') $F_{i+1}'(s) = F_i'(s)$.

5. Базис мировой системы координат (E_1, E_2) в базисе (h_1, h_2) выглядит так:

$$e_1 = \begin{bmatrix} E_1 \cdot h_1 \\ E_1 \cdot h_2 \end{bmatrix}$$
$$e_2 = \begin{bmatrix} E_2 \cdot h_1 \\ E_2 \cdot h_2 \end{bmatrix}.$$

Следовательно, уравнение линии $F_{i+1}(s)$ будет выражаться следующим образом:

$$F_{i+1}(s) = \begin{bmatrix} F_{i+1}'(s) \cdot e_1 \\ F_{i+1}'(s) \cdot e_2 \end{bmatrix} + P_{i+1}.$$

Итого на $i-$том шаге итерации мы имеем следующее: шаг T_{i+1} выбирается целью, шаг преследователя P_{i+1} рассчитывается как точка пересечения окружности $(P_i, V_P \cdot \Delta t)$ и ранее рассчитанной векторной

параметрической линии $F_i(s)$, на основе уже имеющихся данных рассчитывается новая прогнозируемая траектория преследователя $F_i(s)$.

На основе вышеизложенного материала была написана тестовая программа, которая на простом примере показывает, как можно подвести к методу погони, когда в момент начала преследования вектор скорости преследователя направлен не на цель.

Траектория преследователя при этом имеет ограничения на кривизну. Радиус кривизны траектории не может быть меньше некоторого порогового значения.

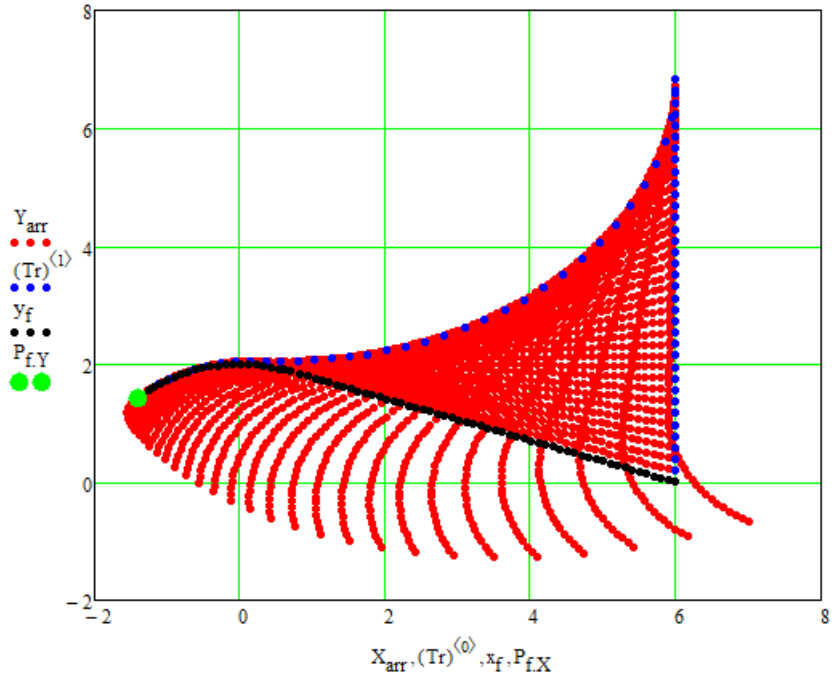


Рисунок 2.5. Кинематическая модель метода погони

Рисунок 2.5 как раз демонстрирует результаты работы программы. Рисунок 2.5 дополнен ссылкой на анимированное изображение [51], где можно в динамике наблюдать процесс преследования цели, совершающей движение по предопределенной траектории.

2.3 ГЕОМЕТРИЧЕСКАЯ МОДЕЛЬ МЕТОДА ПОГОНИ МЕТОДОМ КОРРЕКТИРОВКИ УГЛА НА ПЛОСКОСТИ

Рассмотрим метод погони уже не с точки зрения следования прогнозируемым траекториям, а с точки зрения, когда происходит корректировка угла между реальным направлением движения и «желаемым направлением». В данном параграфе за «желаемые направления» рассматривается линии визирования $[P_i T_i]$, $[P_{i+1} T_{i+1}]$ и т.д. (Рисунок 2.6).

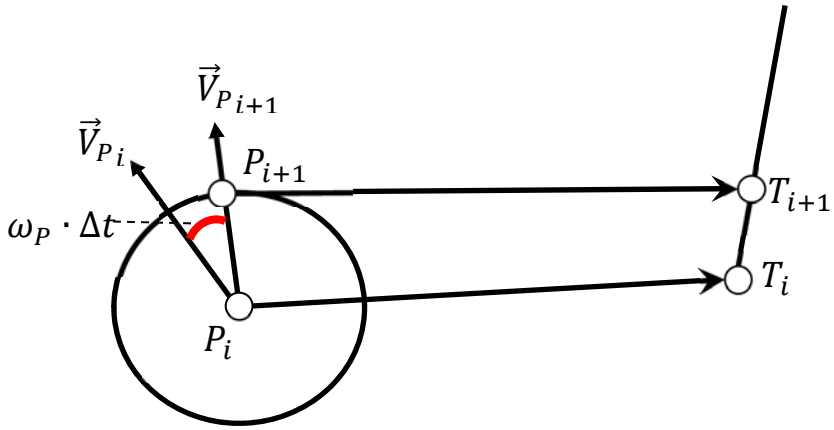


Рисунок 2.6. Корректировка угла направления движения

На рисунке 2.6 показано, что в момент времени t_i преследователь находится в точке P_i, а цель в точке T_i. Преследователь в данный момент имеет вектор скорости $\vec{V}_{P\,i}$

В момент времени t_i преследователь стремится к тому, чтобы направление его движения стремилось принять направление $[P_i T_i]$. Но преследователь не может мгновенно изменить направление своего движения. Поэтому мы ввели в математическую модель минимальный радиус кривизны траектории R_{min}. Если преследователь движется на плоскости с постоянной по модуль скоростью V_P, то через промежуток времени Δt вектор направления $\vec{V}_{P\,i}$ примет значение $\vec{V}_{P\,i+1}$, совершив поворот на угол $\omega \cdot \Delta t$, где $\omega = V_P / R_{min}$. После совершения поворота преследователь переместится вдоль вектора $\vec{V}_{P\,i+1}$ на расстояние $V_P \cdot \Delta t$ (Рисунок 2.6).

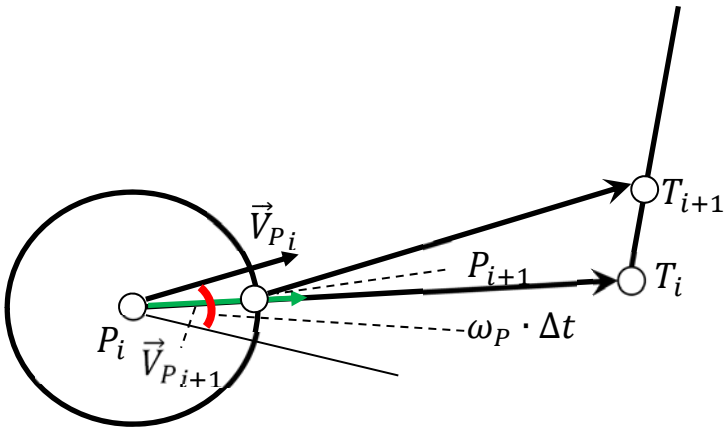


Рисунок 2.7. К корректировке угла направления

Если в какой-то момент времени t_i величина угла между предполагаемым направлением движения $[P_i T_i]$ и вектором скорости $\vec{V}_{P_i}$ меньше, чем значение $\omega \cdot \Delta t$, то следующий шаг преследователя откладывается вдоль линии $[P_i T_i]$ (Рисунок 2.7)

2.3.1 СОВМЕЩЕНИЕ НАПРАВЛЕНИЯ ДВИЖЕНИЯ С НАПРАВЛЕНИЕМ НА ТОЧКУ, ПРИНАДЛЕЖАЩЕЙ ОКРУЖНОСТИ АПОЛЛОНИЯ

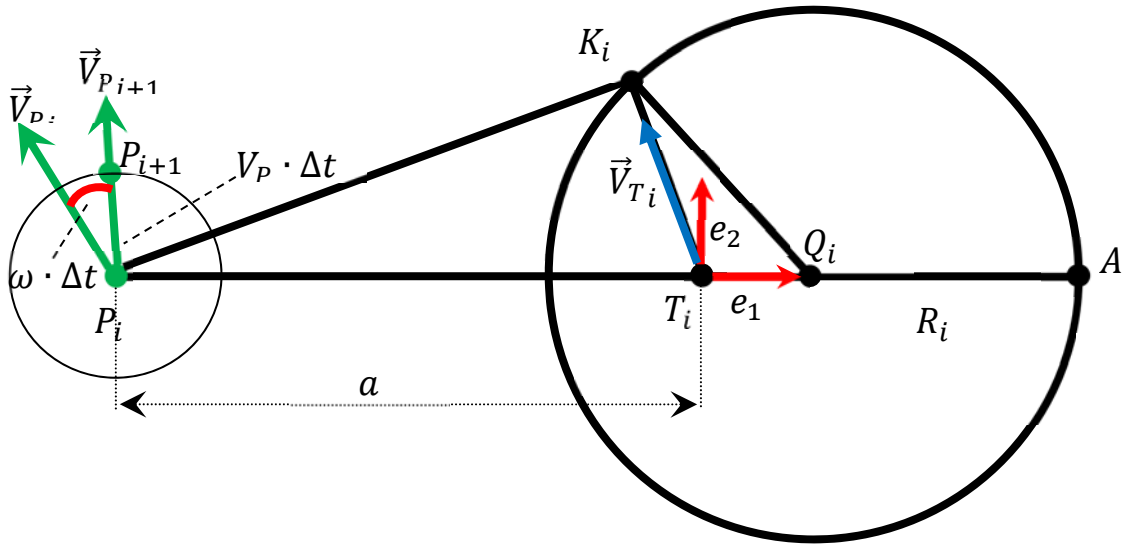


Рисунок 2.8. Корректировка направления при параллельном сближении

Рассмотрим случай, когда в момент t_i преследователь имеет координаты P_i и направление скорости $\vec{V}_{P_i}$ (Рисунок 2.8). Цель в этот момент времени имеет координаты T_i и направление скорости $\vec{V}_{T_i}$.

Координаты точек P_i и T_i, координаты векторов $\vec{V}_{P_i}$ и $\vec{V}_{T_i}$ считаются заданными в системе координат (e_1, e_2) с центром в точке T_i.

В системе координат (e_1, e_2) с центром в точке T_i по данным P_i, T_i и $\vec{V}_{T_i}$ строится окружность Аполлония радиуса R_i с центром в точке Q_i.

Радиус окружности Аполлония R_i вычисляется по формуле:

$$R_i = \frac{V_P \cdot V_T}{V_P^2 - V_T^2} \cdot |T_i - P_i|.$$

Центр окружности Аполлония Q_i вычисляется по формуле:

$$Q_i = T_i + \frac{V_T^2}{V_P^2 - V_T^2} \cdot (T_i - P_i).$$

Точка K_i удовлетворяет системе уравнений (2.1):

$$\begin{cases} (K_i - Q_i)^2 = R_i^2 \\ K_i = T_i + \vec{V}_{T_i} \cdot \tau \end{cases}. \qquad (2.1)$$

Нами была написана процедура, которая вычисляет координаты точки K_i в системе координат (e_1, e_2) с центром в точке T_i. Система уравнений (2.1) может быть преобразована квадратному уравнению относительно τ:

$$V_T^2 \cdot \tau^2 + 2 \cdot \left(\vec{V}_{T_i} \cdot P_i\right) \cdot \tau - \frac{V_T^2}{V_P^2 - V_T^2} \cdot P_i^2 = 0 \qquad (2.2)$$

Уравнение (2.2) приобретает такой вид в системе координат (e_1, e_2) с центром в точке T_i, где:

$$T_i = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Если принято как на рисунке 2.8 : $|P_i T_i| = a$, то для значения радиуса R_i имеем следующее:

$$R_i = \frac{V_P \cdot V_T}{V_P^2 - V_T^2} \cdot a.$$

Для центра Q_i окружности Аполлония имеем:

$$Q_i = \frac{V_T^2}{V_P^2 - V_T^2} \cdot a.$$

Точка P_i в системе координат (e_1, e_2) с центром в точке T_i выглядит так:

$$P_i = \begin{bmatrix} -a \\ 0 \end{bmatrix}.$$

Не забываем, что в уравнении (2.2) выражение $\left(\vec{V}_{T_i} \cdot P_i\right)$ имеет смысл скалярного произведения.

Вектор скорости $\vec{V}_{T_i}$ определяет положение точки K_i на окружности Аполлония, что в свою очередь задает предполагаемое направление движения, к которому будет стремиться вектор скорости $\vec{V}_{P_i}$ преследователя P_i (Рисунок 2.8).

2.4 ГЕОМЕТРИЧЕСКАЯ МОДЕЛЬ МЕТОДА ПОГОНИ НА ПОВЕРХНОСТИ

Рассмотрим задачу преследования на пересеченной местности, когда преследователь и цель передвигаются на поверхности в трехмерном пространстве с постоянными по модулю скоростями.

В данном разделе предлагаются вашему вниманию модели поведения, как преследователя, так и цели. В предлагаемой модели поведения цели будет производиться анализ пространственного положения преследователя, его скорости и в зависимости от этого будет «приниматься» решение целью о направлении его движения. Точно такие же рассуждения можно провести и в отношении модели поведения преследователя.

Отметим, что преследователь выстраивает свою траекторию, использую стратегию корректировки угла, стремясь совместить направление своего движения с направлением на цель.

Цель выстраивает свою траекторию, стремясь двигаться в противоположную от преследователя сторону.

2.4.1 РАСЧЕТ ТРАЕКТОРИИ ПРЕСЛЕДОВАТЕЛЯ НА ПОВЕРХНОСТИ

Пусть поверхность, по которой перемещаются преследователь и цель, задана в явном виде $z = f(x, y)$. Координаты положений преследователя P и цели T на поверхности $z = f(x, y)$ известны:

$$P = \begin{bmatrix} P_x \\ P_y \\ f(P_x, P_y) \end{bmatrix}, T = \begin{bmatrix} T_x \\ T_y \\ f(T_x, T_y) \end{bmatrix}.$$

Проекции положений преследователя P и цели T на плоскость (XY):

$$P_{XY} = \begin{bmatrix} P_x \\ P_y \\ 0 \end{bmatrix}, T_{XY} = \begin{bmatrix} T_x \\ T_y \\ 0 \end{bmatrix}. \qquad (2.3)$$

Через прямую линию, соединяющую проекции P_{XY_i} и T_{XY_i}, L_{XY_i} построим плоскость П, ортогональную плоскости (XY) (Рисунок 2.9).

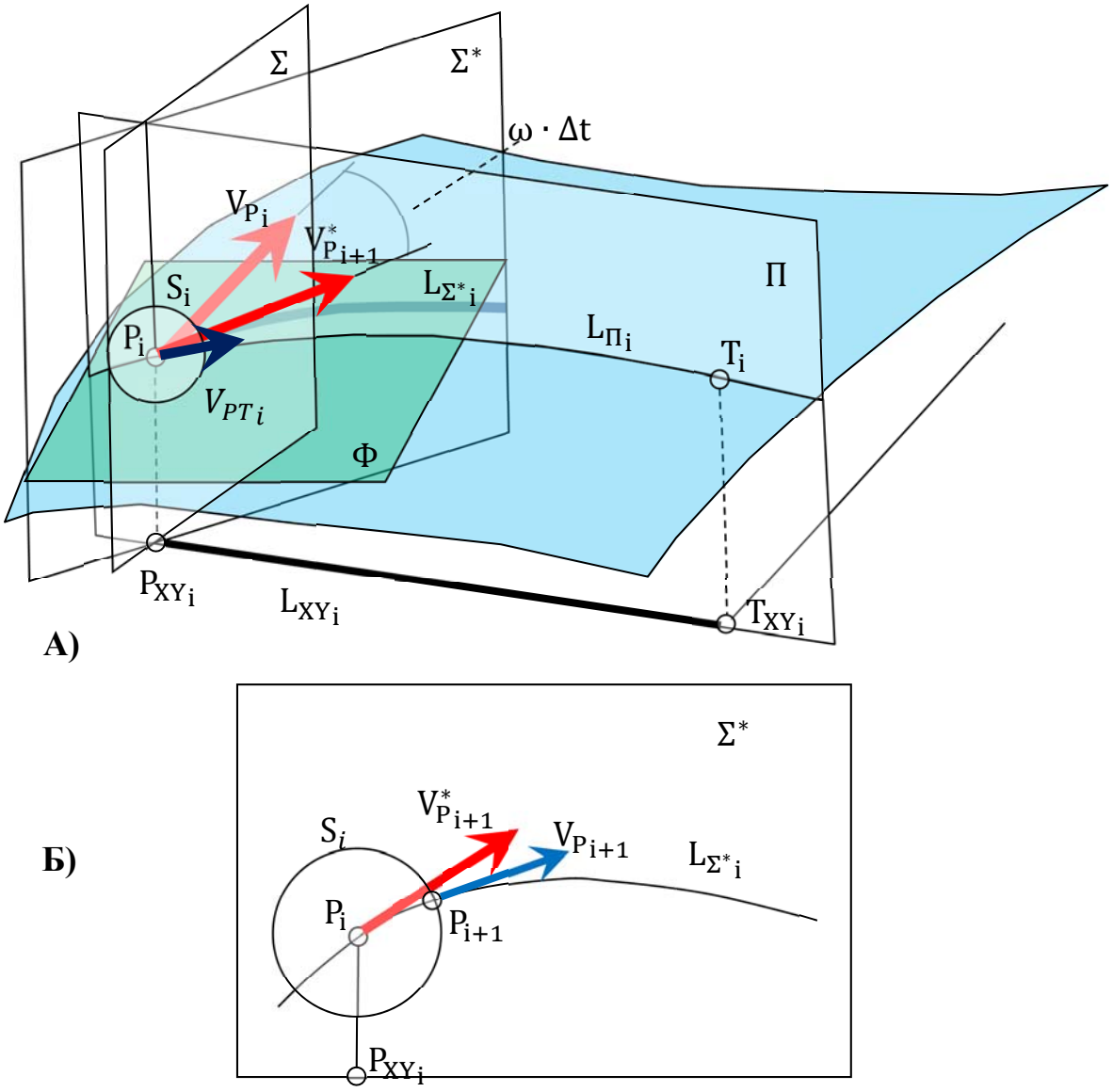


Рисунок 2.9. Построение траектории преследователя на поверхности

Преследователь в момент времени t_i имеет вектор скорости V_{P_i}. Через проецирующую прямую $(P_i P_{XY_i})$ и вектор V_{P_i} проведем проецирующую плоскость Σ.

То, что преследователь стремится к цели, в нашей модели выражается в том, что плоскость Σ стремится принять положение плоскости П.

На рисунке 2.9 А показано, что вектор V_{P_i} совершает поворот на угол $\omega \cdot \Delta t$ и принимает положение $V^*_{P_{i+1}}$. Вращение происходит в плоскости Ф, плоскость Ф – касательная плоскость к поверхности $z = f(x, y)$ в точке P_i.

Вектор $V^*_{P_{i+1}}$ и прямая $(P_i P_{XY_i})$ определяют плоскость Σ^*. Построим сферу S_i с центром в точке P_i и радиусом $|V_{P_i}| \cdot \Delta t$.

Построим линию пересечения $L_{\Sigma^*_i}$ плоскости Σ^* и поверхности $z = f(x, y)$ (Рисунок 2.9 Б). Найдем точку P_{i+1} пересечения линии $L_{\Sigma^*_i}$ и сферы S_i.

Точка P_{i+1} в нашей модели построения траектории преследователя методом корректировки направления движения и является следующим шагом итераций.

Параметрическое уравнение линии L_{XY_i}, учитывая уравнения (2.3), будет таким:

$$L_{XY_i}(\tau) = (1 - \tau) \cdot \begin{bmatrix} P_{x_i} \\ P_{y_i} \\ 0 \end{bmatrix} + \tau \cdot \begin{bmatrix} T_{x_i} \\ T_{y_i} \\ 0 \end{bmatrix}.$$

Параметрическое уравнение линии $L_{\Pi_i}(\tau)$ будет таким:

$$L_{\Pi_i}(\tau) = \begin{bmatrix} (1 - \tau) \cdot P_{x_i} + \tau \cdot T_{x_i} \\ (1 - \tau) \cdot P_{y_i} + \tau \cdot T_{y_i} \\ f\left((1 - \tau) \cdot P_{x_i} + \tau \cdot T_{x_i}, (1 - \tau) \cdot P_{y_i} + \tau \cdot T_{y_i}\right) \end{bmatrix}$$

Найдем касательный вектор V_{PT}:

$$V_{PT_i} = \frac{d}{d\tau} L_{\Pi_i}(\tau) = \begin{bmatrix} -P_{x_i} + T_{x_i} \\ -P_{y_i} + T_{y_i} \\ \left(-P_{x_i} + T_{x_i}\right) \cdot \dfrac{\partial f}{\partial x} + \left(-P_{y_i} + T_{y_i}\right) \cdot \dfrac{\partial f}{\partial y} \end{bmatrix}. \qquad (2.4)$$

В формуле (2.4) частные производные берутся в точке P_i (Рисунок 2.9).

На рисунке 2.9 касательная плоскость к поверхности $z = f(x, y)$ обозначена буквой Ф. Вектор скорости V_{P_i} и вектор V_{PT_i} принадлежат к плоскости Ф. Вектор $V_{P_{i+1}}$ получается вращением вектора V_{P_i} в плоскости Ф на угол $\omega \cdot \Delta t$ по направлению к вектору V_{PT_i}. Проецирующая прямая $\left(P_i P_{XY_i}\right)$ и вектор $V_{P_{i+1}}$ образуют проецирующую плоскость Σ^*. Плоскость Σ^* пересекается с поверхностью $z = f(x, y)$ по линии $L_{\Sigma^*_i}$.

Если компоненты вектора $V_{P_{i+1}}$ можно представить так:

$$V^*_{P_{i+1}} = \begin{bmatrix} Vx_{P_{i+1}} \\ Vy_{P_{i+1}} \\ Vz_{P_{i+1}} \end{bmatrix}.$$

То проекцию прямой линии из точки P_i в направлении вектора $V_{P_{i+1}}$ на поверхность $z = f(x, y)$ от параметра времени t можно представить в виде:

$$L_{\Sigma^*_i}(t) = \begin{bmatrix} P_{x_i} + t \cdot Vx_{P_{i+1}} \\ P_{y_i} + t \cdot Vy_{P_{i+1}} \\ f\left(P_{x_i} + t \cdot Vx_{P_{i+1}}, P_{y_i} + t \cdot Vy_{P_{i+1}}\right) \end{bmatrix}. \qquad (2.5)$$

Точка пересечения линии, заданной уравнением (2.5) со сферой S_i с центром в точке P_i радиуса $\left|V_{P_i}\right| \cdot \Delta t$ и будет следующим шагом в нашей модели построения траектории преследователя.

Теперь можно сказать, в момент времени t_{i+1} преследователь будет находиться в точке P_{i+1} и иметь скорость $V_{P_{i+1}}$, которую можно вычислить по формуле:

$$V_{P_{i+1}} = \frac{dL_{\Sigma^*_i}(t)}{dt} = \begin{bmatrix} Vx_{P_{i+1}} \\ Vy_{P_{i+1}} \\ Vx_{P_{i+1}} \cdot \dfrac{\partial f}{\partial x} + Vy_{P_{i+1}} \cdot \dfrac{\partial f}{\partial y} \end{bmatrix}.$$

Отметим то, что вектор частных производных

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

берется в точке P_{i+1}.

Точка P_{i+1} есть результат решения уравнения (2.6) относительно параметра t:

$$\left(P_i - L_{\Sigma^* i}(t)\right) = \left(V_{P\,i} \cdot \Delta t\right)^2, \qquad (2.6)$$

где Δt - это временной промежуток дискретизации нашего итерационного процесса.

Найденное значение t подставим в формулу (2.5) и получим значение P_{i+1}, как следующего шага траектории преследователя.

2.4.2 РАСЧЕТ ТРАЕКТОРИИ ЦЕЛИ НА ПОВЕРХНОСТИ

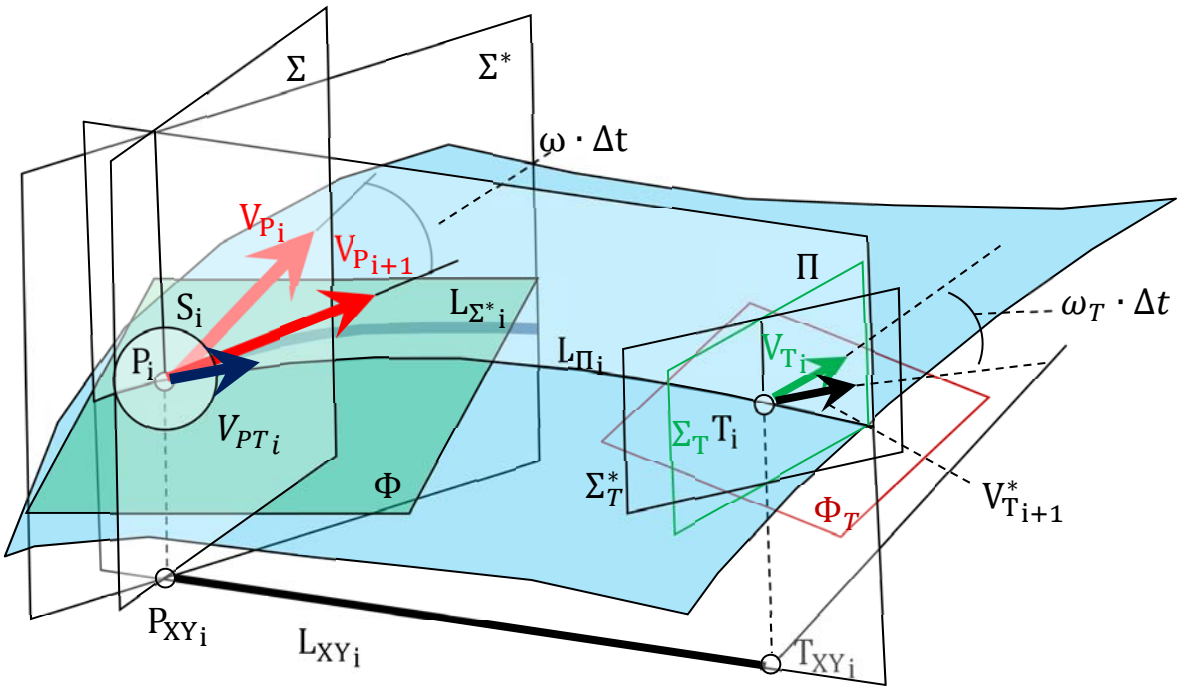


Рисунок 2.10. Построение траектории цели на поверхности

Рассмотрим рисунок 2.10. Цель в момент времени t_i находится в точке поверхности T_i с вектором скорости V_{T_i}. Для нахождения следующего итераций T_{i+1} необходимо сначала к поверхности $z = f(x, y)$ в точке T_i провести касательную плоскость Φ_T. Затем в плоскости Φ_T повернуть вектор V_{T_i} на угол $\omega_T \cdot \Delta t$ и получить вектор $V^*_{T\,i+1}$, .

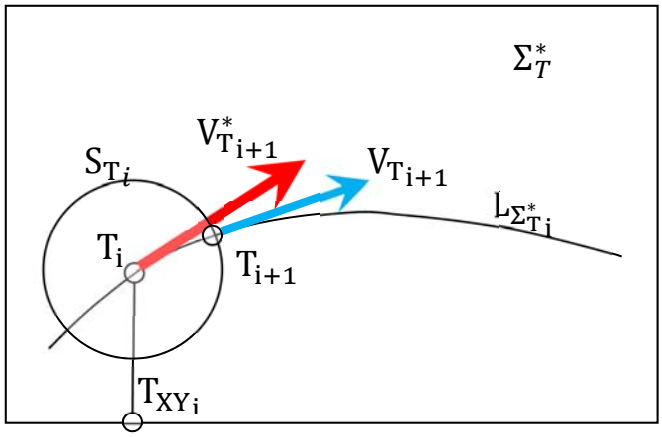


Рисунок 2.11 Построение следующего шага траектории цели

Рассмотрим сечение поверхности $z = f(x, y)$ плоскостью Σ^*_T, показанное на рисунке 2.11. Плоскость Σ^*_T образована проецирующей прямой $(T_i T_{XY\,i})$ и вектором $V_{T\,i+1}$.

Плоскость Σ^*_T пересекает поверхность $z = f(x, y)$ по линии $L_{\Sigma^*_{T_i}}$ (Рисунок 2.11). Как и в параграфе 2.4.1, мы можем представить вектор $V^*_{T\,i+1}$ в виде:

$$V^*_{T\,i+1} = \begin{bmatrix} Vx_{T_{i+1}} \\ Vy_{T_{i+1}} \\ Vz_{T_{i+1}} \end{bmatrix}.$$

Тогда линию пересечения поверхности $z = f(x, y)$ с плоскостью Σ^*_T можно представить в параметрическом виде:

$$L_{\Sigma^*_{T\,i}}(t) = \begin{bmatrix} T_{x_i} + t \cdot Vx_{T_{i+1}} \\ T_{y_i} + t \cdot Vy_{T_{i+1}} \\ f\left(T_{x_i} + t \cdot Vx_{T_{i+1}}, T_{y_i} + t \cdot Vy_{T_{i+1}}\right) \end{bmatrix}. \qquad (2.7)$$

Как и в предыдущем параграфе, построим сферу S_{T_i} с центром в точке T_i и радиусом $|V_{T_i}| \cdot \Delta t$. Найдем точку пересечения сферы S_{T_i} и линии $L_{\Sigma^*_{T\,i}}(\tau)$. Данная точка и будет в нашей модели следующим шагом T_{i+1} итерационного процесса.

Как и в предыдущем параграфе можно сказать, в момент времени t_{i+1} цель будет находиться в точке T_{i+1} и иметь скорость $V_{T_{i+1}}$, которую можно вычислить по формуле:

$$V_{T_{i+1}} = \frac{dL_{\Sigma^*{}_i}(t)}{dt} = \begin{bmatrix} Vx_{T_{i+1}} \\ Vy_{T_{i+1}} \\ Vx_{T_{i+1}} \cdot \dfrac{\partial f}{\partial x} + Vy_{T_{i+1}} \cdot \dfrac{\partial f}{\partial y} \end{bmatrix}.$$

Отметим то, что вектор частных производных

$$\begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

берется в точке T_{i+1}.

Точка T_{i+1} есть результат решения уравнения (8) относительно параметра t:

$$\left(T_i - L_{\Sigma^*_{T_i}}(t)\right) = \left(V_{T_i} \cdot \Delta t\right)^2, \qquad (2.8)$$

где Δt - это временной промежуток дискретизации нашего итерационного процесса.

Найденное значение t подставим в формулу (2.7) и получим значение T_{i+1}, как следующего шага траектории цели.

2.4.3 СОНАПРАВЛЕННОСТЬ СКОРОСТЕЙ, КАК СТРАТЕГИЯ ЦЕЛИ

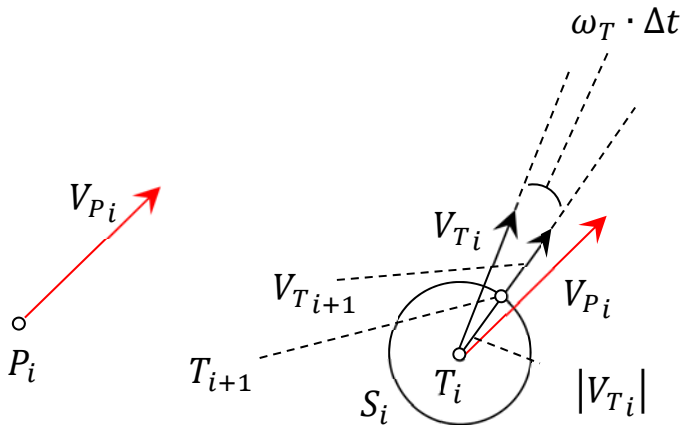


Рисунок 2.12. Стремление цели к сонаправленности скоростей

Сначала рассмотрим ситуацию на плоскости, когда преследователь и цель находятся в точках P_i и T_i в момент времени t_i (Рисунок 2.12). Они имеют направления движения V_{P_i} и V_{T_i}

Скорость преследователя V_{P_i} параллельно перенесем в точку текущего положения цели T_i. Вектор скорости цели V_{T_i} повернем вокруг точки T_i на угол $\omega_T \cdot \Delta t$, где ω_T – допустимая угловая скорость цели, Δt – период дискретизации нашего итерационного процесса.

В результате получим новый вектор скорости цели $V_{T_{i+1}}$, соответствующий моменту времени t_{i+1}.

В точке T_i построим окружность S_i с радиусом $|V_{T_i}| \cdot \Delta t$. Точка пересечения окружности S_i с прямой линией, направленной вдоль вектора $V_{T_{i+1}}$ и будет следующим шагом T_{i+1} итерационного процесса (Рисунок 2.12).

Теперь рассмотрим расчет траектории цели на поверхности, когда горизонтальная проекция скорости цели стремится стать параллельной горизонтальной проекции скорости преследователя.

На рисунке 2.13 показано, что преследователь P_i и цель T_i имеют векторы скоростей V_{P_i} и V_{T_i}, соответственно. Для преследователя P_i имеем ее горизонтальную проекцию P_{XY_i} и горизонтальную проекцию скорости $V_{P_{XY_i}}$. Указанную горизонтальную проекцию скорости преследователя $V_{P_{XY_i}}$ приложим к точке T_{XY_i}, которая является горизонтальной проекцией цели T_i.

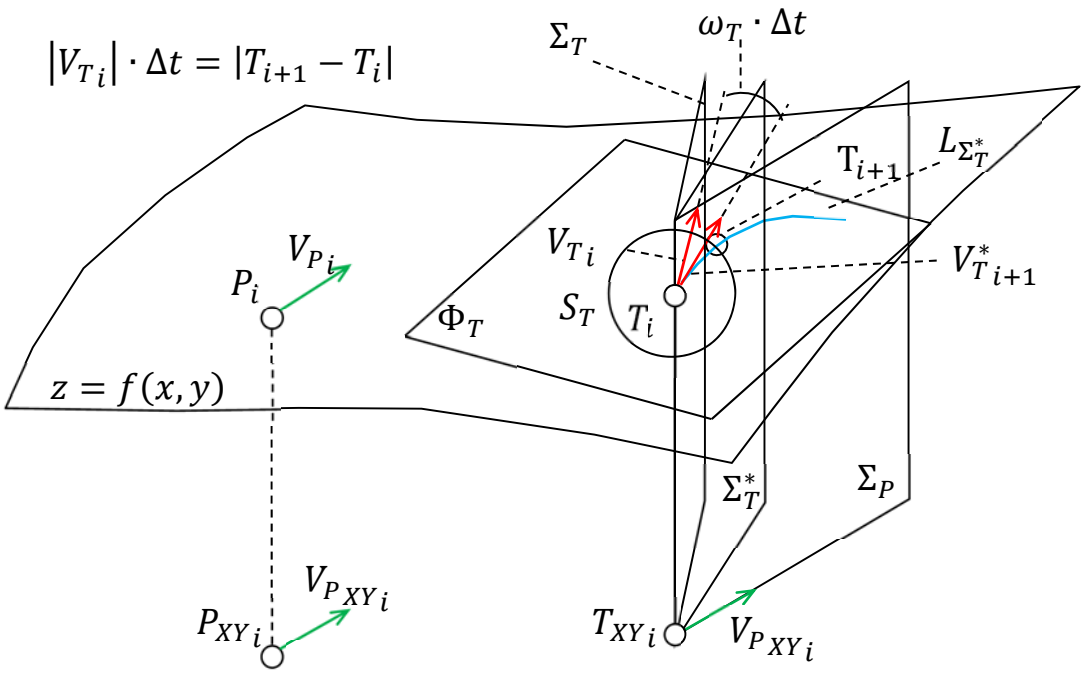


Рисунок 2.13. Корректировка направления цели на поверхности

В точке T_i построим касательную плоскость Φ_T к поверхности $z = f(x, y)$. Прямая $(T_i T_{XY_i})$ и приложенный к точке T_{XY_i} вектор $V_{P_{XY_i}}$ определяют плоскость Σ_P (Рисунок 2.13). Прямая $(T_i T_{XY_i})$ и вектор скорости цели V_{T_i} формируют плоскость Σ_T, которая будет вращаться вокруг прямой $(T_i T_{XY_i})$ по направлению к плоскости Σ_P. Угол вращения будет измеряться в плоскости Φ_T и будет равен $\omega_T \cdot \Delta t$, где ω_T – допустимая угловая скорость цели, Δt – период дискретизации нашего итерационного процесса.

После вращения в плоскости Φ_T вектора V_{T_i} вокруг точки T_i получаем вектор $V^*_{T\,i+1}$ (Рисунок 2.13). Чтобы получить вектор скорости $V_{T_{i+1}}$ на следующем этапе итерации в момент времени t_{i+1}, сделаем следующее.

Вектор $V^*_{T\,i+1}$ представим в виде:

$$V^*_{T\,i+1} = \begin{bmatrix} Vx_{T_{i+1}} \\ Vy_{T_{i+1}} \\ Vz_{T_{i+1}} \end{bmatrix}.$$

Вектор $V^*_{T\,i+1}$ и прямая $(T_i T_{XY_i})$ формируют плоскость Σ_T^*. Линию пересечения поверхности $z = f(x, y)$ с плоскостью Σ_T^* можно представить в параметрическом виде:

$$L_{\Sigma^*_{T_i}}(t) = \begin{bmatrix} T_{x_i} + t \cdot Vx_{T_{i+1}} \\ T_{y_i} + t \cdot Vy_{T_{i+1}} \\ f\left(T_{x_i} + t \cdot Vx_{T_{i+1}}, T_{y_i} + t \cdot Vy_{T_{i+1}}\right) \end{bmatrix}.$$

Как и в предыдущих параграфах, вектор скорости представим в виде:

$$V_{T\,i+1} = \frac{dL_{\Sigma^* i}(t)}{dt} = \begin{bmatrix} Vx_{T\,i+1} \\ Vy_{T\,i+1} \\ Vx_{T\,i+1} \cdot \dfrac{\partial f}{\partial x} + Vy_{T\,i+1} \cdot \dfrac{\partial f}{\partial y} \end{bmatrix}.$$

2.5 КОММЕНТАРИИ К ПРОГРАММЕ РАСЧЕТА ТРАЕКТОРИИ ПРЕСЛЕДОВАТЕЛЯ МЕТОДОМ ПОГОНИ НА ПЛОСКОСТИ

В данном листинге разбирается случай, когда скорость преследователя на плоскости всегда направлена на цель, как показано на рисунке 2.1. Цель в этой модели движется по предопределенной траектории.

$$\begin{aligned}
&\text{Point}_{\text{intersect}}(t,q,v,r) := \\
&\quad A \leftarrow \begin{pmatrix} v_0 & v_1 \end{pmatrix} \cdot \begin{pmatrix} v_0 & v_1 \end{pmatrix}^T \\
&\quad B \leftarrow 2 \cdot \begin{pmatrix} v_0 & v_1 \end{pmatrix} \cdot \left[\begin{pmatrix} t_0 \\ t_1 \end{pmatrix} - \begin{pmatrix} q_0 \\ q_1 \end{pmatrix} \right] \\
&\quad C \leftarrow (t-q)^T \cdot (t-q) - r^2 \\
&\quad D \leftarrow B^2 - 4 \cdot A \cdot C \\
&\quad \tau_1 \leftarrow \frac{-B + \sqrt{D}}{2 \cdot A} \\
&\quad \tau_2 \leftarrow \frac{-B - \sqrt{D}}{2 \cdot A} \\
&\quad k_1 \leftarrow t + v \cdot \tau_1 \\
&\quad k_2 \leftarrow t + v \cdot \tau_2 \\
&\quad \text{augment}(k_1, k_2)
\end{aligned}$$

Приведенная функция находит корни квадратного уравнения, используемого при поиске точек пересечения окружности и произвольной прямой.

Характеристики прямой и окружности показаны на рисунке 2.14.

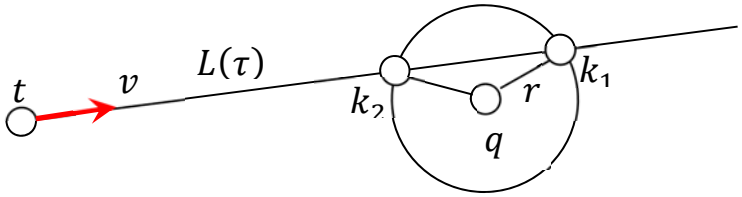


Рисунок 2.14. Точки пересечения прямой и окружности

На рисунке показано, что ищутся точки пересечения прямой $L(k)$, заданной точкой t и вектором p, и окружности с центром в точке q и радиуса r. Точку прямой $L(\tau)$ представим в параметрическом виде:

$$L(\tau) = t + \tau \cdot v.$$

Если точка прямой принадлежит окружности, то она должна удовлетворять уравнению:

$$(v \cdot v) \cdot \tau^2 + 2\big(v \cdot (t - q)\big) \cdot \tau + (t - q) \cdot (t - q) - r^2 = 0.$$

Решение такого уравнения в численном виде относительно параметра τ производит указанная выше функция.

Входными параметрами являются опорная точка прямой t, центр окружности q, направляющий вектор прямой v и радиус окружности r.

Выходными параметрами функции являются два значения точек пересечения прямой с окружностью, k_1 и k_2.

$$\mathrm{Arr_Q}(\mathrm{Target}, P, r) := \left| \begin{array}{l} p_0 \leftarrow P \\ mm \leftarrow \mathrm{rows}(\mathrm{Target}) - 1 \\ \text{for } i \in 0..mm - 1 \\ \quad p_{i+1} \leftarrow \mathrm{Point_{intersect}}\left(\mathrm{Target_i}, p_i, \dfrac{p_i - \mathrm{Target_i}}{\left|p_i - \mathrm{Target_i}\right|}, r\right)^{\langle 1 \rangle} \\ p \end{array} \right.$$

В приведенной функции входными данными являются массив точек траектории цели $\{Target_i\}$, начальная точка P_0 преследователя и радиус окружности r.

В этой функции итеративно рассчитываются координаты точек преследователя, когда в вызываемую в цикле функцию поиска точек пересечения прямой и окружности, подаются соответственная точка цели $Target_i$, в качестве опорной точки t прямой. Также в эту функцию подается соответственная точка положения преследователя P_i, в качестве центра окружности q. В качестве направляющего вектора v прямой, подается единичный вектор, направленный от преследователя к цели:

$$\frac{P_i - Target_i}{|P_i - Target_i|}.$$

Также подается радиус r.

Этих двух функций вполне достаточно, чтобы рассчитывать траекторию движения преследователя, если цель движется по предопределенной траектории.

$$\text{Arr}(LL) := \left| \begin{array}{l} mm \leftarrow \text{cols}(LL) - 1 \\ L \leftarrow LL^{\langle 0 \rangle} \\ \text{for } i \in 1..mm \\ \quad L \leftarrow \text{stack}\left(L, LL^{\langle i \rangle}\right) \\ L \end{array} \right.$$

Данная процедура преобразует матрицу в вектор столбец.

$N := 200$

Все линии, которые присутствуют в нашей программе, разбиты на 200 равных интервалов

$m := 20$

Количество точек траекторий, как цели, так и преследователя равно 21

$$P := \begin{pmatrix} 20 \\ 10 \end{pmatrix}$$

Координаты начального положения преследователя

$$T := \begin{pmatrix} 70 \\ 10 \end{pmatrix}$$

Координаты начального положения цели

$V_P := 14.9$

Модуль скорости движения преследователя

$V_T := 11$

Модуль скорости движения цели

$T_0 := T$

$$T_1 := \begin{pmatrix} 70 \\ 90 \end{pmatrix}$$

Данные операторы необходимы для того, чтобы определить временной промежуток для квазидискретной задачи преследования методом погони

$$\Delta T := \frac{|T_1 - T_0|}{m \cdot V_T} = 0.364$$

$Tar_0 := T_0$

$i := 1..m$

$$Tar_i := Tar_{i-1} + V_T \cdot \frac{T_1 - T_0}{|T_1 - T_0|} \cdot \Delta T$$

Ввод точек предопределенной траектории цели

$$Per_2 := Arr_Q\left(Tar, P, V_P \cdot \Delta T\right)$$

$Per := Per_2$

Определение точек траектории преследователя

$i := 0..m$

$j := 0..N$

Двойной цикл для визуализации всех линий нашей программы

$$L_{x_{i,j}} := \left(1 - \frac{j}{N}\right) \cdot P_{x_i} + \frac{j}{N} \cdot T_{x_i}$$

$$L_{y_{i,j}} := \left(1 - \frac{j}{N}\right) \cdot P_{y_i} + \frac{j}{N} \cdot T_{y_i}$$

Определение однопараметрического множества линий, соединяющих соответствующие точки положения преследователя и цели

$$L_{fx_j} := \left(1 - \frac{j}{N}\right) \cdot P_{x_{FRAME}} + \frac{j}{N} \cdot T_{x_{FRAME}}$$

$$L_{fy_j} := \left(1 - \frac{j}{N}\right) \cdot P_{y_{FRAME}} + \frac{j}{N} \cdot T_{y_{FRAME}}$$

Линия, соединяющая преследователя и цель, в момент времени соответствующему кадру

$$C_{x_{i,j}} := V_P \cdot \Delta T \cdot \cos\left(\frac{j}{N} \cdot 2 \cdot \pi\right) + P_{x_i}$$

$$C_{y_{i,j}} := V_P \cdot \Delta T \cdot \sin\left(\frac{j}{N} \cdot 2 \cdot \pi\right) + P_{y_i}$$

Определение однопараметрического множества окружностей с центрами в точках положения траектории преследователя

$$C_{fx_j} := V_P \cdot \Delta T \cdot \cos\left(\frac{j}{N} \cdot 2 \cdot \pi\right) + P_{x_{FRAME}}$$

$$C_{fy_j} := V_P \cdot \Delta T \cdot \sin\left(\frac{j}{N} \cdot 2 \cdot \pi\right) + P_{y_{FRAME}}$$

Окружность, построенная в точке положения преследователя, в момент времени соответствующему кадру

$X_{sit} := \text{stack}\left(L_{fx}, C_{fx}\right)$

Технический прием системы программирования. Объединение линии,

$Y_{sit} := stack(L_{fy}, C_{fy})$

соответствующей кадру, и соответствующей окружности

$Arr_X := Arr(L_x^T)$

$Arr_Y := Arr(L_y^T)$

Покоординатное объединение однопараметрической сети линий, соединяющих положения преследователя и цели, в вектор-столбцы

$Circle_X := Arr(C_x^T)$

$Circle_Y := Arr(C_y^T)$

Покоординатное объединение однопараметрической сети окружностей в точках положения преследователя в вектор-столбцы

Итоговый результат работы представлен на рисунке 2.15. Рисунок 2.15 дополнен ссылкой на анимированное изображение [50].

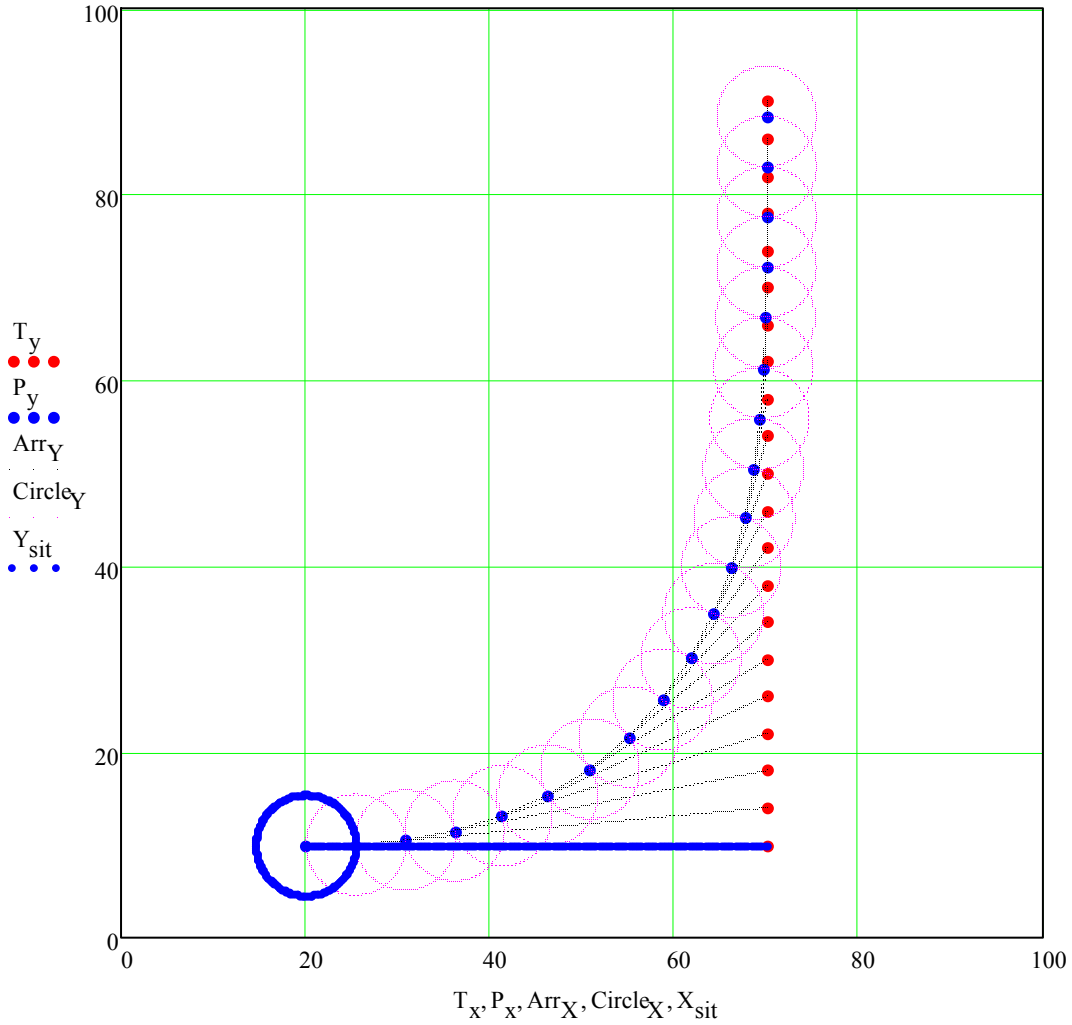


Рисунок 2.15. Визуализация метода погони, когда скорость преследователя всегда направлена на цель

2.6 КОММЕНТАРИИ К ПРОГРАММЕ МЕТОДА ПОГОНИ НА ПЛОСКОСТИ С ОГРАНИЧЕНИЯМИ НА КРИВИЗНУ

В программе листинга, которой изложен в этом параграфе, мы реализовали алгоритм следования прогнозируемым траекториям.

Преследователь, траекторию которого в программе мы строим, придерживается стратегии погони. Причем, в начальный момент времени скорость преследователя не направлена на цель.

В этом случае, если преследователь обладает определенной инертностью, то совершенно ясно, что он не сможет мгновенно изменять направление движения. Тогда преследователь будет стремиться выйти на ту траекторию, которой он следовал бы, если его скорость была всегда направлена на цель.

Мы опирались на математическую модель, изложенную в параграфе 2.2.

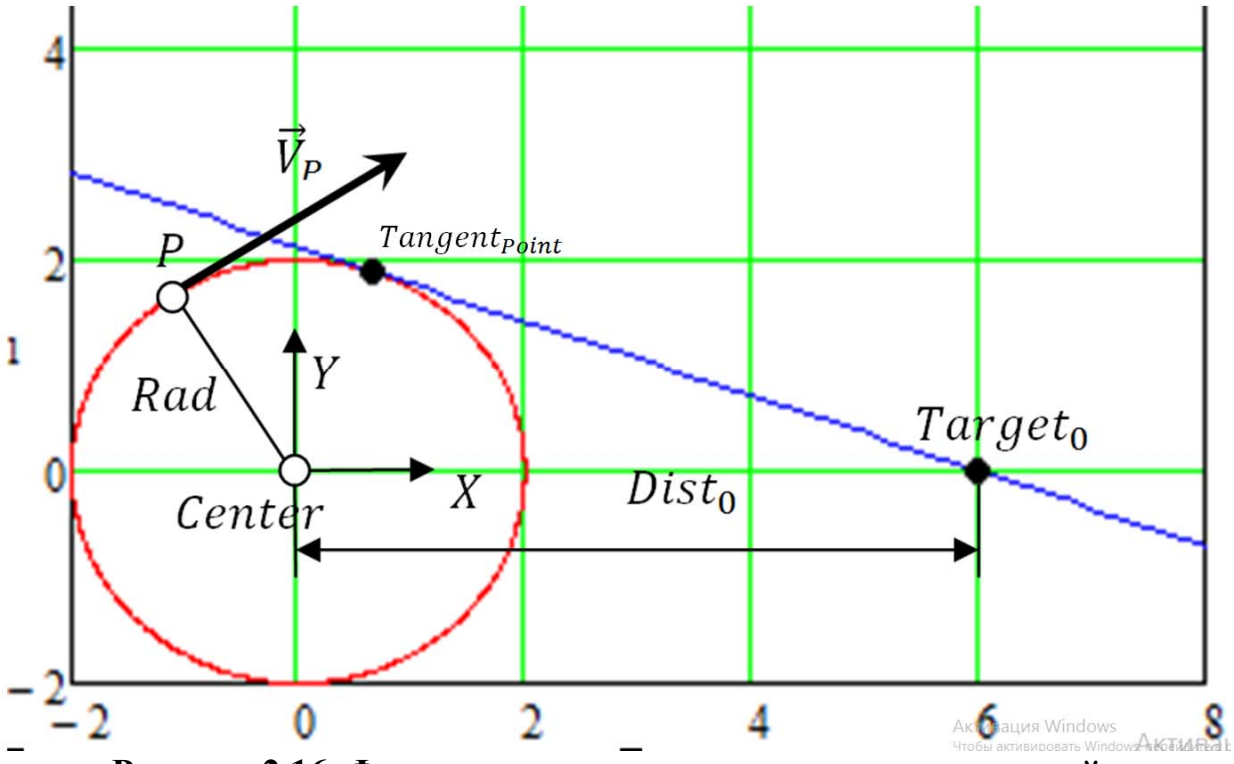


Рисунок 2.16. Формирование прогнозируемых траекторий

Пусть преследователь находится в точке P, имеет вектор скорости V_P, а цель находится в точке $Target_0$. Из точки P построим вектор перпендикулярный вектору V_P, по модулю равный минимальному радиусу кривизны траектории преследователя Rad:

$$Rad \cdot \frac{\begin{bmatrix} V_{P_y} \\ -V_{P_x} \end{bmatrix}}{\sqrt{{V_{P_x}}^2 + {V_{P_x}}^2}}.$$

Центр данной окружности буде находится в точке:

$$Center = P + Rad \cdot \frac{\begin{bmatrix} V_{P_y} \\ -V_{P_x} \end{bmatrix}}{\sqrt{{V_{P_x}}^2 + {V_{P_x}}^2}}.$$

Введем локальную систему координат. Центр системы координат находится в точке $Center$, ось абсцисс направлена вдоль линии, соединяющей точки $Target_0$ и $Center$, ось ординат – перпендикулярная ей линия (Рисунок 2.16).

$$Rad := 2 \quad Dist_0 := 6$$

Введем параметр $Dist_0$, равный модулю $|Center - Target_0|$. В локальной системе координат цель имеет координаты.

$$Target_0 := \begin{pmatrix} Dist_0 \\ 0 \end{pmatrix}$$

Для моделирования составной кривой, состоящей из сегмента дуги и сегмента прямой, касательной к окружности (Рисунок 2.16), нам необходимо рассчитать точку касания$Tangent_{Point}$.

$$Tangent_{Point} := \begin{pmatrix} \frac{Rad^2}{Dist_0} \\ \frac{Rad}{Dist_0} \cdot \sqrt{Dist_0^2 - Rad^2} \end{pmatrix} = \begin{pmatrix} 0.667 \\ 1.886 \end{pmatrix}$$

В локальной системе координат, связанной с точкой $Center$, выражение для точки $Tangent_{Point}$ будет именно таким.

$$Circle(R, t) := R \begin{pmatrix} \cos(-t) \\ \sin(-t) \end{pmatrix} \quad Circle_0(t) := Circle(Rad, t)$$

Параметризация окружности радиуса Rad. Для нас необходимо, чтобы при возрастании параметра t, вращение окружности шло по часовой стрелке.

$$Line(Target_0, Tangent_{Point}, t) := (1 - t) \cdot Tangent_{Point} + t \cdot Target_0$$

$$Line_0(t) := Line(Target_0, Tangent_{Point}, t)$$

Построение касательной линии.

$$Arc_{Segment}(K, \alpha_s) := \left| \begin{array}{l} n \leftarrow 20 \\ \alpha_f \leftarrow 2\pi - acos\left(\frac{K_0}{|K|}\right) \\ \text{for } i \in 0..n \\ \quad \left| \begin{array}{l} x_i \leftarrow Circle_0\left[\left(1 - \frac{i}{n}\right) \cdot \alpha_s + \frac{i}{n} \cdot \alpha_f\right]_0 \\ y_i \leftarrow Circle_0\left[\left(1 - \frac{i}{n}\right) \cdot \alpha_s + \frac{i}{n} \cdot \alpha_f\right]_1 \end{array} \right. \\ P \leftarrow augment(x, y) \\ P \end{array} \right.$$

$$\mathrm{Arc}_{\mathrm{Segment.0}} := \mathrm{Arc}_{\mathrm{Segment}}\left(\mathrm{Tangent}_{\mathrm{Point}}, \frac{5}{4}\pi\right)$$

Вычисление сегмента дуги от точки от точки P до точки $Tangent_{Point}$. В нашей тестовой программе точка P соответствует параметру на окружности $(5/4) \cdot \pi$ в локальной системе координат.

Сегмент дуги $\overset{\frown}{P, Tangent_{Point}}$ равномерно разбит на двадцать подсегментов.

$$\mathrm{Line}_{\mathrm{Segment}} := \left|\begin{array}{l} n \leftarrow 20 \\ \text{for } i \in 0..20 \\ \quad \left|\begin{array}{l} x_i \leftarrow \mathrm{Line}_0\left(\frac{i}{n}\right)_0 \\ y_i \leftarrow \mathrm{Line}_0\left(\frac{i}{n}\right)_1 \end{array}\right. \\ P \leftarrow \mathrm{augment}(x, y) \\ P \end{array}\right.$$

$$\mathrm{Line}_{\mathrm{Segment.0}} := \mathrm{Line}_{\mathrm{Segment}}$$

Данные операторы задают отрезок $[Tangent_{Point}, Target_0]$, равномерно разбитый на двадцать сегментов.

$$\mathrm{Trajectory}_{\mathrm{Point.0}} := \mathrm{reverse}\left(\mathrm{stack}\left(\mathrm{Arc}_{\mathrm{Segment.0}}, \mathrm{submatrix}\left(\mathrm{Line}_{\mathrm{Segment.0}}, 1, \mathrm{rows}\left(\mathrm{Line}_{\mathrm{Segment.0}}\right) - 1, 0, 1\right)\right)\right)$$

Объединение массивов точек окружности и прямой в один массив. Нумерация начинается с точки $Target_0$.

$$i_{fp} := 0..\mathrm{rows}\left(\mathrm{Trajectory}_{\mathrm{Point.0}}\right) - 1$$

$$\mathrm{Formal}_{\mathrm{Parameter}_{i_{fp}}} := i_{fp}$$

Для того, чтобы в нашей модели можно было использовать реальное время при движении по траекториям, мы все траектории, используемые в наших моделях, приводили к параметризации от своей длины дуги.

Мы решили также привести к параметризации от своей длины дуги составную траекторию, которую в дальнейшем будем как эталонную кривую при назначении траекторий, которых хотим придерживаться.

Создадим непрерывный формальный параметр f_p, который будет пробегать через опорные точки численно равные ее номеру в упорядоченном массиве.

$$\mathrm{Coef}_x := \mathrm{cspline}\left(\mathrm{Formal}_{\mathrm{Parameter}}, \mathrm{Trajectory}_{\mathrm{Point.0}}^{\langle 0 \rangle}\right)$$

$$\mathrm{Coef}_y := \mathrm{cspline}\left(\mathrm{Formal}_{\mathrm{Parameter}}, \mathrm{Trajectory}_{\mathrm{Point.0}}^{\langle 1 \rangle}\right)$$

По координатам подсчитаем коэффициенты кубических сплайнов, где параметром аргументов будет массив индексов опорных точек, а параметром значений служит объединенный массив составной кривой.

$$f_{x.formal}(f_p) := \mathrm{interp}\left(\mathrm{Coef}_x, \mathrm{Formal}_{\mathrm{Parameter}}, \mathrm{Trajectory}_{\mathrm{Point.0}}^{\langle 0 \rangle}, f_p\right)$$

$$f_{y.formal}(f_p) := interp\left(Coef_y, Formal_{Parameter}, Trajectory_{Point.0}^{\langle 1\rangle}, f_p\right)$$

На основе полученных коэффициентов кубических сплайнов произведем интерполяцию от непрерывного формального параметра f_p.

$$Predator_{formal}(f_p) := \begin{pmatrix} f_{x.formal}(f_p) \\ f_{y.formal}(f_p) \end{pmatrix}$$

На основе полученных координатных функций от формального параметра f_p составим векторную функцию.

$$\begin{aligned}
Diff_{Point} := \;& hT \leftarrow Formal_{Parameter_1} - Formal_{Parameter_0} \\
& D \leftarrow (0 \;\; 0) \\
& \text{for } i \in 0..rows(Formal_{Parameter}) - 1 \\
& \quad \text{if } i = 0 \\
& \quad\quad Dx \leftarrow \frac{Predator_{formal}(Formal_{Parameter_{i+1}})_0 - Predator_{formal}(Formal_{Parameter_i})_0}{hT} \\
& \quad\quad Dy \leftarrow \frac{Predator_{formal}(Formal_{Parameter_{i+1}})_1 - Predator_{formal}(Formal_{Parameter_i})_1}{hT} \\
& \quad \text{if } i = rows(Formal_{Parameter}) - 1 \\
& \quad\quad Dx \leftarrow \frac{Predator_{formal}(Formal_{Parameter_i})_0 - Predator_{formal}(Formal_{Parameter_{i-1}})_0}{hT} \\
& \quad\quad Dy \leftarrow \frac{Predator_{formal}(Formal_{Parameter_i})_1 - Predator_{formal}(Formal_{Parameter_{i-1}})_1}{hT} \\
& \quad \text{if } (i \neq 0) \wedge (i \neq rows(Formal_{Parameter}) - 1) \\
& \quad\quad Dx \leftarrow \frac{Predator_{formal}(Formal_{Parameter_{i+1}})_0 - Predator_{formal}(Formal_{Parameter_{i-1}})_0}{2hT} \\
& \quad\quad Dy \leftarrow \frac{Predator_{formal}(Formal_{Parameter_{i+1}})_1 - Predator_{formal}(Formal_{Parameter_{i-1}})_1}{2hT} \\
& \quad D \leftarrow stack[D, (Dx \;\; Dy)] \\
& D \leftarrow submatrix(D, 1, rows(Formal_{Parameter}), 0, 1)
\end{aligned}$$

Произведем численное дифференцирование по полученным опорным точкам в зависимости от формального параметра.

$$CoeffDiff_X := cspline\left(Formal_{Parameter}, Diff_{Point}^{\langle 0\rangle}\right)$$

$$CoeffDiff_Y := cspline\left(Formal_{Parameter}, Diff_{Point}^{\langle 1\rangle}\right)$$

$$d_X(f_p) := interp\left(CoeffDiff_X, Formal_{Parameter}, Diff_{Point}^{\langle 0\rangle}, f_p\right)$$

$$d_Y(f_p) := interp\left(CoeffDiff_Y, Formal_{Parameter}, Diff_{Point}^{\langle 1\rangle}, f_p\right)$$

Далее, на основе полученных значений производных в узловых точках, произведем кубическую сплайн-интерполяцию от непрерывного параметра f_p, чтобы иметь непрерывные координатные функции первых производных нашей составной кривой, которую назовем базовой.

Исходя из уравнения для полного дифференциала плоской кривой, имеем:

$$ds^2 = dx^2 + dy^2.$$

При параметризации от формального параметра f_p это примет следующий вид:

$$\frac{ds}{df_p}^2 = \frac{dx}{df_p}^2 + \frac{dy}{df_p}^2.$$

Полученное выражение приведем к виду:

$$\frac{df_p}{ds} = \frac{1}{\sqrt{\frac{dx}{df_p}^2 + \frac{dy}{df_p}^2}}.$$

Или в нашей тестовой программе Якобиан для решения задачи Коши имеет следующий вид.

$$Jc(Arc_{length}, f_p) := \frac{1}{\sqrt{(d_X(f_p))^2 + (d_Y(f_p))^2}}$$

Для того, чтобы Якобиан передать во встроенные решатели задачи Коши, нам необходимо установить примерный диапазон. Это мы сделаем при помощи значения накопленных хорд.

$$ChordLength := \left| \begin{array}{l} Temp \leftarrow 0 \\ \text{for } i \in 0..rows(Trajectory_{Point.0}) - 2 \\ \quad Temp \leftarrow Temp + \sqrt{\left[\left(Trajectory_{Point.0}^{\langle 0 \rangle}\right)_{i+1} - \left(Trajectory_{Point.0}^{\langle 0 \rangle}\right)_i\right]^2} \\ \quad \quad + \left[\left(Trajectory_{Point.0}^{\langle 1 \rangle}\right)_{i+1} - \left(Trajectory_{Point.0}^{\langle 1 \rangle}\right)_i\right]^2 \\ Temp \end{array} \right.$$

$$ChordLength = 7.907$$

Начальное значение формального параметра f_p от длины дуги $Start_0$ равно нулю, а диапазон $[0, ChordLength]$ разделен на 200 равных отрезков.

$$Start_0 := 0 \quad NumberOfKnot := 200$$

Далее, во встроенный решатель Рунге - Кутты передается сформированный Якобиан с начальным значением $Start_0$, диапазоном $[0, ChordLength]$ и числом отрезков $NumberOfKnox$.

$$Formal_{Arc} := rkfixed\left(Start_0, 0, ChordLength, NumberOfKnot, Jc\right)$$

Результатом этой процедуры является матрица $Formal_{Arc}$, состоящая из двух столбцов. Нулевой столбец это диапазон значений длины дуги $[0, ChordLength]$, а первый столбец – это соответствующие значения формального параметра f_p.

$$Coeff_{Formal.Arc} := cspline\left(Formal_{Arc}^{\langle 0 \rangle}, Formal_{Arc}^{\langle 1 \rangle}\right)$$

$$Form(dist) := interp\left(Coeff_{Formal.Arc}, Formal_{Arc}^{\langle 0 \rangle}, Formal_{Arc}^{\langle 1 \rangle}, dist\right)$$

Для получения непрерывной функции формального параметра от длины дуги произведем кубическую сплайн-интерполяцию на основе матрицы $Formal_{Arc}$.

В программном коде это функция $Form(dist)$.

$$Predator_x(dist) := Predator_{formal}(Form(dist))_0$$

$$Predator_y(dist) := Predator_{formal}(Form(dist))_1$$

Сформируем координатные функции базовой линии от длины дуги.

$$Predator(dist) := \begin{pmatrix} Predator_x(dist) \\ Predator_y(dist) \end{pmatrix}$$

Сформируем векторную функцию базовой линии от своей длины дуги.

На рисунке 2.17 приведен вывод базовой линии в параметризации от своей длины дуги.

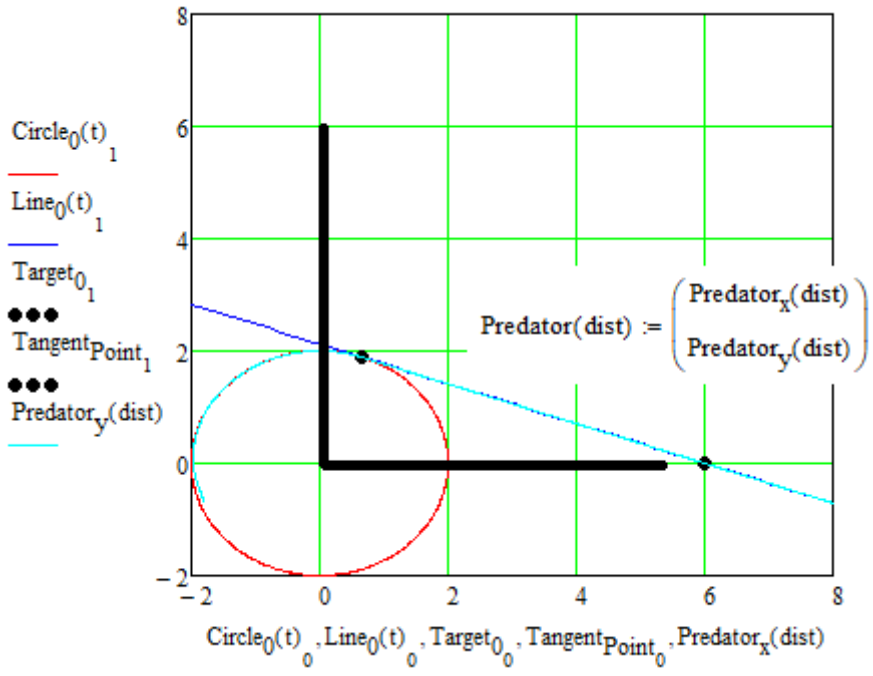


Рисунок 2.17. Базовая линия в параметризации от своей длины дуги

Значение накопленной длины хорд является приблизительным, поэтому мы рассчитали конечное значение диапазона длины дуги с заданной степенью точности.

$$\mathrm{Dist}_{start} := 0$$

$$\mathrm{Dist}_{final} := \mathrm{root}\left(\mathrm{Predator}_x(q) - \mathrm{Circle}_0\left(\frac{5}{4}\pi\right)_0, q, 0, 10\right) = 7.906$$

$$\mathrm{Predator}\left(\mathrm{Dist}_{start}\right) = \begin{pmatrix} 6 \\ 0 \end{pmatrix}$$

$$\mathrm{Predator}\left(\mathrm{Dist}_{final}\right) = \begin{pmatrix} -1.414 \\ 1.414 \end{pmatrix}$$

Далее, базовую линию представим в виде массива точек, в полученном диапазоне. Это необходимо для визуализации базовой линии.

$$N_{point} := 75$$

$$i := 0..N_{point}$$

$$\mathrm{dist}_i := \mathrm{Dist}_{start} + \frac{i}{N_{point}} \cdot \left(\mathrm{Dist}_{final} - \mathrm{Dist}_{start}\right)$$

$$\mathrm{Tr}_{point.X_i} := \mathrm{Predator}\left(\mathrm{dist}_i\right)_0$$

$$\mathrm{Tr}_{point.Y_i} := \mathrm{Predator}\left(\mathrm{dist}_i\right)_1$$

У нас есть векторная функция базовой линии $Predator(dist)$ с параметризацией от длины дуги, нашей задачей

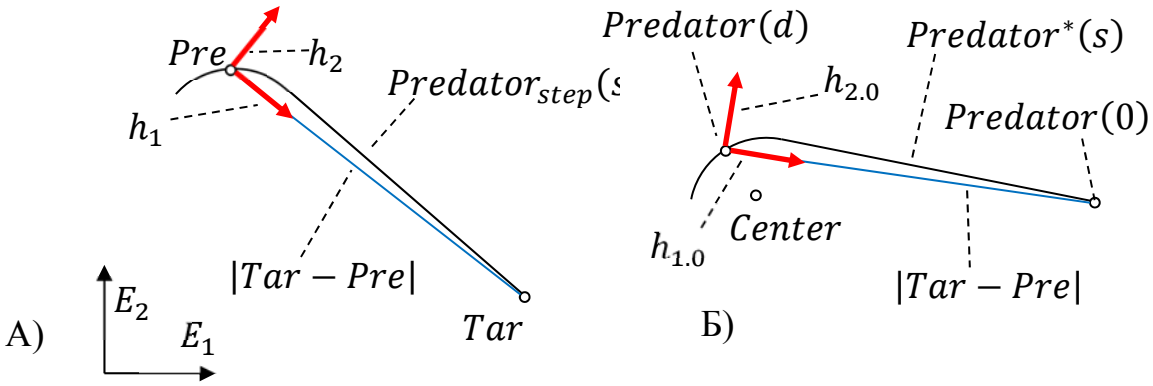


Рисунок 2.18. Аффинные преобразования с базовой линией

Для нас является необходимым решить такую задачу: через две точки плоскости Tar и Pre (Рисунок 2.18 А) построить линию, конгруэнтную базовой, таким образом, чтобы конец, соответствующий цели, был закреплен в точке Tar, а сама линия проходила через точку Pre.

Для решения этой задачи на базовой линии в ее локальной системе координат (Рисунок 2.18 Б), построим аналог точки Pre. Это будет точка пересечения базовой линии $Predator(s)$ с окружностью с центром в точке

$Predator(0)$. Точкой пересечения является точка $Predator(d)$, где – это соответствующий ей параметр длины дуги.

В локальном базисе, связанном с базовой линией, создадим новый базис $(h_{1.0}, Predator(d), h_{2.0})$, где:

$$h_{1.0} = \frac{Predator(0) - Predator(d)}{|Predator(0) - Predator(d)|}$$

$$h_{2.0} = \begin{bmatrix} -h_{1.0_y} \\ h_{1.0_x} \end{bmatrix}.$$

Пересчитаем базовую линию в базисе $(h_{1.0}, Predator(d), h_{2.0})$:

$$Predator^*(s) = \begin{bmatrix} \big(Predator(s) - Predator(d)\big) \cdot h_{1.0} \\ \big(Predator(s) - Predator(d)\big) \cdot h_{2.0} \end{bmatrix}.$$

Указанная ниже функция выполняет описанные действия. Входными параметрами являются: параметр длины дуги s, вектор положения цели Tar и вектор положения преследователя Pre.

$$\begin{aligned}
&\text{Predator}_L(s, Tar, Pre) := \left| \begin{array}{l} \text{Tar}_0(Tar, Pre) \leftarrow \text{Predator}(0) \\ \text{Pre}_0(Tar, Pre) \leftarrow \text{Predator}\Big(\text{root}\big(|\text{Predator}(d) - \text{Predator}(0)| \\ \qquad\qquad - |Tar - Pre|, d, 0, \text{Dist}_{final}\big)\Big) \end{array} \right. \\
&\qquad \left| \begin{array}{l} \text{h}_{1.0}(Tar, Pre) \leftarrow \dfrac{\text{Tar}_0(Tar, Pre) - \text{Pre}_0(Tar, Pre)}{\left|\text{Tar}_0(Tar, Pre) - \text{Pre}_0(Tar, Pre)\right|} \\ \text{h}_{2.0}(Tar, Pre) \leftarrow \begin{pmatrix} -\text{h}_{1.0}(Tar, Pre)_1 \\ \text{h}_{1.0}(Tar, Pre)_0 \end{pmatrix} \\ \begin{bmatrix} \big(\text{Predator}(s) - \text{Pre}_0(Tar, Pre)\big)^T \cdot \text{h}_{1.0}(Tar, Pre) \\ \big(\text{Predator}(s) - \text{Pre}_0(Tar, Pre)\big)^T \cdot \text{h}_{2.0}(Tar, Pre) \end{bmatrix} \end{array} \right.
\end{aligned}$$

Выходными параметрами являются координатные функции базисной линии, пересчитанные в базисе $(h_{1.0}, Predator(d), h_{2.0})$.

Представление базовой линии в базисе $(h_{1.0}, Predator(d), h_{2.0})$ совпадает с представлением базовой линии в базисе (h_1, Pre, h_2) (Рисунок 2.18 А), где:

$$h_1 = \frac{Tar - Pre}{|Tar - Pre|}$$

$$h_2 = \begin{bmatrix} -h_{1_y} \\ h_{1_x} \end{bmatrix}.$$

В базисе (h_1, Pre, h_2) базисные векторы мировой системы координат E_1 и E_2 выглядят так:

$$e_1 = \begin{bmatrix} E_1 \cdot h_1 \\ E_1 \cdot h_2 \end{bmatrix}, \text{где } E_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, E_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$
$$e_2 = \begin{bmatrix} E_2 \cdot h_1 \\ E_2 \cdot h_2 \end{bmatrix}$$

В итоге мы получим выражение для базовой линии, над которой совершили аффинное преобразование, выраженной в мировой системе координат:

$$Predator_{step}(s) = \begin{bmatrix} Predator^*(s) \cdot e_1 \\ Predator^*(s) \cdot e_2 \end{bmatrix} + Pre.$$

Следующая функция рассчитывает координаты базовой линии, перенесенной в базис (h_1, Pre, h_2), способом, который мы описали.

Входными параметрами являются: параметр длины дуги s, вектор положения цели Tar и вектор положения преследователя Pre.

$$\begin{aligned}
&\text{Predator}_F(s, Tar, Pre) := \\
&\quad h_1(Tar, Pre) \leftarrow \frac{Tar - Pre}{|Tar - Pre|} \\
&\quad E_1 \leftarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\
&\quad E_2 \leftarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\
&\quad h_2(Tar, Pre) \leftarrow \begin{pmatrix} -h_1(Tar, Pre)_1 \\ h_1(Tar, Pre)_0 \end{pmatrix} \\
&\quad e_1(Tar, Pre) \leftarrow \begin{pmatrix} E_1^{T} \cdot h_1(Tar, Pre) \\ E_1^{T} \cdot h_2(Tar, Pre) \end{pmatrix} \\
&\quad e_2(Tar, Pre) \leftarrow \begin{pmatrix} E_2^{T} \cdot h_1(Tar, Pre) \\ E_2^{T} \cdot h_2(Tar, Pre) \end{pmatrix} \\
&\quad \text{Predator}_{step.X}(s, Tar, Pre) \leftarrow \text{Predator}_L(s, Tar, Pre)^{T} \cdot e_1(Tar, Pre) + Pre_0 \\
&\quad \text{Predator}_{step.Y}(s, Tar, Pre) \leftarrow \text{Predator}_L(s, Tar, Pre)^{T} \cdot e_2(Tar, Pre) + Pre_1 \\
&\quad \begin{pmatrix} \text{Predator}_{step.X}(s, Tar, Pre) \\ \text{Predator}_{step.Y}(s, Tar, Pre) \end{pmatrix}
\end{aligned}$$

Выходными параметрами являются координатные функции базисной линии, которую перенесли в базис (h_1, Pre, h_2) и выразили в мировой системе координат.

$$\text{Point}_{circle}(Radius, Length, Tar, Pre) :=$$

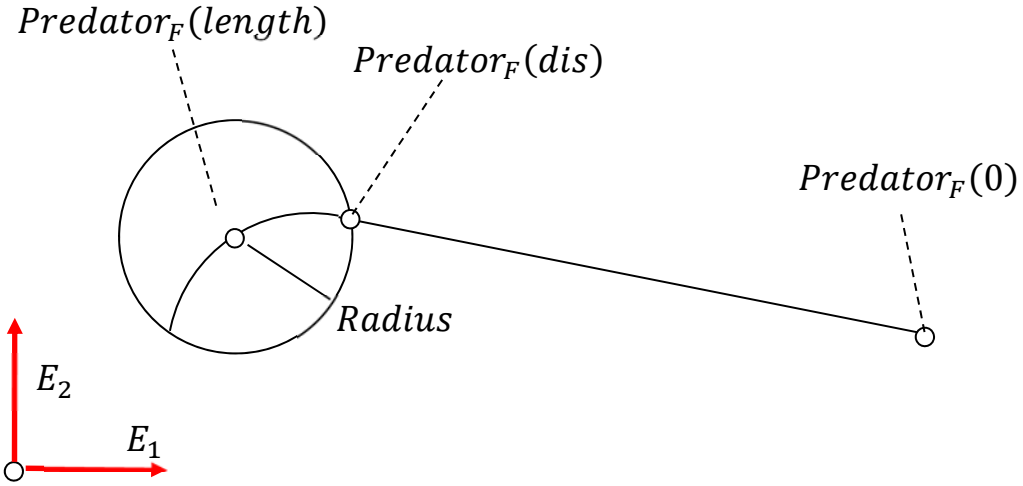
$$:= \text{root}\left(\left|\text{Predator}_F(\text{Length}, \text{Tar}, \text{Pre}) - \text{Predator}_F(\text{dis}, \text{Tar}, \text{Pre})\right| - \text{Radius}, \text{dis}, 0, \text{Length}\right)$$

Далее, сформируем функцию, которая рассчитывает параметр точки пересечения окружности и линии (Рисунок 2.19).

На рисунке 2.19 показано, что заданную в мировой системе координат линию $Predator_F(s)$, пересекает окружность радиуса $Radius$ и с центром в точке $Predator_F(Length)$.

Центр окружности точка $Predator_F(Length)$ также принадлежит линии $Predator_F(s)$. Следует отметить, что линия $Predator_F(s)$ конгруэнтна базовой линии.

В нашем программном коде при помощи встроенной функции поиска корней уравнений в заданном диапазоне $root$, ищется значение параметра dis.



2.19. Точка пересечения окружности и линии

Значение параметра dis соответствует точке $Predator_F(dis)$, которая является точкой пересечения. Значение параметра dis ищется в диапазоне $[0, Length]$.

$$
\begin{aligned}
&\mathrm{Dyn}_{.}\mathrm{Tr}(v_p, v_t, n_t, \Delta t, N) := \\
&\quad t_0 \leftarrow \mathrm{Predator}(\mathrm{Dist}_{start}) \\
&\quad p_0 \leftarrow \mathrm{Predator}(\mathrm{Dist}_{final}) \\
&\quad \mathrm{dis}_{F_0} \leftarrow \mathrm{Dist}_{final} \\
&\quad P \leftarrow p_0 \\
&\quad PT \leftarrow \mathrm{augment}(p_0, t_0) \\
&\quad \text{for } i \in 1..N \\
&\qquad t_i \leftarrow t_{i-1} + (n_t \cdot v_t \cdot \Delta t) \\
&\qquad \mathrm{dis}_{F_i} \leftarrow \mathrm{Point}_{circle}(v_p \cdot \Delta t, \mathrm{dis}_{F_{i-1}}, t_{i-1}, p_{i-1}) \\
&\qquad p_i \leftarrow \mathrm{Predator}_F(\mathrm{dis}_{F_i}, t_{i-1}, p_{i-1}) \\
&\qquad P \leftarrow \mathrm{augment}(P, p_i) \\
&\qquad PT \leftarrow \mathrm{augment}(PT, p_i, t_i) \\
&\quad \text{for } i \in 0..N \\
&\qquad \text{for } j \in 0..N_{point} \\
&\qquad\quad d_j \leftarrow \mathrm{Dist}_{start} + \frac{j}{N_{point}} \cdot (\mathrm{Dist}_{final} - \mathrm{Dist}_{start}) \\
&\qquad\quad x_{i,j} \leftarrow \mathrm{Predator}_F(d_j, t_i, p_i)_0 \\
&\qquad\quad y_{i,j} \leftarrow \mathrm{Predator}_F(d_j, t_i, p_i)_1 \\
&\quad (PT^T \;\; x^T \;\; y^T \;\; P^T)
\end{aligned}
$$

Нами была написана функция расчета точек траектории преследователя, если движется равномерно и прямолинейно.

Входными параметрами этой процедуры – функции служат: вектор скорости преследователя, модуль скорости движения цели, вектор направления цели, период дискретизации итерационного процесса, количество расчетных циклов.

Далее, нами вводится число шагов, в течении которого будет продолжаться итерационный процесс. Это число, когда преследователь достигает своей цели, в нашей тестовой программе подобрано эмпирически, хотя можно было рассчитать это значение. Но это было бы отдельной функцией.

$$N_{tr} := 35$$

$$Tr := \mathrm{Dyn}_{.}\mathrm{Tr}(V_P, V_T, \mathrm{Vector}_{Target}, \Delta T, N_{tr})_{0,0}$$

$$X_f := \mathrm{Dyn}_{.}\mathrm{Tr}(V_P, V_T, \mathrm{Vector}_{Target}, \Delta T, N_{tr})_{0,1} \qquad Y_f := \mathrm{Dyn}_{.}\mathrm{Tr}(V_P, V_T, \mathrm{Vector}_{Target}, \Delta T, N_{tr})_{0,2}$$

$$X_{arr} := \left| \begin{array}{l} p \leftarrow X_f^{\langle 0 \rangle} \\ \text{for } i \in 1..N_{tr} \\ \quad p \leftarrow \text{stack}\left(p, X_f^{\langle i \rangle}\right) \\ p \end{array} \right.$$

$$Y_{arr} := \left| \begin{array}{l} p \leftarrow Y_f^{\langle 0 \rangle} \\ \text{for } i \in 1..N_{tr} \\ \quad p \leftarrow \text{stack}\left(p, Y_f^{\langle i \rangle}\right) \\ p \end{array} \right.$$

$$x_f := X_f^{\langle FRAME \rangle} \qquad y_f := Y_f^{\langle FRAME \rangle}$$

$$Pred_f := Dyn\_Tr\left(V_P, V_T, Vector_{Target}, \Delta T, N_{tr}\right)_{0,3}$$

$$P_{f.X} := \left(Pred_f^{\langle 0 \rangle}\right)_{FRAME} \qquad P_{f.Y} := \left(Pred_f^{\langle 1 \rangle}\right)_{FRAME}$$

Потом мы обращаемся к процедуре Dyn_{Tr} для получения всей картины динамического итерационного процесса.

Переменная Tr - это объединенный массив точек преследователя и цели, в течении всего итерационного процесса. На рисунке 2.20 точки обеих траекторий выделены синим цветом.

На экран выводится вся сеть прогнозируемых траекторий движения. Точки этой сети объединены в единые массивы по координатам. На рисунке 2.20 они выделены красным цветом. В программном коде они соответствуют переменным X_f и Y_f. Затем их преобразовали в массивы X_{arr} и Y_{arr}. На экран выводятся переменные X_{arr} и Y_{arr}.

Также, на экран выводится линия, соединяющая движущиеся точки преследователя и цели. В программном это переменные X_f и Y_f после вызова функции Dyn_{Tr}. После привязки к отдельному кадру, эти переменные преобразованы в переменные x_f и y_f. Именно они выводятся на экран. На рисунке 2.20 точки, соединяющие преследователя и цель в определенный момент времени обозначены черным цветом.

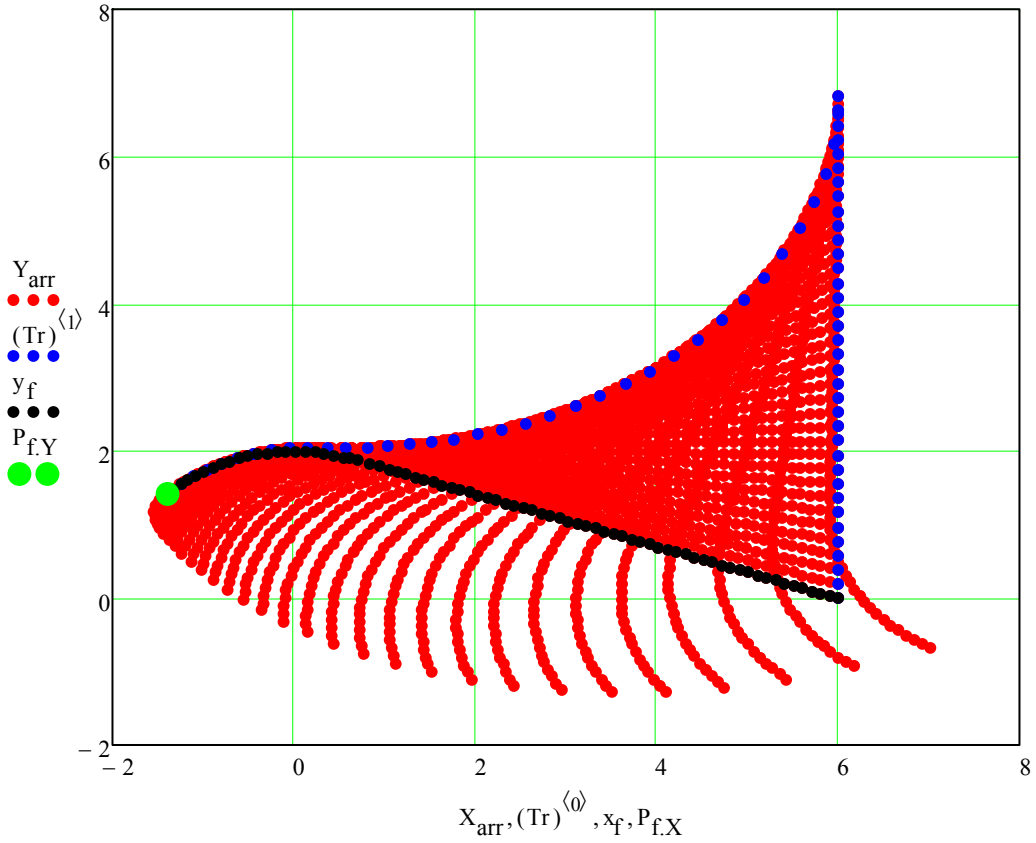


Рисунок 2.20. Итерационный процесс задачи преследования методом погони

На рисунке 2.20 изображен конечный результат программы расчета траектории преследователя, который догоняет цель, используя стратегию погони.

Рисунок 2.20 дополнен ссылкой на анимированное изображение [51] на канале автора.

2.7 КОММЕНТАРИИ К ПРОГРАММЕ КОРРЕКТИРОВКИ УГЛА В НАПРАВЛЕНИИ ПРЕСЛЕДВАТЕЛЯ ПРИ ИСПОЛЬЗВАНИИ СТРАТЕГИИ ПОГОНИ

В предыдущих тестовых программах мы использовали методику построения сети прогнозируемых траекторий и рассматривали на принадлежность к прогнозируемой траектории точки следующего шага преследователя.

$$
\begin{aligned}
\text{Step}(P, N, V, \omega, \Delta t) := \;& E_1 \leftarrow \frac{N}{|N|} \\
& E_2 \leftarrow \frac{\begin{pmatrix} -N_1 \\ N_0 \end{pmatrix}}{|N|} \\
& S \leftarrow P + V \cdot \Delta t \\
& S_p \leftarrow \begin{bmatrix} (S - P)^T \cdot E_1 \\ (S - P)^T \cdot E_2 \end{bmatrix} \\
& \alpha_0 \leftarrow \omega \cdot \Delta t \\
& \alpha \leftarrow \operatorname{acos}\left(\frac{S_{p_0}}{|S_p|}\right) \\
& \alpha_n \leftarrow \alpha - \alpha_0 \quad \text{if } \alpha > \alpha_0 \\
& \alpha_n \leftarrow 0 \quad \text{otherwise} \\
& S_{n.p} \leftarrow \begin{pmatrix} |S_p| \cdot \cos(\alpha_n) \\ |S_p| \cdot \sin(\alpha_n) \end{pmatrix} \\
& H_1 \leftarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\
& H_2 \leftarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\
& h_1 \leftarrow \begin{pmatrix} H_1^T \cdot E_1 \\ H_1^T \cdot E_2 \end{pmatrix} \\
& h_2 \leftarrow \begin{pmatrix} H_2^T \cdot E_1 \\ H_2^T \cdot E_2 \end{pmatrix} \\
& S_n \leftarrow \begin{pmatrix} S_{n.p}^T \cdot h_1 \\ S_{n.p}^T \cdot h_2 \end{pmatrix} + P \\
& S_n
\end{aligned}
$$

Мы, сначала, сформировали функцию $Step$. Ее работа продемонстрирована на рисунке 2.21. Входными параметрами служат: вектор положения преследователя P в мировой системе координат, вектор направления N, к которому должен стремиться вектор скорости преследователя, вектор скорости преследователя V, допустимая угловая скорость вращения преследователя ω и период дискретизации итерационного процесса Δt.

Выходными параметрами служат координаты преследователя после поворота на угол $\omega \cdot \Delta t$ и шага на расстояние $V \cdot \Delta t$.

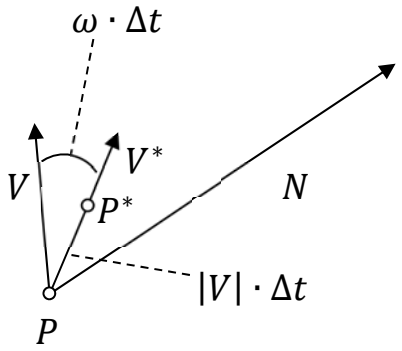


Рисунок 2.21. Функция вращения и шага точки

Задаем абсолютные величины скоростей преследователя и цели, V_P и V_T, а также векторы направления преследователя и цели, $Vector_{Pers}$ и $Vector_{Target}$.

$$V_P := 16 \quad V_T := 12$$

$$Vector_{Pers} := \begin{pmatrix} \sin\left(\frac{5}{4}\pi\right) \\ -\cos\left(\frac{5}{4}\pi\right) \end{pmatrix} = \begin{pmatrix} -0.707 \\ 0.707 \end{pmatrix} \quad Vector_{Target} := \frac{\begin{pmatrix} 0.0 \\ 1 \end{pmatrix}}{\sqrt{1 + 0.}}$$

$$\Delta T := 0.015 \quad Rad := 2$$

$$\omega := \frac{V_P}{Rad} = 8$$

$$Predator_0 := \begin{pmatrix} -1.414 \\ 1.414 \end{pmatrix} \quad Target_0 := \begin{pmatrix} 6 \\ 0 \end{pmatrix}$$

Также задаем период дискретизации итерационного процесса ΔT и минимальный радиус кривизны траектории преследователя Rad. Если известна скорость преследователя V_P и минимальный радиус кривизны Rad, то можем вычислить допустимую угловую частоту вращения ω преследователя. В этом блоке операторов мы задаем начальные положения преследователя и цели.

В следующих операторах мы задаем параметрические уравнения вдоль направлений векторов $Vector_{Pers}$ и $Vector_{Target}$.

$$Target_{line}(t) := Target_0 + t \cdot V_T \cdot Vector_{Target} \quad Predator_{line}(t) := Predator_0 + t \cdot V_P \cdot Vector_{Pers}$$

Определение начальных элементов массивов преследователя и цели.

$$Tar_0 := Target_0 \quad Tar_{X_0} := \left(Tar_0\right)_0 \quad Tar_{Y_0} := \left(Tar_0\right)_1$$

$$Pred_0 := Predator_0 \quad Pred_{X_0} := \left(Pred_0\right)_0 \quad Pred_{Y_0} := \left(Pred_0\right)_1$$

Перевод параметрических уравнений прямых в массивы точек. Это технический прием, необходим для визуализации отрезков.

$$i := 1..N_{Target}$$

$$Tar_i := Tar_{i-1} + \Delta T \cdot V_T \cdot Vector_{Target}$$

$$Tar_{X_i} := (Tar_i)_0 \quad Tar_{Y_i} := (Tar_i)_1$$

$$Pred_i := Pred_{i-1} + \Delta T \cdot V_P \cdot Vector_{Pers}$$

$$Pred_{X_i} := (Pred_i)_0 \quad Pred_{Y_i} := (Pred_i)_1$$

Вводим число временных отрезков итерационного процесса.

$$N_{Target} := 69$$

Основная расчетная процедура.

$Calc_1 :=$

$P_0 \leftarrow Predator_0$

$V_0 \leftarrow V_P \cdot Vector_{Pers}$

$v \leftarrow V_P$

$N_0 \leftarrow Tar_0 - P_0$

$\Delta R_{animal} \leftarrow |P_0 - Tar_0|$

$\delta r \leftarrow v \cdot \Delta T$

$PP \leftarrow P_0$

$NN \leftarrow N_0$

$VV \leftarrow V_0$

for $i \in 1..N_{Target}$

$P_i \leftarrow Step(P_{i-1}, N_{i-1}, V_{i-1}, \omega, \Delta T)$

$PP \leftarrow augment(PP, P_i)$

$N_i \leftarrow Tar_i - P_i$

$NN \leftarrow augment(NN, N_i)$

$V_i \leftarrow v \cdot \frac{P_i - P_{i-1}}{|P_i - P_{i-1}|}$

$VV \leftarrow augment(VV, V_i)$

$\Delta R_{animal} \leftarrow |Tar_i - P_i|$

$(PP^T \quad NN^T \quad VV^T)$

У этой процедуры нет входных параметров. Она использует глобальный массив точек траектории цели, глобально определенные начальные координаты преследователя, начальную скорость преследователя.

Выходными параметрами служат массив рассчитанных итерационно координат преследователя. Массив векторов, соединяющих соответствующие точки преследователя и цели. Массив векторов скоростей преследователя.

В расчетном цикле используется для расчета текущего шага преследователя функция $Step$. В нее подаются предыдущее положение

преследователя P_{i-1}, вектор направления N_{i-1}, к которому должно стремиться направление скорости преследователя, скорость преследователя V_{i-1}. В тексте процедуры приняты именно такие обозначения.

Следующий оператор позволяет подготовить для визуализации линию, соединяющую преследователя и цель.

$$Dir(t) := \left(Calc1_{0,0}^{T}\right)^{\langle FRAME\rangle} + \left(Calc1_{0,1}^{T}\right)^{\langle FRAME\rangle} \cdot t$$

Формальная визуализация вектора скорости преследователя.

$$V_{Speed}(t) := \left(Calc1_{0,0}^{T}\right)^{\langle FRAME\rangle} + \left(Calc1_{0,2}^{T}\right)^{\langle FRAME\rangle} \cdot t$$

$$i := 0..1 \quad tt_i := \frac{i}{6}$$

$$V_{x_i} := V_{Speed}(tt_i)_0 \quad V_{y_i} := V_{Speed}(tt_i)_1$$

Результат работы программы представлен на рисунке 2.22.

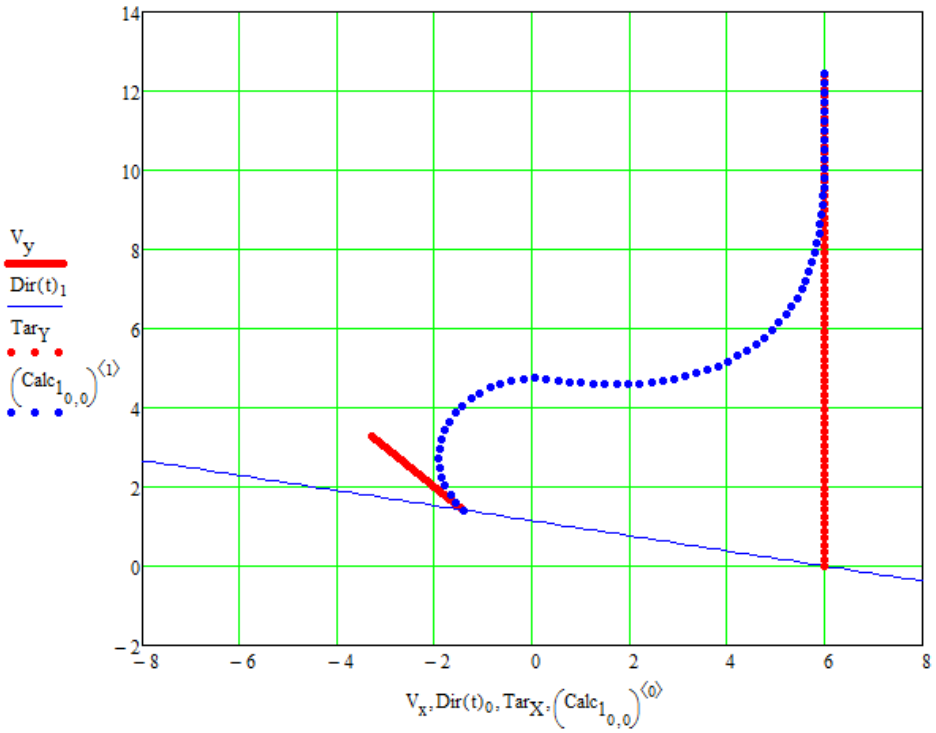


Рисунок 2.22. Корректировка направления движения преследователя

Рисунок 2.22 дополнен ссылкой на анимированное изображение [52], которое можно будет посмотреть на канале автора.

2.8 КОММЕНТАРИИ К ПРОГРАММЕ МОДЕЛИ ПОВЕДЕНИЯ ОБЪЕКТОВ В ЗАДАЧЕ ПРЕСЛЕДОВАНИЯ

В этом параграфе мы отдельно рассмотрим листинг программы «Модели поведения объектов в задаче преследования». В данной программе показано

формирование траектории движения преследователя, который придерживается стратегии погони с корректировкой направления движения, стремясь вектор своей скорости совместить с направлением вектора, соединяющим преследователя и цель. Цель, придерживается такой же стратегии, стремясь от преследователя, корректируя свое направление движения.

Считывание из внешних точек поверхности, упорядоченных по горизонталям.

$A1 := READPRN("C:\WorkMathCAD\land2\l1.txt")$
$A2 := READPRN("C:\WorkMathCAD\land2\l2.txt")$
$A3 := READPRN("C:\WorkMathCAD\land2\l3.txt")$
$A4 := READPRN("C:\WorkMathCAD\land2\l4.txt")$
$A5 := READPRN("C:\WorkMathCAD\land2\l5.txt")$

Горизонтали поверхности созданы в системе "AutoCAD", в дальнейшем были импортированы в текстовые файлы

$i := 0..100$

Введение дополнительных нулевых точек по краям, с целью уменьшения осцилляций во время регрессии

$Xkr1_i := i \; Ykr1_i := 0 \; Zkr1_i := 0 \; Xkr3_i := 0 \; Ykr3_i := i \; Zkr3_i := 0$

$Xkr2_i := i \; Ykr2_i := 100 \; Zkr2_i := 0 \; Xkr4_i := 100 \; Ykr4_i := i \; Zkr4_i := 0$

Подготовка к двумерной полиномиальной регрессии

$LandPointX := stack(A1^{\langle 0 \rangle}, A2^{\langle 0 \rangle}, A3^{\langle 0 \rangle}, A4^{\langle 0 \rangle}, A5^{\langle 0 \rangle}, Xkr1, Xkr2, Xkr3, Xkr4)$
$LandPointY := stack(A1^{\langle 1 \rangle}, A2^{\langle 1 \rangle}, A3^{\langle 1 \rangle}, A4^{\langle 1 \rangle}, A5^{\langle 1 \rangle}, Ykr1, Ykr2, Ykr3, Ykr4)$
$LandPointZ := stack(A1^{\langle 2 \rangle}, A2^{\langle 2 \rangle}, A3^{\langle 2 \rangle}, A4^{\langle 2 \rangle}, A5^{\langle 2 \rangle}, Zkr1, Zkr2, Zkr3, Zkr4)$

Покоординатное объединение массивов

$LandPoint := (LandPointX \;\; LandPointY \;\; LandPointZ)^T$

Итоговая матрица из трех строк

Полиномиальная двумерная регрессия

$CoeffRegressionXY := augment(LandPointX, LandPointY)$

Объединение массивов X и Y в одну матрицу из двух столбцов. Подготовка к двумерной регрессии.

$RegressionXY := regress(CoeffRegressionXY, LandPointZ, 8)$

Двумерная полиномиальная регрессия. Степень полиномиальной функции подобрана опытным путем

$ArrayRangeLandPointX := cols(LandPointX) - 1$

Количество точек в массивах

$Surface(u,v) := interp\left[RegressionXY, CoeffRegressionXY, LandPointZ, \begin{pmatrix} u \\ v \end{pmatrix}\right]$

Итоговая поверхность

$SurfacePicture := CreateMesh(Surface, 0, 100, 0, 100, 200, 200)$

Вывод на экран

$Surface_1(u,v) := Surface\left(\frac{u+100}{2}, \frac{v+100}{2}\right)$

$SurfacePicture_1 := CreateMesh(Surface_1, -100, 100, -100, 100, 200, 200)$

Преобразование исходной поверхности заданной точечным базисом.

$$Surface_2(u,v) := 50 \cdot Surface_1(u,v)$$

$$SurfacePicture_2 := CreateMesh(Surface_2, -100, 100, -100, 100, 200, 200)$$

$$P_2(u,v) := \begin{pmatrix} u \\ v \\ Surface_2(u,v) \end{pmatrix}$$

Расчетная поверхность

Расчет узлов

$$i := 0..100 \qquad j := 0..100$$

$$LandX_{2_i} := -100 + \frac{i}{100}(100 + 100) \quad LandY_{2_j} := -100 + \frac{j}{100}\cdot(100 + 100) \quad LandZ_{2_{i,j}} := Surface_2(LandX_{2_i}, LandY_{2_j})$$

Расчет частных производных в узлах поверхности

$$LandDiffZfromXpoint_2 := \left| \begin{array}{l} hX \leftarrow LandX_{2_1} - LandX_{2_0} \\ \text{for } i \in 0..100 \\ \quad \text{for } j \in 0..100 \\ \quad\quad DxZZZ_{i,j} \leftarrow \dfrac{LandZ_{2_{i+1,j}} - LandZ_{2_{i,j}}}{LandX_{2_{i+1}} - LandX_{2_i}} \text{ if } i = 0 \\ \quad\quad DxZZZ_{i,j} \leftarrow \dfrac{LandZ_{2_{i,j}} - LandZ_{2_{i-1,j}}}{LandX_{2_i} - LandX_{2_{i-1}}} \text{ if } i = 100 \\ \quad\quad DxZZZ_{i,j} \leftarrow \dfrac{LandZ_{2_{i+1,j}} - LandZ_{2_{i-1,j}}}{2 \cdot hX} \text{ if } (i \neq 0) \wedge (i \neq 100) \\ DxZZZ \end{array} \right.$$

$$LandDiffZfromYpoint_2 := \left| \begin{array}{l} hY \leftarrow LandY_{2_1} - LandY_{2_0} \\ \text{for } i \in 0..100 \\ \quad \text{for } j \in 0..100 \\ \quad\quad DyZZZ_{i,j} \leftarrow \dfrac{LandZ_{2_{i,j+1}} - LandZ_{2_{i,j}}}{LandY_{2_{j+1}} - LandY_{2_j}} \text{ if } j = 0 \\ \quad\quad DyZZZ_{i,j} \leftarrow \dfrac{LandZ_{2_{i,j}} - LandZ_{2_{i,j-1}}}{LandY_{2_j} - LandY_{2_{j-1}}} \text{ if } j = 100 \\ \quad\quad DyZZZ_{i,j} \leftarrow \dfrac{LandZ_{2_{i,j+1}} - LandZ_{2_{i,j-1}}}{2 \cdot hY} \text{ if } (j \neq 0) \wedge (j \neq 100) \\ DyZZZ \end{array} \right.$$

Задание поверхностей из частных производных

$$LandXY_2 := augment(LandX_2, LandY_2)$$

$$\text{CoeffLandSplineZX}_2 := \text{cspline}(\text{LandXY}_2, \text{LandDiffZfromXpoint}_2)$$

$$\text{CoeffLandSplineZY}_2 := \text{cspline}(\text{LandXY}_2, \text{LandDiffZfromYpoint}_2)$$

$$\text{DiffZX}_2(u,v) := \text{interp}\left[\text{CoeffLandSplineZX}_2, \text{LandXY}_2, \text{LandDiffZfromXpoint}_2, \begin{pmatrix} u \\ v \end{pmatrix}\right]$$

$$\text{DiffZY}_2(u,v) := \text{interp}\left[\text{CoeffLandSplineZY}_2, \text{LandXY}_2, \text{LandDiffZfromYpoint}_2, \begin{pmatrix} u \\ v \end{pmatrix}\right]$$

Функция вычисления единичного вектора нормали $n(X,Y)$ к поверхности $Z = Z(X,Y)$.

$$n(X,Y) = \frac{\begin{bmatrix} -\frac{\partial Z}{\partial X} \\ -\frac{\partial Z}{\partial Y} \\ 1 \end{bmatrix}}{\sqrt{1 + \frac{\partial Z}{\partial X}^2 + \frac{\partial Z}{\partial Y}^2}}$$

В динамическом базисе вектор нормали – это вектор аппликаты

$$\text{Normal}_S(u,v) := \frac{-1}{\sqrt{1 + \text{DiffZX}_2(u,v)^2 + \text{DiffZY}_2(u,v)^2}} \begin{pmatrix} \text{DiffZX}_2(u,v) \\ \text{DiffZY}_2(u,v) \\ -1 \end{pmatrix}$$

Расчет вектора нормали

Функция расчета единичного вектора производной поверхности $Z = Z(X,Y)$ по направлению, заданному углами $(cos(\lambda), sin(\lambda))$

$$v(X,Y,\lambda) = \begin{bmatrix} \frac{cos(\lambda)}{\sqrt{1 + \left(\frac{\partial Z}{\partial X} \cdot cos(\lambda) + \frac{\partial Z}{\partial Y} \cdot sin(\lambda)\right)^2}} \\ \frac{sin(\lambda)}{\sqrt{1 + \left(\frac{\partial Z}{\partial X} \cdot cos(\lambda) + \frac{\partial Z}{\partial Y} \cdot sin(\lambda)\right)^2}} \\ \frac{\frac{\partial Z}{\partial X} \cdot cos(\lambda) + \frac{\partial Z}{\partial Y} \cdot sin(\lambda)}{\sqrt{1 + \left(\frac{\partial Z}{\partial X} \cdot cos(\lambda) + \frac{\partial Z}{\partial Y} \cdot sin(\lambda)\right)^2}} \end{bmatrix}$$

В динамическом базисе вектор производной по направлению - это вектор абсциссы

$$\mathrm{SpeedVector_S}(u,v,\lambda_{cos},\lambda_{sin}) := \begin{bmatrix} \dfrac{\lambda_{cos}}{\sqrt{1+\left(\mathrm{DiffZX_2}(u,v)\cdot\lambda_{cos}+\mathrm{DiffZY_2}(u,v)\cdot\lambda_{sin}\right)^2}} \\ \dfrac{\lambda_{sin}}{\sqrt{1+\left(\mathrm{DiffZX_2}(u,v)\cdot\lambda_{cos}+\mathrm{DiffZY_2}(u,v)\cdot\lambda_{sin}\right)^2}} \\ \dfrac{\left(\mathrm{DiffZX_2}(u,v)\cdot\lambda_{cos}+\mathrm{DiffZY_2}(u,v)\cdot\lambda_{sin}\right)}{\sqrt{1+\left(\mathrm{DiffZX_2}(u,v)\cdot\lambda_{cos}+\mathrm{DiffZY_2}(u,v)\cdot\lambda_{sin}\right)^2}} \end{bmatrix}$$

Расчет вектора абсциссы, сонаправленного вектору скорости движения объекта

Функция вычисления ординаты динамического базиса, как продукт векторного произведения вектора нормали к поверхности и вектора производной по направлению (вдоль скорости)

$$\mathrm{OrdinataVector_S}(u,v,\lambda_{cos},\lambda_{sin}) := \mathrm{Normal_S}(u,v)\times\mathrm{SpeedVector_S}(u,v,\lambda_{cos},\lambda_{sin})$$

Расчет вектора ординаты, векторное произведение нормали и абсциссы

Задаются направления движения и скорости перемещения объектов, участников задачи преследования.

Также задаются периоды вращения. Это необходимо для расчета угловых скоростей вращения объектов. Задание угловой скорости вращения является основным моментом в модели задачи преследования в нашей программе.

Задание порогового значения, при достижении которого будет считаться, что преследователь догнал цель. Цель достигнута и расчетный цикл прерывается. Программа работает, если нет достижения цели в течении 500 расчетных циклов.

Временной интервал

$$\Delta T := \frac{25.3}{500}$$

Начальный угол движения преследователя

$$\lambda 0_{rabbit} := \frac{3}{2}\pi$$

Скорость движения цели

$$V_{rabbit} := 12$$

Период вращения, (к угловой скорости вращения цели)

$$T_{rabbit} := 22$$

Угловая скорость вращения цели

$$\omega_{rabbit} := \frac{2\pi}{T_{rabbit}}$$

Скорость движения преследователя

$$V_{fox} := 26$$

Начальный угол движения преследователя

$$\lambda 0_{fox} := \frac{3\pi}{2} - \frac{\pi}{4}$$

Период вращения, (к угловой скорости вращения преследователя)

$$T_{fox} := 6$$

Угловая скорость вращения преследователя

$$\omega_{fox} := \frac{2\pi}{T_{fox}}$$

Расстояние между целью и преследователем, при достижении которого считается, что преследователь достиг цели

$$\Delta R_0 := V_{fox}\cdot\Delta T$$

Задание ортонормированного базиса мировой системы координат

$H_1 := (1\ 0\ 0)^T$ $H_2 := (0\ 1\ 0)^T$ $H_3 := (0\ 0\ 1)^T$

Задания начальных положений цели и преследователя на горизонтальной плоскости проекций

Начальное положение цели

$$\begin{pmatrix} U0_{rabbit} \\ V0_{rabbit} \end{pmatrix} := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Начальное положение преследователя

$$\begin{pmatrix} U0_{fox} \\ V0_{fox} \end{pmatrix} := \begin{pmatrix} 100 \\ 100 \end{pmatrix}$$

Функция вычисления векторов базиса мировой системы координат в динамическом базисе. Входной параметр, как видно, это компоненты динамического базиса

$$\begin{aligned} \text{ConjugateBasis}(E) := \; & h_1 \leftarrow \left(H_1 \cdot E^{\langle 0 \rangle} \;\; H_1 \cdot E^{\langle 1 \rangle} \;\; H_1 \cdot E^{\langle 2 \rangle}\right)^T \\ & h_2 \leftarrow \left(H_2^T \cdot E^{\langle 0 \rangle} \;\; H_2^T \cdot E^{\langle 1 \rangle} \;\; H_2^T \cdot E^{\langle 2 \rangle}\right)^T \\ & h_3 \leftarrow \left(H_3^T \cdot E^{\langle 0 \rangle} \;\; H_3^T \cdot E^{\langle 1 \rangle} \;\; H_3^T \cdot E^{\langle 2 \rangle}\right)^T \\ & h \leftarrow \text{augment}(h_1, h_2, h_3) \end{aligned}$$

Процедура определения, как будет выглядеть базис H из базиса E

Операторы основного расчетного блока разберем подробнее. Они пронумерованы и мы постараемся объяснить значение каждого оператора.

1 Оператор вычисления динамического базиса цели. Динамический базис «Цельа» определяется направлением движения «Цели» и нормалью к поверхности в точке, где находится «Цель». Начальное положение базиса. Номер цикла 0

2 Точка пространства, где находится «Цель». Начальное положение «Цели». Номер цикла 0

3 Оператор вычисления динамического базиса «Преследователя». Динамический базис «Преследователя» определяется направлением движения «Преследователя» и нормалью к поверхности в точке, где находится «Преследователь». Начальное положение базиса. Номер цикла 0

4 Точка пространства, где находится «Лиса». Начальное положение «Преследователя». Номер цикла 0

5 Номер цикла 1

6 Начальное расстояние между «Целью» и «Лисой»

7 Условный цикл «Исполнять, пока индекс цикла лежит между 1 и 500 и расстояние между объектами не меньше порогового значения»

8 Проекция «Преследователя» на плоскость, образованную абсциссой и ординатой, динамического базиса «Цели»

9 Анализ положения «Преследователя» на предмет того, в какой полуплоскости динамического базиса «Цели», она находится

10 Условный оператор того, куда совершать шаг «Цельу». Если ордината «Преследователя» больше 0, то «Цели» надо вращаться по часовой и делать шаг. Если ордината «Преследователя» меньше 0, то «Цели»

надо вращаться против часовой и делать шаг. Все, разумеется, рассчитывается в динамической системе координат «Цели». Расчет новых координат «Цели».

11 Расчет того как выглядит базис мировой системы координат в системе координат «Цели»

12 Перевод координат «Цели» из динамического базиса в мировую систему координат

13 Новое пространственное положение «Цели»

14 Вычисление углов направления движения «Цели»

15 Новый динамический базис «Цели». Для следующего этапа итераций

16 Проекция «Цели» на плоскость, образованную абсциссой и ординатой, динамического базиса «Преследователя»

17 Анализ положения «Цели» на предмет того, в какой полуплоскости динамического базиса «Преследователя», он находится

18 Условный оператор того, куда совершать шаг «Лисе». Если ордината «Цели» больше 0, то «Цели» надо вращаться против часовой и делать шаг. Если ордината «Цели» меньше 0, то «Цельу» надо вращаться по часовой и делать шаг. Все, разумеется, рассчитывается в динамической системе координат «Преследователя». Расчет новых координат «Преследователя».

19 Расчет того как выглядит базис мировой системы координат в системе координат «Преследователя»

20 Перевод координат «Преследователя» из динамического базиса в мировую систему координат

21 Новое пространственное положение «Преследователя»

22 Вычисление углов направления движения «Преследователя»

23 Новый динамический базис «Преследователя». Для следующего этапа итераций

24 Расчет нового расстояния между «Цельом» и «Лисой»

25 Переход на следующий цикл

26 Число циклов

27 Возвращаемые переменные из процедуры. Массив точек траектории «Цели», массивы направлений движения «Цели» в каждой точке, массив точек траектории «Преследователя», массивы направлений движения «Преследователя» в каждой точке, число циклов

$$\begin{aligned}
&Sit := \left| \left(VX_{rabbit_0} \;\; VZ_{rabbit_0} \;\; VY_{rabbit_0} \right) \leftarrow \right. && 1\\
&\qquad \left(SpeedVector_S\left(U0_{rabbit}, V0_{rabbit}, \cos\left(\lambda 0_{rabbit}\right), \sin\left(\lambda 0_{rabbit}\right)\right)\right. && 1\\
&\qquad\qquad Normal_S\left(U0_{rabbit}, V0_{rabbit}\right) && 1\\
&\qquad\quad \left. Normal_S\left(U0_{rabbit}, V0_{rabbit}\right) \times SpeedVector_S\left(U0_{rabbit}, V0_{rabbit}, \cos\left(\lambda 0_{rabbit}\right), \sin\left(\lambda 0_{rabbit}\right)\right)\right) && 1\\
&\left| R_{rabbit_0} \leftarrow P_2\left(U0_{rabbit}, V0_{rabbit}\right) \right. && 2
\end{aligned}$$

$$\left(VX_{fox_0}\ \ VZ_{fox_0}\ \ VY_{fox_0}\right) \leftarrow \left(SpeedVector_S\left(U0_{fox}, V0_{fox}, \cos\left(\lambda 0_{fox}\right), \sin\left(\lambda 0_{fox}\right)\right)\right. \quad 3$$

$$Normal_S\left(U0_{fox}, V0_{fox}\right) \quad 3$$

$$\left.Normal_S\left(U0_{fox}, V0_{fox}\right) \times SpeedVector_S\left(U0_{fox}, V0_{fox}, \cos\left(\lambda 0_{fox}\right), \sin\left(\lambda 0_{fox}\right)\right)\right) \quad 3$$

$$R_{fox_0} \leftarrow P_2\left(U0_{fox}, V0_{fox}\right) \quad 4$$

$$i \leftarrow 1 \quad 5$$

$$\Delta R_{animal} \leftarrow \left|R_{rabbit_0} - R_{fox_0}\right| \quad 6$$

$$\text{while } (i \geq 1) \wedge (i \leq 500) \wedge \left(\Delta R_{animal} \geq \Delta R_0\right) \quad 7$$

$$r_{projectionFox} \leftarrow R_{fox_{i-1}} + \frac{\left(R_{rabbit_{i-1}} - R_{fox_{i-1}}\right) \cdot VZ_{rabbit_{i-1}}}{VZ_{rabbit_{i-1}} \cdot VZ_{rabbit_{i-1}}} \cdot VZ_{rabbit_{i-1}} \quad 8$$

$$y_{projectionFox} \leftarrow \left(r_{projectionFox} - R_{rabbit_{i-1}}\right) \cdot VY_{rabbit_{i-1}} \quad 9$$

$$Rabbit_{local} \leftarrow V_{rabbit} \cdot \Delta T \cdot \left(\cos\left(\omega_{rabbit} \cdot \Delta T\right)\ \ -\sin\left(\omega_{rabbit} \cdot \Delta T\right)\ \ 0\right) \text{ if } y_{projectionFox} \geq 0 \quad 10$$

$$Rabbit_{local} \leftarrow V_{rabbit} \cdot \Delta T \cdot \left(\cos\left(\omega_{rabbit} \cdot \Delta T\right)\ \ \sin\left(\omega_{rabbit} \cdot \Delta T\right)\ \ 0\right) \text{ otherwise} \quad 10$$

$$h_{rabbit} \leftarrow ConjugateBasis\left(augment\left(VX_{rabbit_{i-1}}, VY_{rabbit_{i-1}}, VZ_{rabbit_{i-1}}\right)\right) \quad 11$$

$$Rabbit_{world} \leftarrow \left[Rabbit_{local} \cdot h_{rabbit}^{\langle 0 \rangle} + \left(R_{rabbit_{i-1}}\right)_0\ \ Rabbit_{local} \cdot h_{rabbit}^{\langle 1 \rangle}\right. \quad 12$$

$$\left. + \left(R_{rabbit_{i-1}}\right)_1\ \ Rabbit_{local} \cdot h_{rabbit}^{\langle 2 \rangle} + \left(R_{rabbit_{i-1}}\right)_2\right]^T \quad 12$$

$$R_{rabbit_i} \leftarrow P_2\left(Rabbit_{world_0}, Rabbit_{world_1}\right) \quad 13$$

$$\left(\cos\lambda_{rabbit}\ \ \sin\lambda_{rabbit}\right) \leftarrow \left[\frac{\left(R_{rabbit_i}\right)_0 - \left(R_{rabbit_{i-1}}\right)_0}{\sqrt{\left[\left(R_{rabbit_i}\right)_0 - \left(R_{rabbit_{i-1}}\right)_0\right]^2 + \left[\left(R_{rabbit_i}\right)_1 - \left(R_{rabbit_{i-1}}\right)_1\right]^2}}\right. \quad 14$$

$$\left. \frac{\left(R_{rabbit_i}\right)_1 - \left(R_{rabbit_{i-1}}\right)_1}{\sqrt{\left[\left(R_{rabbit_i}\right)_0 - \left(R_{rabbit_{i-1}}\right)_0\right]^2 + \left[\left(R_{rabbit_i}\right)_1 - \left(R_{rabbit_{i-1}}\right)_1\right]^2}}\right] \quad 14$$

$$\left(VX_{rabbit_i}\ \ VZ_{rabbit_i}\ \ VY_{rabbit_i}\right) \leftarrow \left(SpeedVector_S\left(Rabbit_{world_0}, Rabbit_{world_1}, \cos\lambda_{rabbit}, \sin\lambda_{rabbit}\right)\right. \quad 15$$

$$Normal_S\left(Rabbit_{world_0}, Rabbit_{world_1}\right) \quad 15$$

$$\left.Normal_S\left(Rabbit_{world_0}, Rabbit_{world_1}\right) \times SpeedVector_S\left(Rabbit_{world_0}, Rabbit_{world_1}, \cos\lambda_{rabbit}, \sin\lambda_{rabbit}\right)\right) \quad 15$$

$$r_{projectionRabbit} \leftarrow R_{rabbit_{i-1}} + \frac{\left(R_{fox_{i-1}} - R_{rabbit_{i-1}}\right) \cdot VZ_{fox_{i-1}}}{VZ_{fox_{i-1}} \cdot VZ_{fox_{i-1}}} \cdot VZ_{fox_{i-1}} \quad 16$$

$$y_{projectionRabbit} \leftarrow \left(r_{projectionRabbit} - R_{fox_{i-1}}\right) \cdot VY_{fox_{i-1}} \quad 17$$

$$Fox_{local} \leftarrow V_{fox} \cdot \Delta T \cdot \left(\cos\left(\omega_{fox} \cdot \Delta T\right) \quad \sin\left(\omega_{fox} \cdot \Delta T\right) \quad 0\right) \text{ if } y_{projectionRabbit} \geq 0 \qquad 18$$

$$Fox_{local} \leftarrow V_{fox} \cdot \Delta T \cdot \left(\cos\left(\omega_{fox} \cdot \Delta T\right) \quad -\sin\left(\omega_{fox} \cdot \Delta T\right) \quad 0\right) \text{ otherwise} \qquad 18$$

$$h_{fox} \leftarrow \text{ConjugateBasis}\left(\text{augment}\left(VX_{fox_{i-1}}, VY_{fox_{i-1}}, VZ_{fox_{i-1}}\right)\right) \qquad 19$$

$$Fox_{world} \leftarrow \left[Fox_{local} \cdot h_{fox}^{\langle 0 \rangle} + \left(R_{fox_{i-1}}\right)_0 \quad Fox_{local} \cdot h_{fox}^{\langle 1 \rangle} + \left(R_{fox_{i-1}}\right)_1 \quad Fox_{local} \cdot h_{fox}^{\langle 2 \rangle} + \left(R_{fox_{i-1}}\right)_2\right]^T \qquad 20$$

$$R_{fox_i} \leftarrow P_2\left(Fox_{world_0}, Fox_{world_1}\right) \qquad 21$$

$$\left(\cos\lambda_{fox} \quad \sin\lambda_{fox}\right) \leftarrow \left[\frac{\left(R_{fox_i}\right)_0 - \left(R_{fox_{i-1}}\right)_0}{\sqrt{\left[\left(R_{fox_i}\right)_0 - \left(R_{fox_{i-1}}\right)_0\right]^2 + \left[\left(R_{fox_i}\right)_1 - \left(R_{fox_{i-1}}\right)_1\right]^2}} \quad \frac{\left(R_{fox_i}\right)_1 - \left(R_{fox_{i-1}}\right)_1}{\sqrt{\left[\left(R_{fox_i}\right)_0 - \left(R_{fox_{i-1}}\right)_0\right]^2 + \left[\left(R_{fox_i}\right)_1 - \left(R_{fox_{i-1}}\right)_1\right]^2}}\right] \qquad 22$$

$$\left(VX_{fox_i} \quad VZ_{fox_i} \quad VY_{fox_i}\right) \leftarrow \left(\text{SpeedVector}_S\left(Fox_{world_0}, Fox_{world_1}, \cos\lambda_{fox}, \sin\lambda_{fox}\right)\right. \qquad 23$$

$$\text{Normal}_S\left(Fox_{world_0}, Fox_{world_1}\right) \qquad 23$$

$$\left.\text{Normal}_S\left(Fox_{world_0}, Fox_{world_1}\right) \times \text{SpeedVector}_S\left(Fox_{world_0}, Fox_{world_1}, \cos\lambda_{fox}, \sin\lambda_{fox}\right)\right) \qquad 23$$

$$\Delta R_{animal} \leftarrow \left|R_{rabbit_i} - R_{fox_i}\right| \qquad 24$$

$$i \leftarrow i + 1 \qquad 25$$

$$Total \leftarrow i - 1 \qquad 26$$

$$\left(R_{rabbit} \quad VX_{rabbit} \quad VY_{rabbit} \quad VZ_{rabbit} \quad R_{fox} \quad VX_{fox} \quad VY_{fox} \quad VZ_{fox} \quad Total\right) \qquad 27$$

$Frame_{Total} := \left(Sit^T\right)_8$ $\qquad Frame_{Total} = 201$ ***Общее количество кадров***

Подготовка к визуализации динамического базиса «Цели»

Задание лучей, вдоль векторов динамического базиса "Цели"

$$L3_2(u,v,t) := P_2(u,v) + t \cdot \left[\left(Sit^T\right)_3\right]_{FRAME} \qquad Line3_2(t) := L3_2\left(\left[\left[\left(Sit^T\right)_0\right]_{FRAME}\right]_0, \left[\left[\left(Sit^T\right)_0\right]_{FRAME}\right]_1, t\right)$$

$$L1_2(u,v,t) := P_2(u,v) + t \cdot \left[\left(Sit^T\right)_1\right]_{FRAME} \qquad Line1_2(t) := L1_2\left(\left[\left[\left(Sit^T\right)_0\right]_{FRAME}\right]_0, \left[\left[\left(Sit^T\right)_0\right]_{FRAME}\right]_1, t\right)$$

$$L2_2(u,v,t) := P_2(u,v) + t \cdot \left[\left(Sit^T\right)_2\right]_{FRAME} \qquad Line2_2(t) := L2_2\left(\left[\left[\left(Sit^T\right)_0\right]_{FRAME}\right]_0, \left[\left[\left(Sit^T\right)_0\right]_{FRAME}\right]_1, t\right)$$

Вывод на экран динамического базиса «Цели»

Вывод на экран

$NormalPicture_2 := CreateSpace(Line3_2, 0, 15, 50)$

$SpeedPicture_2 := CreateSpace(Line1_2, 0, 15, 50)$

$OrdinataPicture_2 := CreateSpace(Line2_2, 0, 15, 50)$

К расчету и выводу на экран динамического базиса

Формирование "Шлейфа" "Цели"

$i := 0 .. Frame_{Total}$ $X_{rabbit_i} := \left[\left[\left(Sit^T\right)_0\right]_i\right]_0$ $Y_{rabbit_i} := \left[\left[\left(Sit^T\right)_0\right]_i\right]_1$ $Z_{rabbit_i} := \left[\left[\left(Sit^T\right)_0\right]_i\right]_2$

Формирование "Шлейфа" "Преследователя"

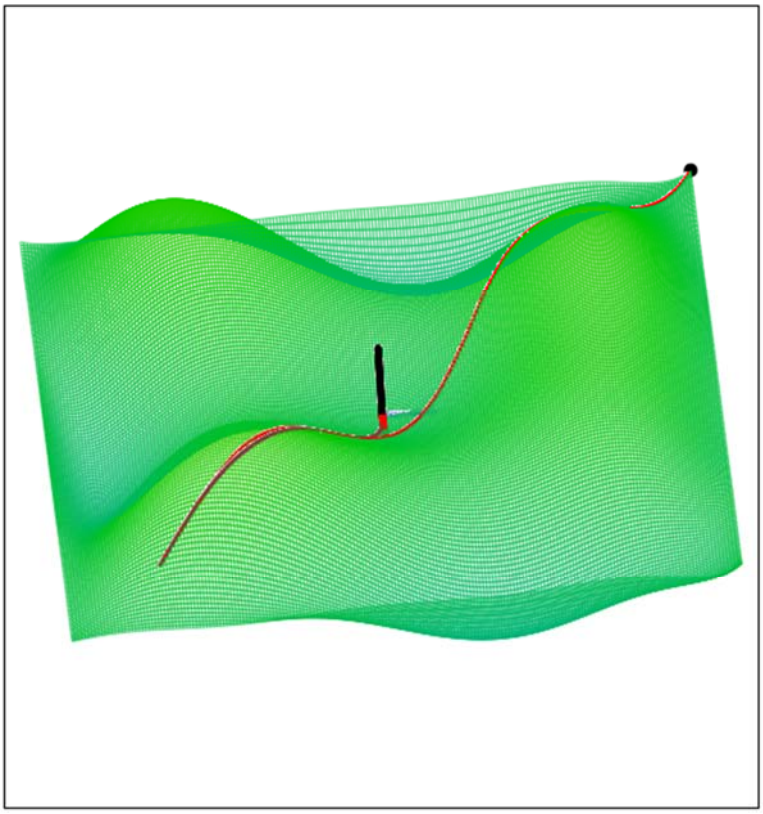
$X_{fox_i} := \left[\left[\left(Sit^T\right)_4\right]_i\right]_0$ $Y_{fox_i} := \left[\left[\left(Sit^T\right)_4\right]_i\right]_1$ $Z_{fox_i} := \left[\left[\left(Sit^T\right)_4\right]_i\right]_2$

Точка с "Лисой" в кадре

$FrFox_{X_{0,0}} := X_{fox_{FRAME}}$ $FrFox_{Y_{0,0}} := Y_{fox_{FRAME}}$ $FrFox_{Z_{0,0}} := Z_{fox_{FRAME}}$

$Frame_{Total} = 201$ *Итоговая картинка*

На рисунке 2.23 показана траектория движения цели. Визуализация цели сопровождается динамическим базисом. Рисунок 2.23 дополнен ссылкой на анимированное изображение [53], которое можно будет посмотреть на канале автора.



$SurfacePicture_2, NormalPicture_2, SpeedPicture_2, OrdinataPicture_2, (X_{rabbit}, Y_{rabbit}, Z_{rabbit}), (X_{fox}, Y_{fox}, Z_{fox}), (FrFox_X, FrFox_Y, FrFox_Z)$

Рисунок 2.23. Корректировка направления движения преследователем и целью

3. МОДЕЛИ ГРУППОВОГО ПРЕСЛЕДОВАНИЯ

В данной главе мы рассмотрим несколько моделей группового преследования. Модель преследования группой одиночной цели описывает то, что участники в процессе преследования имеют различные цели и придерживаются различных стратегий. Данная модель преследования похожа на взятие противника в кольцо.

Модель группового преследования с жесткими связями описывает процесс, когда преследователи имеют строй в процессе движения. Нарушения строя не происходит. Группа преследователей имеет жесткую ориентацию в пространстве. Процесс похож на то, как будто на цель накидывают лассо.

Модель группового преследования нескольких целей с одновременным их достижением описывает итерационный процесс на основе метода следования прогнозируемым траекториям. Для того, чтобы обеспечить одновременное достижение цели модифицируются несколько параметров движения преследователей.

В предложенной нами программе изменяются такие величины, как модуль скорости движения преследователя, так и минимальный радиус кривизны траектории.

Математические модели группового преследования реализованы в системе компьютерной математики «MathCAD».

3.1 ПРЕСЛЕДОВАНИЕ ГРУППОЙ ОДИНОЧНОЙ ЦЕЛИ

Рассмотрим группу преследователей, которая будет догонять одиночную цель. В этой группе преследователей, каждый будет иметь свою цель и свою стратегию достижения цели.

Пусть группа преследователей состоит из четырех участников. Разберем в этой группе цели и стратегии для каждого участника.

Пусть цель в некоторый момент времени имеет положение T на плоскости и движется со скоростью V_T.

3.1.1 ЦЕЛЬ И СТРАТЕГИЯ ПЕРВОГО ОБЪЕКТА — ПРЕСЛЕДОВАТЕЛЯ

Преследователь P_1 со скоростью V_1 имеет целью просто догнать объект T, что означает совмещение координат P_1 и T с некоторой степенью точности $|P_1 - T| \leq \varepsilon$. В качестве показателя точности можно предложить $\varepsilon = |V_1| \cdot \Delta T$,

где ΔT - это период дискретизации по времени. Помимо этого объект P_1 обладает максимальной угловой скоростью вращения ω_1, что ограничивает радиус кривизны траектории движения $R_1 = \frac{|V_1|}{\omega_1}$.

Стратегия преследователя P_1 (Рисунок 3.1) заключается в том, что координаты точки T пересчитываются в систему координат (v_1, v_2) с началом координат в точке P_1:

$$v_1 = \frac{V_P}{|V_P|}$$
$$v_2 = \begin{bmatrix} -v_{1_y} \\ v_{1_x} \end{bmatrix}.$$

Там координаты точки T будут выглядеть так:

$$T_v = \begin{bmatrix} (T - P_1) \cdot v_1 \\ (T - P_1) \cdot v_2 \end{bmatrix}.$$

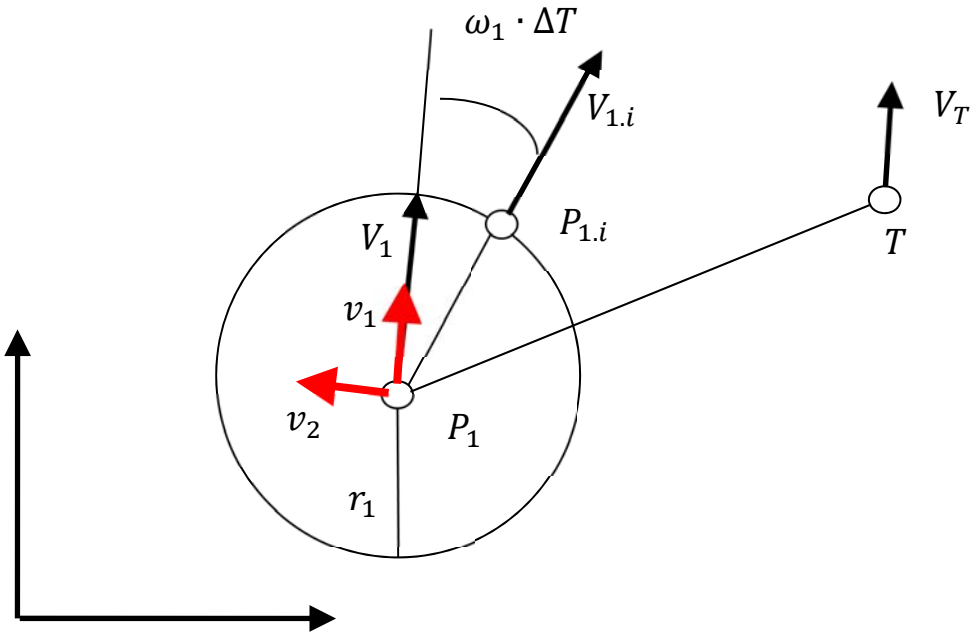


Рисунок. 3.1. Стратегия первого преследователя

Далее, совершаем анализ координат точки T_v на принадлежность к верхней или нижней полуплоскости в системе координат (v_1, v_2) с началом координат в точке P_1

$$P_{1.v} = \begin{matrix} if\ T_{v_y} \geq 0 \\ if\ T_{v_y} < 0 \end{matrix} \begin{bmatrix} |V_1| \cdot \Delta T \cdot cos(\omega_1 \cdot \Delta T) \\ |V_1| \cdot \Delta T \cdot sin(\omega_1 \cdot \Delta T) \end{bmatrix} \begin{bmatrix} |V_1| \cdot \Delta T \cdot cos(\omega_1 \cdot \Delta T) \\ -|V_1| \cdot \Delta T \cdot sin(\omega_1 \cdot \Delta T) \end{bmatrix}.$$

Необходимо, постоянно сравнивать значения углов $\omega_1 \cdot \Delta T$ и $\propto$, где $\propto$ - это угол между векторами $\overrightarrow{P_1 T}$ и V_1.

Если угол $\propto$ меньше, чем угол $\omega_P \cdot \Delta T$, тогда координаты точки $P_{1.v}$ будут выглядеть иначе:

$$P_{i.v} = \begin{matrix} if\ T_{v_y} \geq 0 \\ if\ T_{v_y} < 0 \end{matrix} \begin{bmatrix} |V_P| \cdot \Delta T \cdot cos(\propto) \\ |V_P| \cdot \Delta T \cdot sin(\propto) \end{bmatrix} \begin{bmatrix} |V_P| \cdot \Delta T \cdot cos(\propto) \\ -|V_P| \cdot \Delta T \cdot sin(\propto) \end{bmatrix}.$$

3.1.2 ЦЕЛИ И СТРАТЕГИИ ВТОРОГО И ТРЕТЬЕГО ОБЪЕКТОВ - ПРЕСЛЕДОВАТЕЛЕЙ

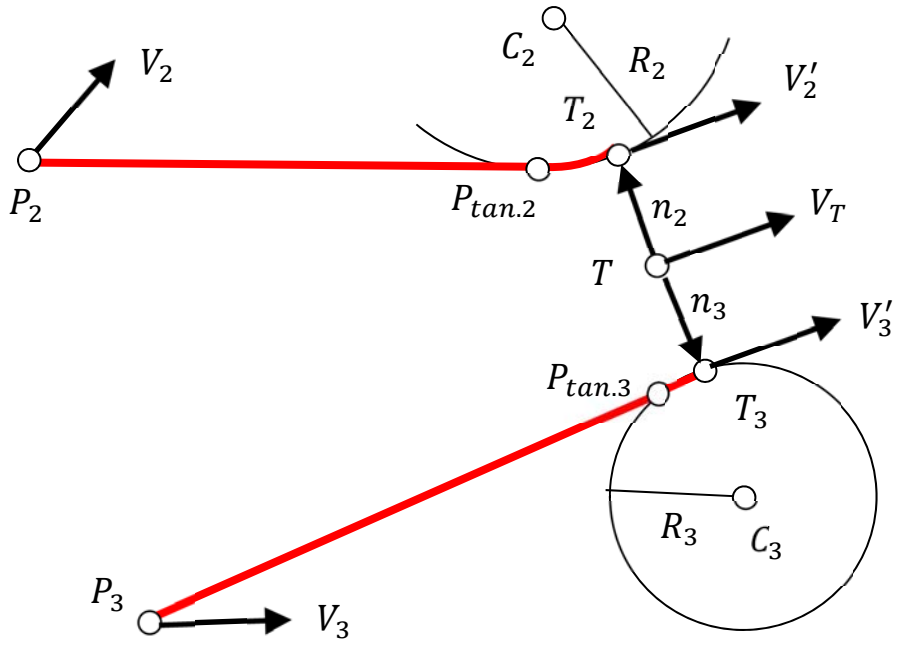


Рисунок. 3.2. Стратегии второго и третьего преследователей

Преследователи P_2 и P_3 совершают движение со скоростями V_2 и V_3, соответственно. Для объектов P_2 и P_3 целью является совмещение с определенной степенью точности ε не с точкой T, а с точками T_2 и T_3, соответственно (Рисунок 3.2).

Координаты точек T_2 и T_3 формируются следующим образом:

$$T_{2,3} = T + n_{2,3}$$

Векторы нормалей $n_{2,3} = \pm\frac{1}{|V_T|}\begin{bmatrix} -V_{T_y} \\ V_{T_x} \end{bmatrix} \cdot \Delta S_{2,3}$, где $\Delta S_{2,3}$ - это расстояния, на которые отстоят точки T_2 и T_3 от точки T.

Для траекторий объектов P_2 и P_3 выбираются такие условия, они подошли к точкам T_2 и T_3 с направлениями скоростей V_2' и V_3'. Радиусы кривизны траекторий не должны быть меньше $R_{2,3} = \frac{|V_{2,3}|}{\omega_{1,2}}$, где $\omega_{2,3}$ – максимальные угловые скорости вращения преследователей P_2 и P_3.

Моделируемая траектория в некоторый момент времени состоит из прямолинейного участка $[P_{2,3}, P_{tan.2,3}]$ и сегмента дуги $\overset{\frown}{P_{tan.2,3}, T_{2,3}}$.

На каждом этапе итераций объекты P_2 и P_3 совершают дискретное вращение и дискретное поступательное перемещение, чтобы выйти на моделируемые траектории.

В нашей тестовой программе, написанной по материалам данного параграфа, объекты P_2 и P_3, как только выходят на курс параллельный курсу T, начинают двигаться со скоростями, равными V_T.

3.1.3 ЦЕЛЬ И СТРАТЕГИЯ ЧЕТВЕРТОГО ПРЕСЛЕДОВАТЕЛЯ

Рассмотрим четвертого участника из группы преследователей. Если поведение первого участника можно квалифицировать как основного «загонщика». Поведение второго и третьего преследователей, квалифицировать как помощников, не дающих ускользнуть цели, то роль четвертого преследователя можно трактовать как игрока из «засады».

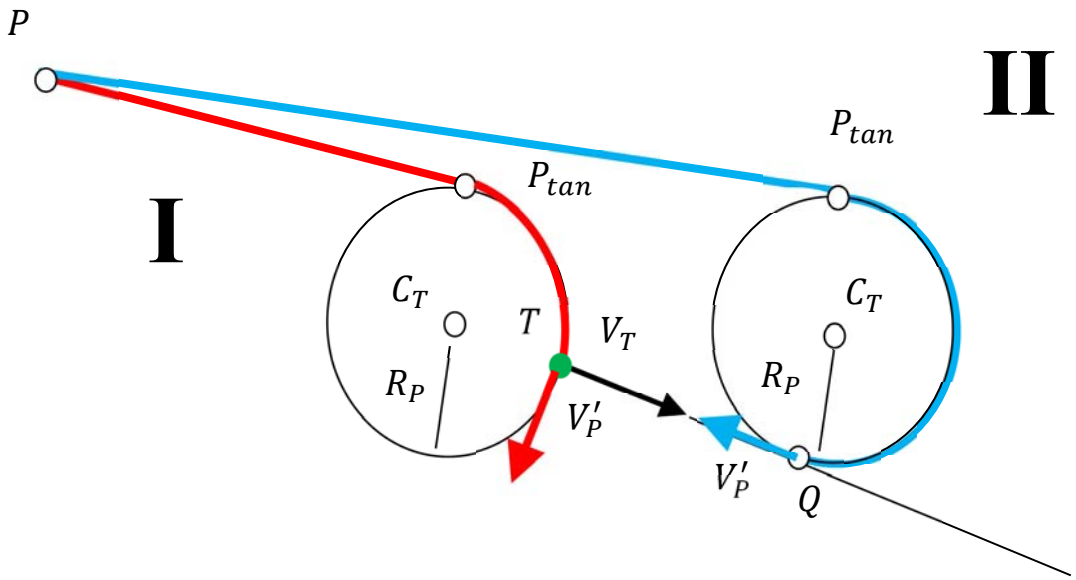


Рисунок 3.3. Стратегия преследователя из "засады"

На рисунке 3.3 показано два случая формирования траекторий четвертого преследователя. В первом случае траектория преследователя входит в точку положения непосредственно цели T, перпендикулярно ее скорости V_T. Во втором случае траектория преследователя входит в точку Q со скоростью противоположно направленной скорости цели V_T. Точка Q расположена на прямой из точки T с образующей V_T.

Точку Q можно расположить в любой точке плоскости, ничего нам этого не запрещает. Просто цель может быть не достигнута.

При достижении точки Q можно поменять стратегию преследователя. Допустим, сбросить скорость до 0 и ожидать приближения цели до расстояния меньше ε. Можно поменять стратегию при достижении точки Q на стратегию первого преследователя.

3.1.4 ЦЕЛЬ И СТРАТЕГИЯ ОБЪЕКТА ПРЕСЛЕДОВАНИЯ

Рассмотрим поведение объекта преследования. В нашей рассматриваемой модели целью объекта преследования выбрано уклонение от первого преследователя.

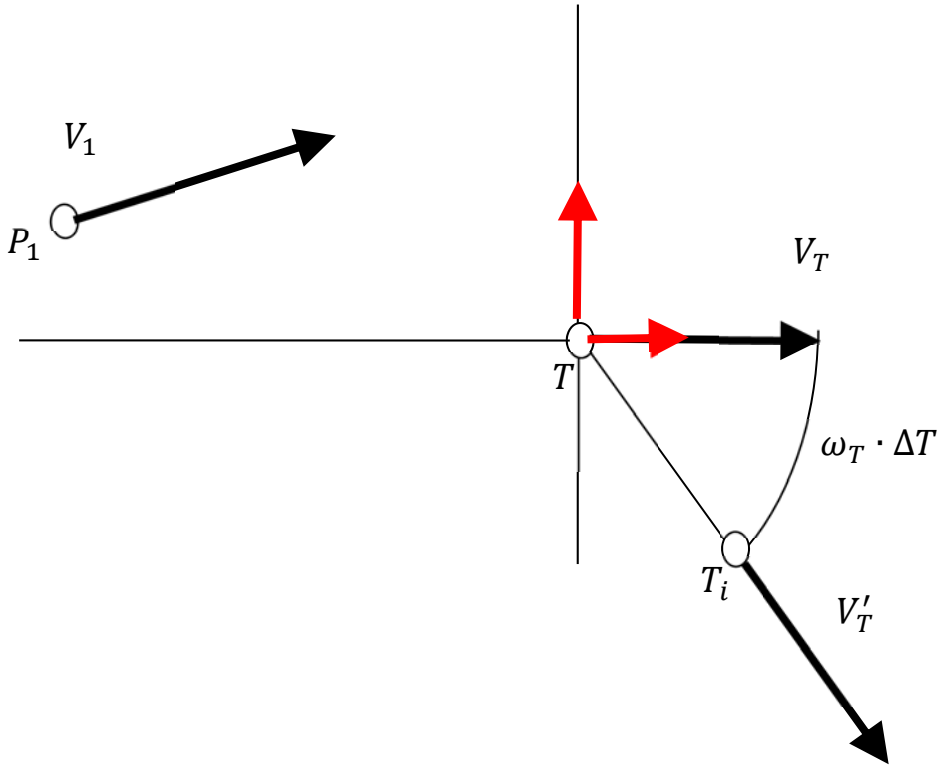


Рисунок 3.4. Стратегия объекта преследования

Рисунок 3.4 иллюстрирует стратегию преследуемого объекта T. На этом рисунке объект T движется со скоростью V_T и с угловой скоростью вращения ω_T за период дискретизации ΔT совершает поворот на угол $\omega_T \cdot \Delta T$ и перемещение на расстояние $|T_i - T| = |V_T| \cdot \Delta T$.

Направление вращения точки T зависит от того, в какой полуплоскости находится преследователь P_1.

Как альтернативную стратегию можно предложить стратегию, иллюстрация которой представлена на рисунке 3.5

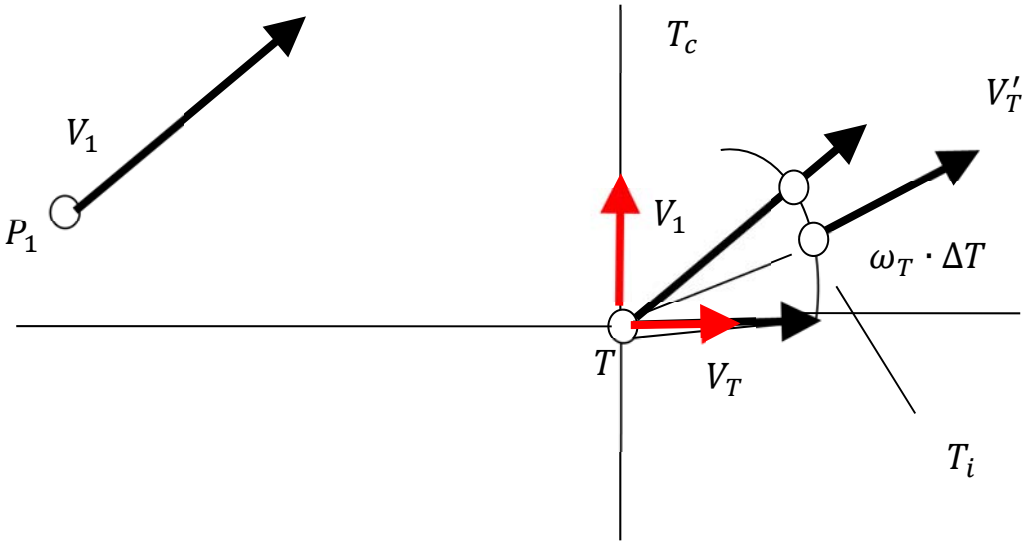


Рисунок 3.5. Дополнительная стратегия объекта преследования

На рисунке 3.5 показано, что объект преследования T стремится свою скорость V_T сделать параллельной вектору скорости преследователя V_1.

Когда преследователь находится далеко, то предпочтительней для цели использовать стратегию параллельных скоростей, как на рисунке 3.5. Когда преследователь подходит на дистанцию нескольких шагов, то есть для заключительного прыжка, для цели будет выгодна стратегия уклонения, как на рисунке 3.5.

3.1.5 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ ГРУППОВОГО ПРЕСЛЕДОВАНИЯ С РАЗЛИЧНЫМИ СТРАТЕГИЯМИ

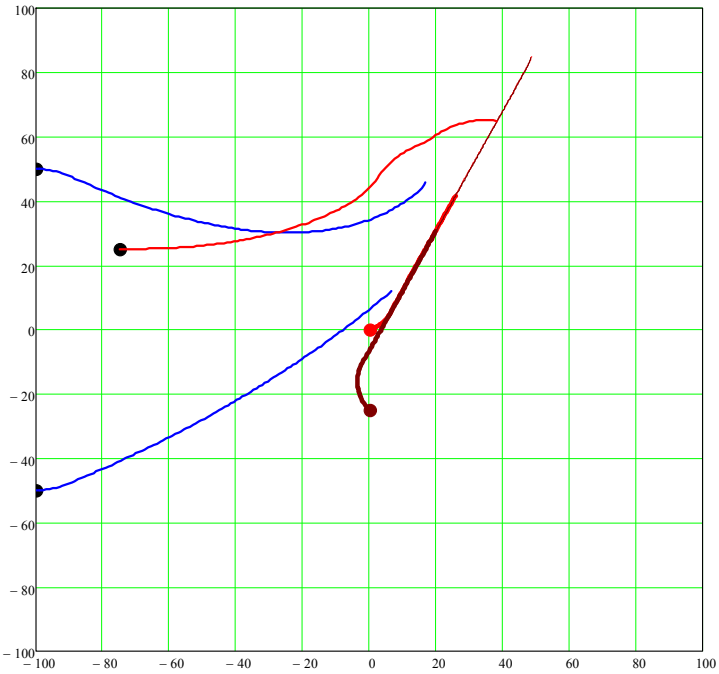


Рисунок 3.6. Результат моделирования группового преследования одиночной цели

На материалах, изложенных в данной главе, была написана тестовая программа в системе MathCAD, которая рассчитывает траектории группы из четырех преследователей и цели, уклоняющейся от них. У каждого участника геометрической модели своя цель и своя стратегия. На рисунке 3.6 показан скриншот с видео, где видно как один преследователь реализует погоню по следу. Два преследователя берут и сопровождают цель по параллельным траекториям. Один преследователь заходит перпендикулярно прогнозируемой траектории цели. В программе мы намеренно поменяли цель и стратегию четвертого преследователя, чтобы показать, что в рамках нашей программы достаточно просто, задав координаты точек входа и векторов входа в точки.

Рисунок 3.6 дополнен ссылкой на анимированное изображение [53], где размещено видео по результатам работы программы.

3.2 ПРЕСЛЕДОВАНИЕ ГРУППОЙ ОДНОЙ ЦЕЛИ С ЖЕСТКИМИ СВЯЗЯМИ

Рассмотрим группу преследователей состоящих из пяти участников. Пять участников – это цифра, взятая случайно, группа преследователей может составлять любое число.

Сценарий моделируемых событий таков. Один преследователь P, на рисунке 3.7 он выделен жирной точкой, преследует цель T со скоростью V_P. Преследователь P имеет угловую скорость вращения ω_P. Целью преследователя P является догнать цель T с использованием следующей стратегии.

Производится переход в локальную динамическую систему координат (v_1, v_1) с центром в точке P, $v_1 = \frac{V_P}{|V_P|}$, $v_2 = \begin{bmatrix} -v_{1_y} \\ v_{1_x} \end{bmatrix}$. В локальную динамическую систему координат (v_1, v_1) пересчитываются координаты цели T:

$$T_v = \begin{bmatrix} (T-P) \cdot v_1 \\ (T-P) \cdot v_2 \end{bmatrix}.$$

Далее, совершаем анализ координат точки T_v на принадлежность к верхней или нижней полуплоскости в системе координат (v_1, v_2) с началом координат в точке P, совершаем вращение на угол $\omega_P \cdot \Delta T$ и шаг на расстояние $|V_P| \cdot \Delta T$:

$$P_{i,v} = \begin{matrix} if\ T_{v_y} \geq 0 \begin{bmatrix} |V_P| \cdot \Delta T \cdot cos(\omega_P \cdot \Delta T) \\ |V_P| \cdot \Delta T \cdot sin(\omega_P \cdot \Delta T) \end{bmatrix} \\ if\ T_{v_y} < 0 \begin{bmatrix} |V_P| \cdot \Delta T \cdot cos(\omega_P \cdot \Delta T) \\ -|V_P| \cdot \Delta T \cdot sin(\omega_P \cdot \Delta T) \end{bmatrix} \end{matrix}.$$

В системе координат (v_1, v_2) с началом координат в точке P преследователь приобретает скорость

$$V_{P\,i,v} = |V_P| \cdot \frac{P_{i,v}}{|P_{i,v}|}$$

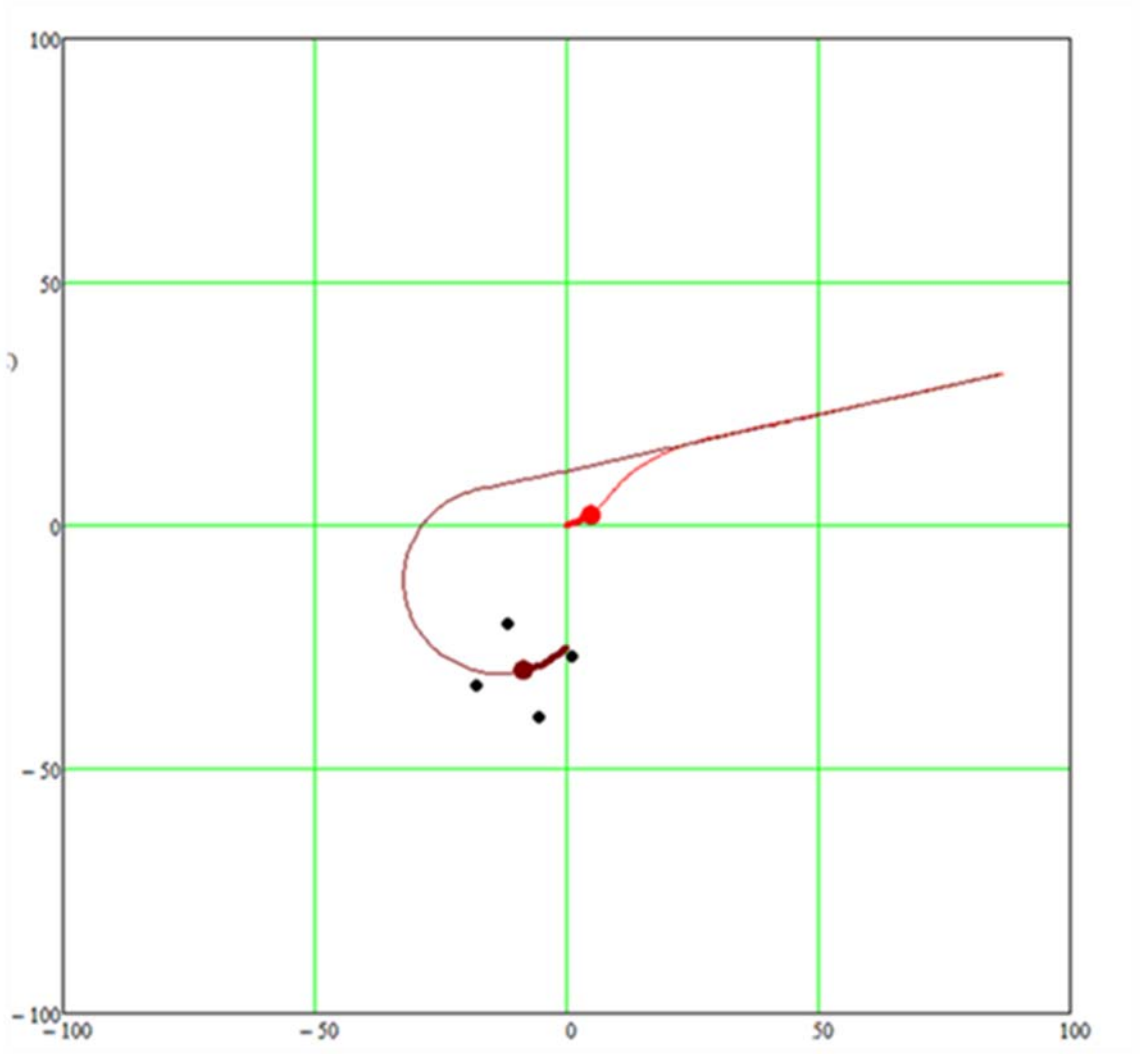


Рисунок 3.7. Групповое преследование с жесткими связями

В системе координат (v_1, v_2) группа сопровождения имеет координаты:

$$P_{1,v} = \Delta S \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}, P_{2,v} = S \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}, P_{3,v} = \Delta S \cdot \begin{bmatrix} -1 \\ 0 \end{bmatrix}, P_{4,v} = S \cdot \begin{bmatrix} 0 \\ -1 \end{bmatrix}.$$

В мировой системе координат группа сопровождения будет выглядеть так:

$$P_1 = \begin{bmatrix} P_{1,v} \cdot h_1 \\ P_{1,v} \cdot h_2 \end{bmatrix} + P, P_2 = \begin{bmatrix} P_{2,v} \cdot h_1 \\ P_{2,v} \cdot h_2 \end{bmatrix} + P$$
$$P_3 = \begin{bmatrix} P_{3,v} \cdot h_1 \\ P_{3,v} \cdot h_2 \end{bmatrix} + P, P_4 = \begin{bmatrix} P_{4,v} \cdot h_1 \\ P_{4,v} \cdot h_2 \end{bmatrix} + P.$$

Координаты преследователя P в мировой системе координат после совершения шага итерации будут выглядеть так:

$$P_i = \begin{bmatrix} P_{i,v} \cdot h_1 \\ P_{i,v} \cdot h_2 \end{bmatrix} + P.$$

Направление и модуль скорости в мировой системе координат будут такими:

$$V_{P\,i} = \begin{bmatrix} V_{P\,i,v} \cdot h_1 \\ V_{P\,i,v} \cdot h_2 \end{bmatrix}.$$

Где h_1 и h_2 - это разложение векторов H_1 и H_2 мировой системы координат по базису (v_1, v_2) преследователя.

В модели, приведенной в данном параграфе, поведение цели T определяется поведением преследователя P. Цель уклоняется от преследователя после анализа в своей динамической системе координат преследователя:

$$T_{i,v} = \begin{matrix} if\ P_{v_y} \geq 0 \begin{bmatrix} |V_T| \cdot \Delta T \cdot cos(\omega_T \cdot \Delta T) \\ -|V_T| \cdot \Delta T \cdot sin(\omega_T \cdot \Delta T) \end{bmatrix} \\ if\ P_{v_y} < 0 \begin{bmatrix} |V_T| \cdot \Delta T \cdot cos(\omega_T \cdot \Delta T) \\ |V_T| \cdot \Delta T \cdot sin(\omega_T \cdot \Delta T) \end{bmatrix} \end{matrix}.$$

Скорость после совершения шага итерации, системе координат цели будет:

$$V_{T\,i,v} = |V_T| \cdot \frac{T_{i,v}}{|T_{i,v}|}.$$

Перевод в мировую систему координат из системы координат цели будет следующим:

$$T_i = \begin{bmatrix} T_{i,v} \cdot h_1 \\ T_{i,v} \cdot h_2 \end{bmatrix} + T$$
$$V_{T\,i} = \begin{bmatrix} V_{T\,i,v} \cdot h_1 \\ V_{T\,i,v} \cdot h_2 \end{bmatrix}.$$

Где h_1 и h_2 - это разложение векторов H_1 и H_2 мировой системы координат по базису (v_1, v_2) цели.

Рисунок 3.7 дополнен ссылкой на анимированное изображение [54], изготовленное по результатам работы тестовой программы.

3.3 ОДНОВРЕМЕННОЕ ДОСТИЖЕНИЕ ГРУППОЙ ПРЕСЛЕДОВАТЕЛЕЙ ГРУППЫ ЦЕЛЕЙ

В данном параграфе рассматривается кинематическая модель преследования группой нескольких цели методом параллельного сближения. Модель основывается на том, что преследователи стараются придерживаться заранее спроектированных траекторий. Траектории преследователей имеют ограничения по кривизне. Начальные направления скоростей преследователей имеют произвольный характер, что вносит изменения в известный метод параллельного сближения. В нашей модели цели достигаются преследователями одновременно. Это происходит из-за изменения длин прогнозируемых траекторий таким образом, чтобы синхронизировать время достижения цели. Изменение длин прогнозируемых траекторий происходит за счет увеличения радиуса кривизны на первоначальном участке траектории.

3.3.1 МЕТОД ПАРАЛЛЕЛЬНОГО СБЛИЖЕНИЯ

Одной из особенностью метода параллельного сближения на плоскости является то, что скорость преследователя P_i в некоторый момент времени направлена в точку на окружности Аполлония. На рисунке 3.8 это точка K_i, а точка T_i – положение цели в данный момент времени.

Итерационная схема метода параллельного сближения представлена на рисунке 1. Координаты преследователя P_i будут рассчитываться таким образом:

$$P_{i+1} = P_i + V_P \cdot \frac{P_i K_i}{|P_i K_i|} \cdot \Delta T, T_{i+1} = T_i + V_T \cdot \frac{T_i K_i}{|T_i K_i|} \cdot \Delta T.$$

Где ΔT - дискретный временной промежуток.

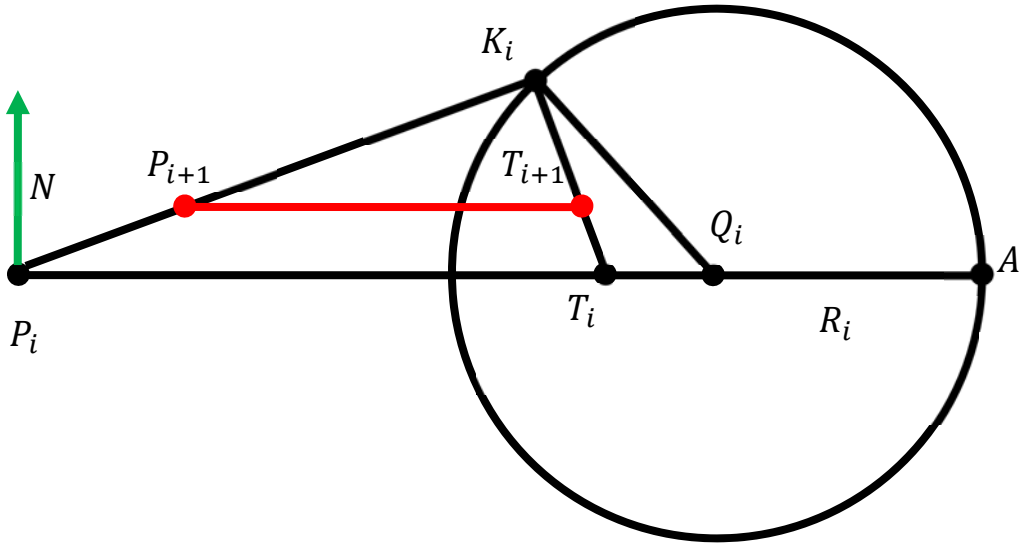


Рисунок 3.8. Метод параллельного преследования

Радиус R_i и центр окружности Аполлония Q_i рассчитываются следующим образом:

$$R_i = \frac{V_T^2}{V_P^2 - V_T^2} \cdot |T_i - P_i|, Q_i = T_i + \frac{V_T^2}{V_P^2 - V_T^2} \cdot (T_i - P_i).$$

Координаты точки K_i есть результат решения системы уравнений относительно непрерывного параметра t:

$$\begin{cases} (K_i - Q_i)^2 = R_i^2 \\ K_i = T_i + V_T \cdot \frac{T_{i+1} - T_i}{|T_{i+1} - T_i|} \cdot t \end{cases}.$$

3.3.2 ПРЕСЛЕДОВАНИЕ ОДНОЙ ЦЕЛИ ДВУМЯ ПРЕСЛЕДОВАТЕЛЯМИ

Из описания метода параллельного сближения видно, что начальная скорость преследователя не может иметь произвольного направления.

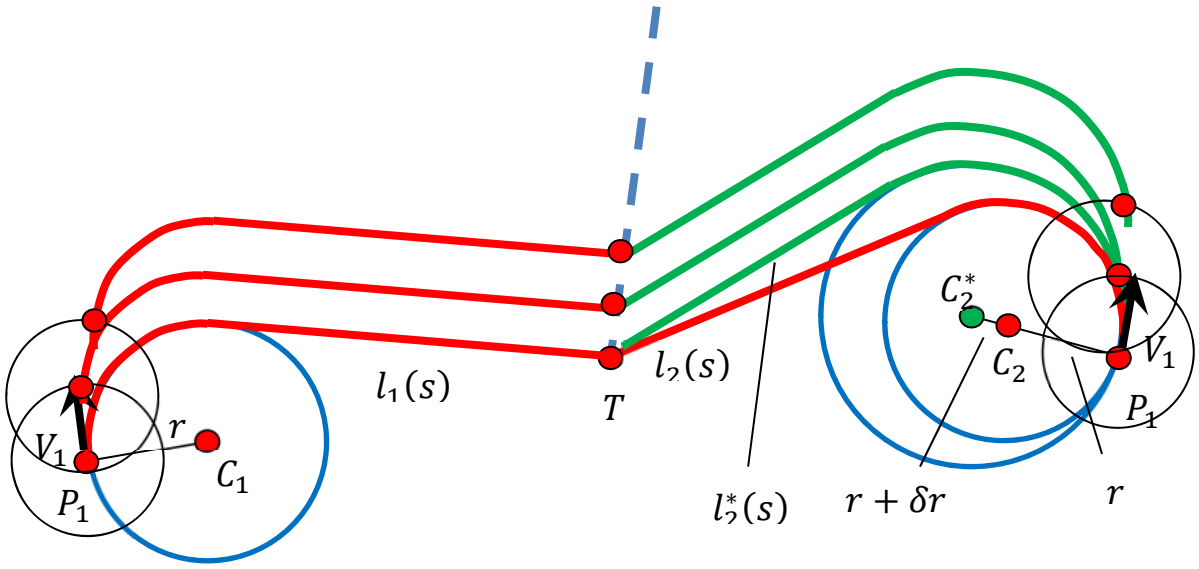


Рисунок 3.9. Преследование одной цели двумя преследователями

В данном параграфе мы хотим реализовать метод, близкий к методу параллельного сближения. На начальном этапе решения рассмотрим двух преследователей P_1, P_2, скорости которых V_1, V_2 направлены произвольно (Рисунок 3.9). Цель T движется прямолинейно и равномерно.

Радиус кривизны траекторий движения преследователей не может меньше определенной величины. Поэтому мы формируем однопараметрические множества составных линий, являющихся аналогом линии визирования (P_iT_i) (Рисунок 3.8). В нашем случае, это будут составные линии, соединяющих точки P_1, P_2 с точкой T (Рисунок 3.9), состоящие из сегмента дуги и прямолинейного отрезка.

Допустим, преследователь P_2 при расчете траектории для достижения цели T имеет меньшее время. Мы можем изменять радиус кривизны прогнозируемой траектории преследователя P_2 в сторону увеличения (Рисунок 3.9), чтобы достичь одновременного достижения цели совместно с P_1.

Если мы добавим еще одну цель и еще одного преследователя (Рисунок 3.10), то в этом случае, эталоном выбирается преследователь имеющий наибольшее время достижения своей цели при предварительном расчете.

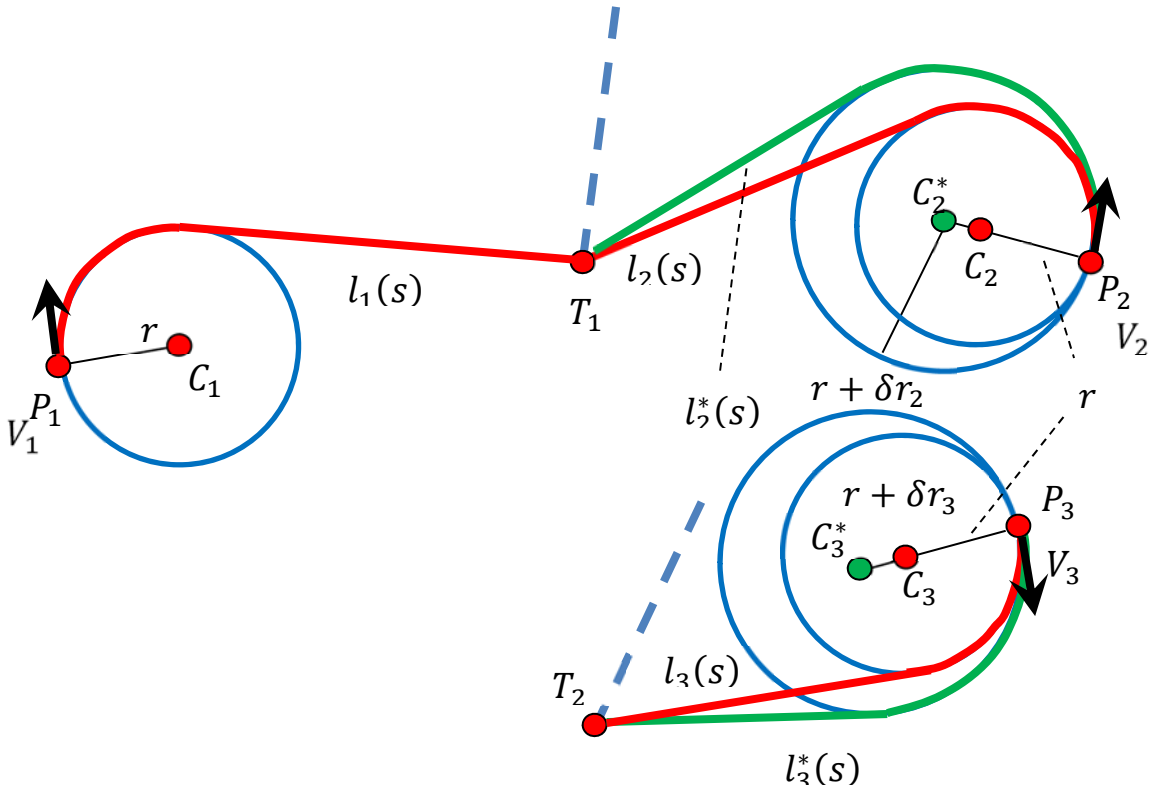


Рисунок 3.10. Групповое преследование группы целей

На рисунке 3.10 показано, преследователи P_1, P_2 преследуют цель T_1, а преследователь P_3 преследует цель T_2.

Нами написана тестовая программа, где преследователи P_2 и P_3 изменяют радиус кривизны прогнозируемых траекторий, подстраиваясь под время достижения преследователем P_1 цели T_1.

3.3.3 МОДЕЛИРОВАНИЕ СОСТАВНОЙ КРИВОЙ

Для решения поставленной задачи, мы должны смоделировать для каждого из преследователей составную кривую (Рисунок 3.11).

Так как в нашей модели существуют ограничения на кривизну траектории всех участников задачи преследования, то наш преследователь P в прогнозируемой траектории (Рисунок 3.11) пройдет по дуге $\overset{\frown}{PP_t}$, потом выйдет на прямолинейный участок $[P_tT]$ до цели T.

Радиус кривизны r окружности (C,r) в нашей модели считается заданным и может изменяться только в сторону увеличения.

Центр C окружности (C, r) удовлетворяет системе уравнений:

$$|C - P| = r$$
$$V \cdot (C - P) = 0$$

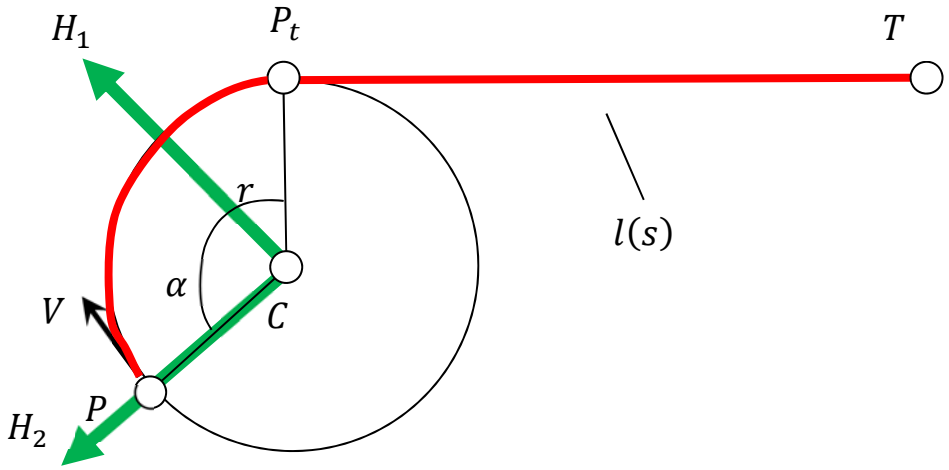


Рисунок 3.11. Моделирование множества параллельных линий

В локальной системе координат (H_1, H_2) с центром в точке C уравнение дуги $\widetilde{PP_t}$ будет:

$$L_{circle}(\alpha)^* = r \cdot \begin{bmatrix} \cos(\frac{\pi}{2} - \alpha) \\ sin\left(\frac{\pi}{2} - \alpha\right) \end{bmatrix}.$$

Где α принимает значения от 0 до $\arccos(\frac{(P-C)\cdot(P_t-C)}{|P-C|\cdot|P_t-C|})$. Базисные векторы (H_1, H_2) равны:

$$H_1 = \frac{V}{|V|}, H_2 = \frac{P - C}{|P - C|}.$$

Перевод в мировую систему координат линии $L_{circle}(\alpha)$ будет таким:

$$L_{circle}(\alpha) = \begin{bmatrix} L_{circle}(\alpha)^* \cdot E_1^* \\ L_{circle}(\alpha)^* \cdot E_2^* \end{bmatrix} + C$$
$$E_1^* = \begin{bmatrix} E_1 \cdot H_1 \\ E_1 \cdot H_2 \end{bmatrix}, E_2^* = \begin{bmatrix} E_2 \cdot H_1 \\ E_2 \cdot H_2 \end{bmatrix}$$
$$E_1, E_2 - \text{базисные векторы МСК}$$

Уравнение для прямолинейного участка $[P_t T]$ представим в виде:

$$L_{line}(\varepsilon) = (1 - \varepsilon) \cdot P_t + \varepsilon \cdot T.$$

Полученные сегменты линий $L_{circle}(\alpha)$ и $L_{line}(\varepsilon)$ необходимо объединить в одну составную линию и выполнить параметризацию от длины дуги.

В тестовой программе, написанной по материалам статьи, мы получили объединенные массивы координат $\{X_i, Y_i\}, i \in 0..N$ нашей составной кривой. Введем формальный параметр τ , который непрерывно пробегает значения от 0 до N.

После процедуры кубической сплайн-интерполяции будем иметь непрерывные координатные функции $X(\tau)$ и $Y(\tau)$ от параметра τ.

3.3.4 РАСЧЕТ ИТЕРАЦИОННОГО ПРОЦЕССА

Из уравнения для полного дифференциала длины дуги $ds^2 = dX^2 + dY^2$ мы получим дифференциальное уравнение первого порядка для дальнейшей передачи во встроенные решатели задачи Коши:

$$D(\tau, s) = \frac{d\tau}{ds} = \frac{1}{\sqrt{\frac{dX}{d\tau}^2 + \frac{dY}{d\tau}^2}}, \tau(0) = 0.$$

Таким образом, нами получены зависимости $X(s)$ и $Y(s)$ от параметра длины дуги. Если параметр длину будет удовлетворять соотношению $s = V \cdot t$, где t - это реальное время, то мы получим зависимости $X(t)$ и $Y(t)$, являющимися координатными функциями базовой линии $l(t)$.

Составную линию, которая соединяет преследователя и цель, в момент начала преследования назовем базовой линией.

Для того чтобы выделить линию, соответствующую положению цели $T(t)$, необходимо к базовому уравнению линии $l(t)$ прибавить вектор $T(t) - T(0)$ (Рисунок 3.9).

Если у нас возникает необходимость увеличения длины базовой линии, то мы увеличиваем радиус минимальной кривизны. Вообще, длина базовой линии зависит от следующих параметров: координат цели T, координат преследователя P, вектора скорости V преследователя и радиуса r минимального радиуса кривизны (Рисунки 3.9, 3.10).

Пусть наш преследователь P имеет модуль скорости V_P. В нашей задаче в момент t_i рассчитаны координаты точек преследователя P_i и цели T_i. Рассчитана от своей длины дуги уравнение линии прогнозируемого движения $l_i(s)$.

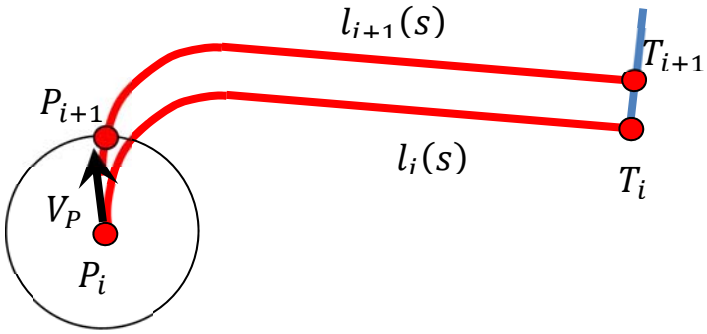


Рисунок 3.12. Расчет следующего шага преследователя

В момент времени t_{i+1} является известным координаты цели T_{i+1}. Тогда линия параллельного сдвига $l_{i+1}(s)$ вычисляется так:

$$l_{i+1}(s) = l_i(s) + (T_{i+1} - T_i).$$

Точка следующего шага преследователя P_{i+1} есть точка перечения линии $l_{i+1}(s)$ и окружности радиуса $V_P \cdot (t_{i+1} - t_i)$ с центром в точке P_i (Рисунок 3.12).

В тестовой программе рассчитываются сначала ориентировочные промежутки времени достижения преследователями своих целей. Затем, выбирается наибольший в качестве эталонного. Затем в цикле делаются малые приращения радиуса допустимой кривизны базовых траекторий δr (Рисунки 3.9, 3.10) до тех, пока не произойдет выравнивание значений временных промежутков.

3.3.5 РЕЗУЛЬТАТЫ МОДЕЛИРОВАНИЯ

По материалам статьи разработана тестовая программа, в которой два преследователя с первоначальными произвольными направлениями скоростей начинают преследовать цель, движущуюся прямолинейно с постоянной скоростью.

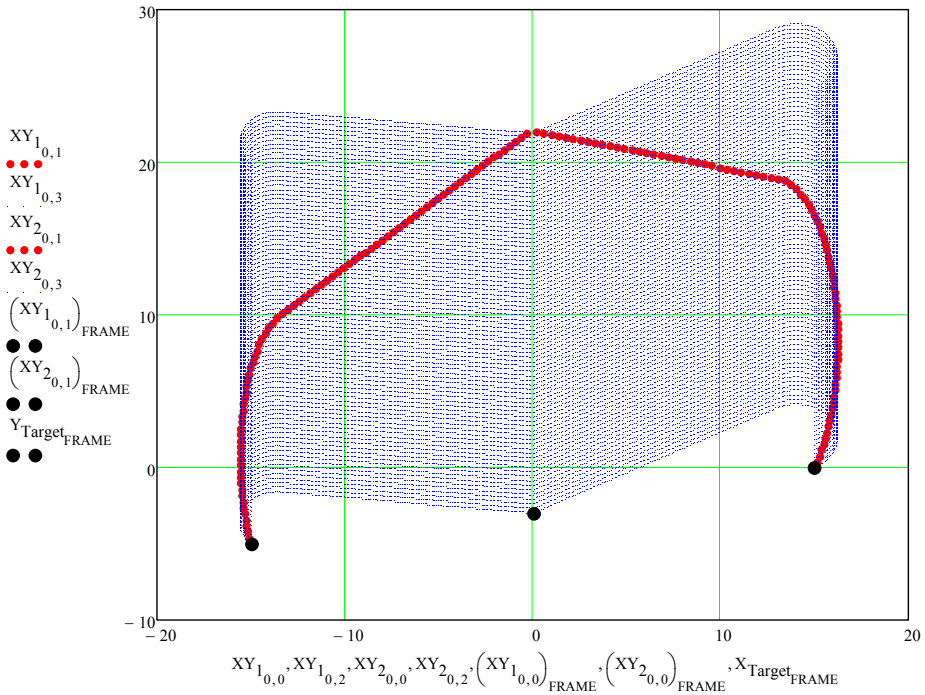


Рисунок 3.13. Преследование одной цели двумя преследователями

На рисунке 3.13 представлен первый кадр работы программы. Рисунок 3.13 дополнен ссылкой на анимированное изображение [55].

Отметим, что с текстом программы можно ознакомиться на сайте автора. Также, авторами была написана программа преследования группой из трех преследователей группы из двух целей. Достижение целей происходит одновременно.

На рисунке 3.14 представлен первый кадр анимированного изображения [56] модели одновременного достижения цели. На рисунке 3.15 размещен первый кадр анимированного изображения, где показан итерационный процесс преследования без прогнозируемых линий движения траекторий преследователей.

Рисунок 3.15 также дополнен ссылкой на анимированное изображение [57].

Изложенная в данном параграфе, математическая модель задачи преследования предполагает, что траектории преследователей в определенный момент времени рассчитываются так, как будто цели движутся прямолинейно и равномерно. Но ничего не мешает нам сделать расчеты

прогнозируемых траекторий для иных направлений движения целей, с иными скоростями.

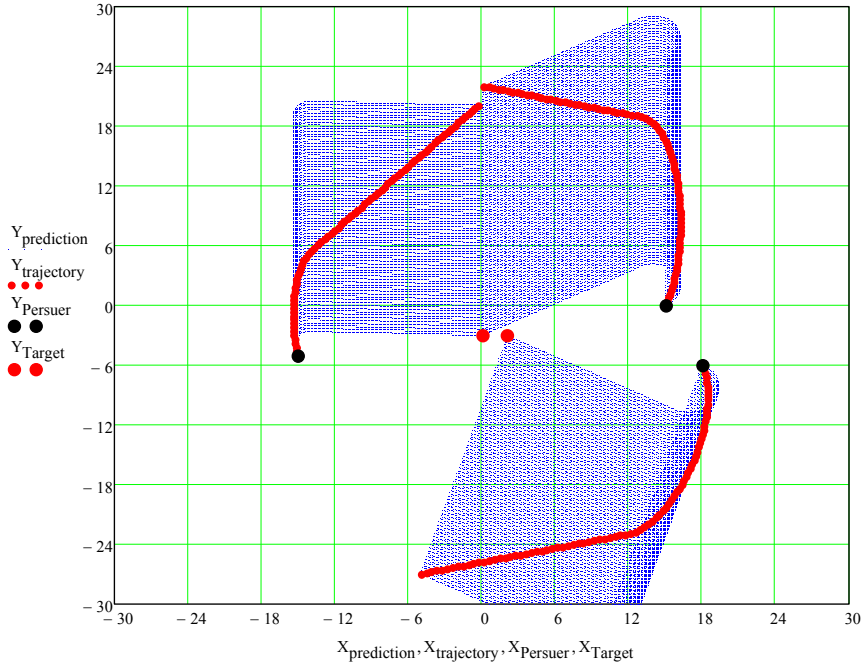


Рисунок 3.14. Преследование двух целей группой из трех преследователей

Мы решили не делать изменения направлений в тестовой программе, чтобы не добавлять дополнительный вложенный расчетный цикл в итерационный процесс.

Основным в предлагаемой модели является наложение ограничений на кривизну траекторий преследователей. Что является характерным для объектов, не имеющих возможности изменять направление скорости мгновенно.

Важным вопросом в представленной модели является распределение преследователей по целям. В тестовой программе распределение производилось вручную. Хотелось бы иметь автоматизированное распределение по целям, без участия оператора. Основным в разработанной модели является расчет и модификация базовых линий для синхронизации с максимальным временем достижения одного из преследователей своей цели.

Если является возможным произвести моделирование процесса одновременного достижения целей, то мы сможем изменить модель, где достижение целей будет происходить по таймеру.

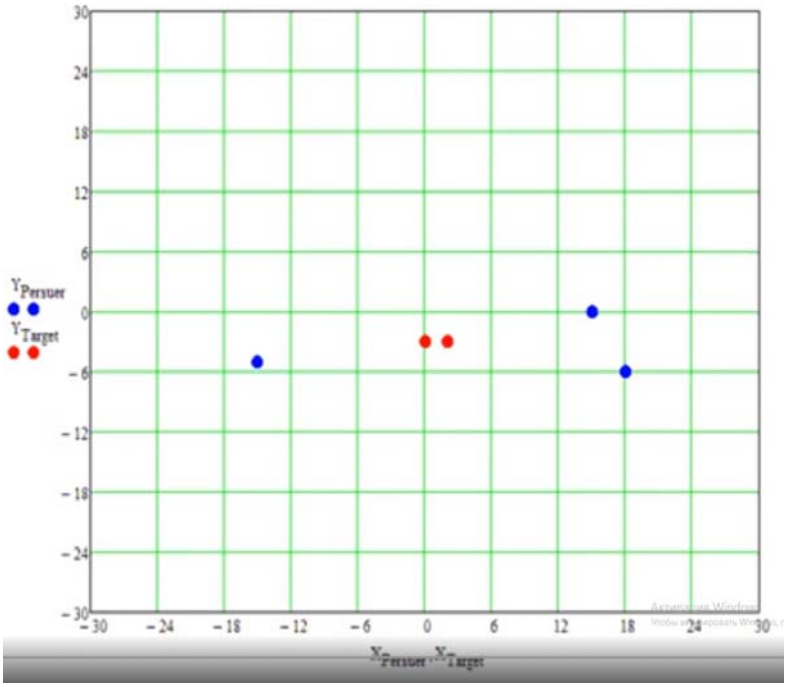


Рисунок 3.15. Преследование двух целей группой из трех преследователей без вспомогательных линий

3.4 КОММЕНТАРИИ К ПРОГРАММЕ ГРУППОВОГО ПРЕСЛЕДОВАНИЯ С РАЗЛИЧНЫМИ СТРАТЕГИЯМИ

В этом параграфе мы подробно разберем код, написанной нами программы, которая описывает преследование группой одиночной цели.

Расстановка всех участников процесса преследования показана на рисунке 3.16.

$Rabbit$ - это цель, $Fox, Wolf\ 1, Wolf\ 2, Wolf\ 3$ - это преследователи.

Задачей, которой хочет добиться преследователь Fox, является то, чтобы расстояние между движущимися точками Fox и $Rabbit$, было меньше заданного порогового значения ΔR_0, используя при этом стратегию погони. Суть стратегии погони заключается в корректировке направления движения преследователя Fox. Преследователь Fox будет постоянно стремиться совместить вектор своей скорости совместить с линией визирования, соединяющей объекты Fox и $Rabbit$.

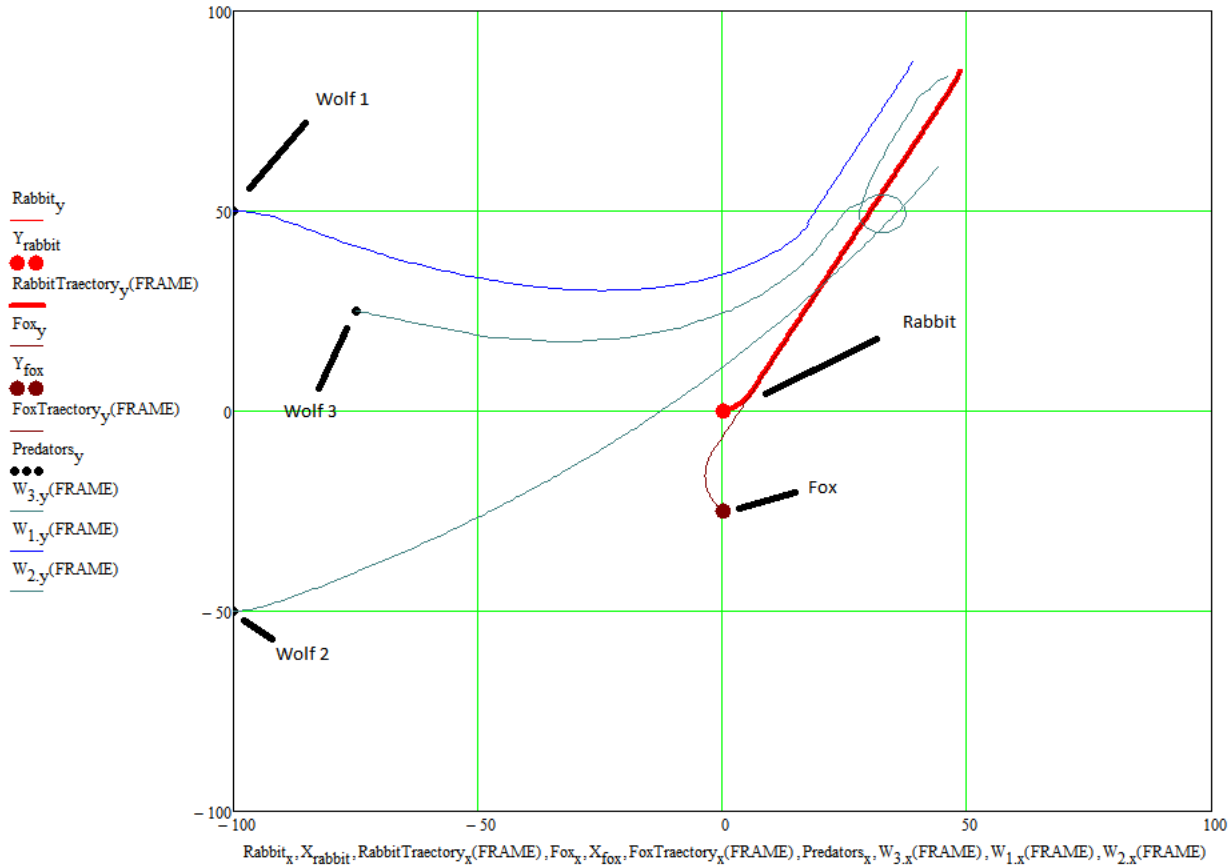


Рисунок 3.16. Положения цели и преследователей в момент начала преследования.

Задачи, которые хотят выполнить преследователи $Wolf\ 1$ и $Wolf\ 2$, являются точки плоскости, которые движутся параллельно самой цели преследования $Rabbit$ на некотором расстоянии от нее. Если они достигают этих, то движутся параллельно со скоростью цели.

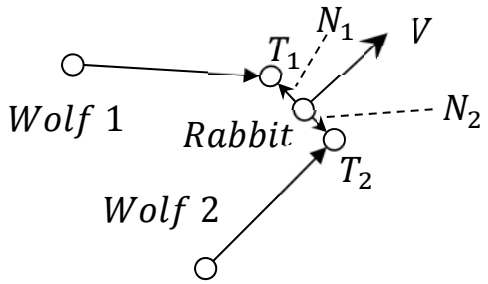


Рисунок 3.17. Задачи для $Wolf\ 1$ и $Wolf\ 2$

Мы в тестовой программе упростили алгоритм, изложенный в параграфе 3.1. На рисунке 3.17, что для преследователей $Wolf\ 1$ и $Wolf\ 2$ задачами, которые надо выполнить, служит достижение точек T_1 и T_2.

Векторы N_1 и N_2, которые соединяют точки T_1 и T_2 с точкой $Rabbit$, перпендикулярны вектору скорости V.

Задачу, которую хочет выполнить преследователь $Wolf\ 3$, состоит в том, чтобы настичь объект преследования $Rabbit$ под определенным углом. В нашей тестовой программе $Wolf\ 3$ настигает точку $Rabbit$ под прямым углом.

$$V_{rabbit} := 8 \quad \lambda_{rabbit} := \frac{\pi}{12} \quad T_{rabbit} := 8 \quad \omega_{rabbit} := \frac{2\pi}{T_{rabbit}}$$

Задаем параметры объекта $Rabbit$. Задается абсолютная величина скорости, угол наклона к оси абсцисс в мировой системе координат, вводится период вращения вектора скорости, как модель инертности объекта $Rabbit$, в следствие, рассчитывается угловая частота вращения.

$$V_{fox} := 10 \quad \lambda_{fox} := \frac{3\pi}{4} \quad T_{fox} := 8 \quad \omega_{fox} := \frac{2\pi}{T_{fox}}$$

Задаем параметры объекта Fox. Задается абсолютная величина скорости, угол наклона к оси абсцисс в мировой системе координат, вводится период вращения вектора скорости, как модель инертности объекта Fox, в следствие, рассчитывается угловая частота вращения.

$$\Delta T := \frac{25.3}{500} \qquad \Delta T = 0.051$$

Определяем период дискретизации итерационного процесса.

$$\beta_0 := \omega_{fox} \cdot \Delta T \quad \Delta R_0 := V_{fox} \cdot \Delta T \quad \Delta R_0 = 0.506$$

Определение угла поворота объекта $Rabbit$ за период дискретизации итерационного процесса и шага. В нашей модели шаг ΔR_0 принят за пороговое значение. Если расстояние между движущимися точками Fox и $Rabbit$, было меньше заданного порогового значения ΔR_0, то будет считаться, что объект Fox настиг объект $Rabbit$.

$$R_{Rabbit0} := \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad V_{Rabbit0} := V_{rabbit} \cdot \begin{pmatrix} \cos(\lambda_{rabbit}) \\ \sin(\lambda_{rabbit}) \end{pmatrix}$$

Определение начального положения объекта $Rabbit$ и вектора его скорости.

$$R_{Fox0} := \begin{pmatrix} 0 \\ -25 \end{pmatrix} \quad V_{Fox0} := V_{fox} \cdot \begin{pmatrix} \cos(\lambda_{fox}) \\ \sin(\lambda_{fox}) \end{pmatrix}$$

Определение начального положения объекта Fox и вектора его скорости.

$$V_{wolf.1.0} := 20 \quad \lambda_{wolf.1} := 0 \quad T_{wolf.1} := 8 \quad \omega_{wolf.1} := \frac{2\pi}{T_{wolf.1}}$$

Задаем параметры объекта $Wolf\ 1$. Задается абсолютная величина скорости, угол наклона к оси абсцисс в мировой системе координат, вводится период вращения вектора скорости, как модель инертности объекта $Wolf\ 1$, в следствие, рассчитывается угловая частота вращения.

$$R_{wolf.1} := \begin{pmatrix} -100 \\ 50 \end{pmatrix} \quad V_{wolf.1} := V_{wolf.1.0} \cdot \begin{pmatrix} \cos(\lambda_{wolf.1}) \\ \sin(\lambda_{wolf.1}) \end{pmatrix}$$

Определение начального положения объекта $Wolf\ 1$ и вектора его скорости.

$$V_{wolf.2.0} := 15 \quad \lambda_{wolf.2} := 0 \quad T_{wolf.2} := 8 \quad \omega_{wolf.2} := \frac{2\pi}{T_{wolf.2}}$$

Задаем параметры объекта $Wolf\ 2$. Задается абсолютная величина скорости, угол наклона к оси абсцисс в мировой системе координат, вводится период вращения вектора скорости, как модель инертности объекта $Wolf\ 2$, в следствие, рассчитывается угловая частота вращения.

$$R_{wolf.2} := \begin{pmatrix} -100 \\ -50 \end{pmatrix} \quad V_{wolf.2} := V_{wolf.2.0} \cdot \begin{pmatrix} \cos(\lambda_{wolf.2}) \\ \sin(\lambda_{wolf.2}) \end{pmatrix}$$

Определение начального положения объекта $Wolf\ 2$ и вектора его скорости.

$$V_{wolf.3.0} := 20 \quad \lambda_{wolf.3} := \frac{\pi}{2} \quad T_{wolf.3} := 2 \quad \omega_{wolf.3} := \frac{2\pi}{T_{wolf.3}}$$

Задаем параметры объекта $Wolf\ 3$. Задается абсолютная величина скорости, угол наклона к оси абсцисс в мировой системе координат, вводится период вращения вектора скорости, как модель инертности объекта $Wolf\ 3$, в следствие, рассчитывается угловая частота вращения.

$$R_{wolf.3} := \begin{pmatrix} -75 \\ 25 \end{pmatrix} \quad V_{wolf.3} := V_{wolf.2.0} \cdot \begin{pmatrix} \cos(\lambda_{wolf.2}) \\ \sin(\lambda_{wolf.2}) \end{pmatrix}$$

Определение начального положения объекта $Wolf\ 3$ и вектора его скорости.

Далее, нами была написана функция, выполняющая поворот на угол $\omega \cdot \Delta T$. Геометрический смысл этой функции отображен на рисунке 3.18.

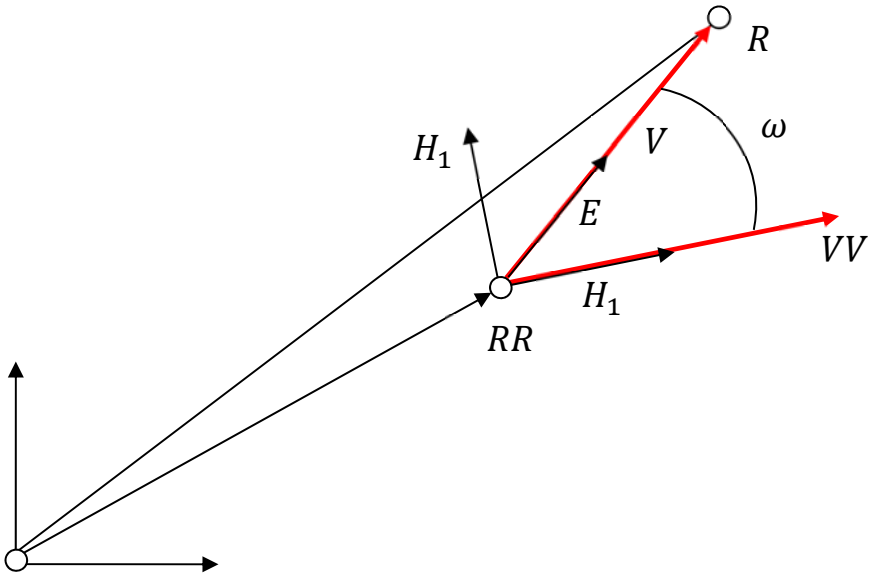


Рисунок 3.18. Поворот вектора угол

Входными параметрами функции $Predator\_Vector_1$ служат вектор скорости VV, вектор положения точки RR, угловая частота вращения ω.

Выходными параметрами функции $Predator\_Vector_1$ являются вектор поворота E в локальной системе координат (H_1, RR, H_2), базисные векторы H, выраженные в мировой системе координат. Вектор точки R, в которую преобразуется точка RR после поворота на угол $\omega \cdot \Delta T$ и смещения на расстояние $|VV| \cdot \Delta T$, выраженный в мировой системе координат. Новый вектор скорости V, выраженный в мировой системе координат.

$$
\begin{aligned}
&\text{Predator\_Vector}_1(VV, RR, \omega) := \\
&\quad E \leftarrow (\cos(\omega \cdot \Delta T) \quad \sin(\omega \cdot \Delta T)) \\
&\quad H \leftarrow \frac{1}{\sqrt{(VV_0)^2 + (VV_1)^2}} \cdot \begin{pmatrix} VV_0 & VV_1 \\ -VV_1 & VV_0 \end{pmatrix} \\
&\quad R \leftarrow \begin{bmatrix} \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \Delta T \cdot E \cdot H^{\langle 0 \rangle} + RR_0 \\ \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \Delta T \cdot E \cdot H^{\langle 1 \rangle} + RR_1 \end{bmatrix} \\
&\quad V \leftarrow \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \begin{pmatrix} E \cdot H^{\langle 0 \rangle} \\ E \cdot H^{\langle 1 \rangle} \end{pmatrix} \\
&\quad (E \;\; H \;\; R \;\; V)
\end{aligned}
$$

В функции $Predator\_Vector_1$ поворот производится против часовой стрелки. В функции $Predator\_Vector_2$ поворот производится по часовой стрелке.

$$\text{Predator\_Vector}_2(VV, RR, \omega) := \left| \begin{array}{l} E \leftarrow (\cos(\omega \cdot \Delta T) \quad -\sin(\omega \cdot \Delta T)) \\ H \leftarrow \dfrac{1}{\sqrt{(VV_0)^2 + (VV_1)^2}} \cdot \begin{pmatrix} VV_0 & VV_1 \\ -VV_1 & VV_0 \end{pmatrix} \\ R \leftarrow \begin{bmatrix} \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \Delta T \cdot E \cdot H^{\langle 0 \rangle} + RR_0 \\ \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \Delta T \cdot E \cdot H^{\langle 1 \rangle} + RR_1 \end{bmatrix} \\ V \leftarrow \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \begin{pmatrix} E \cdot H^{\langle 0 \rangle} \\ E \cdot H^{\langle 1 \rangle} \end{pmatrix} \\ (E \;\; H \;\; R \;\; V) \end{array} \right.$$

Функции $Predator\_Vector_1$ и $Predator\_Vector_2$ написаны для выбора направления вращения участников задачи преследования $Wolf\ 1, Wolf\ 2, Wolf\ 3$ в зависимости от положения цели преследования в их локальной системе координат.

Функция расчета цели $Rabbit$ в локальной системе координат преследователя Fox. В локальной системе координат ось абсцисс сонаправлена вектором скорости VV преследователя Fox.

$$\text{Predator\_Vector}_0(VV, R_{fox}, R_{rabbit}) := \left| \begin{array}{l} E_{fox} \leftarrow \dfrac{1}{\sqrt{(VV_0)^2 + (VV_1)^2}} \cdot \begin{pmatrix} VV_0 & -VV_1 \\ VV_1 & VV_0 \end{pmatrix} \\ Rabbit_x \leftarrow (R_{rabbit} - R_{fox})^T \cdot E_{fox}^{\langle 0 \rangle} \\ Rabbit_y \leftarrow (R_{rabbit} - R_{fox})^T \cdot E_{fox}^{\langle 1 \rangle} \\ (Rabbit_x \;\; Rabbit_y) \end{array} \right.$$

Функция расчета точки пересечения прямой и окружности. Геометрический смысл показан на рисунке 3.19.

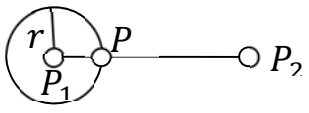


Рисунок 3.19. Пересечение прямой и окружности

Функция $InterSection_1$ рассчитывает точку пересечения отрезка $[P_1P_2]$ и окружности с центром в точке P_1 и радиуса r. На рисунке 3.19 это точка P.

$$InterSection_1(P_1,P_2,r) := \left[1 - \frac{r}{\sqrt{(P_{2_0} - P_{1_0})^2 + (P_{2_1} - P_{1_1})^2}}\right] \cdot P_1 + \frac{r}{\sqrt{(P_{2_0} - P_{1_0})^2 + (P_{2_1} - P_{1_1})^2}} \cdot P_2$$

Далее, следует функция расчета точки в локальной системе координат, связанной с преследователем, к которой будет стремиться вектор скорости.

$$Direction\_Predator_1(VV, R_{wolf}, R_{rabbit}, r) := \left| \begin{array}{l} E_{wolf} \leftarrow \frac{1}{\sqrt{(VV_0)^2 + (VV_1)^2}} \cdot \begin{pmatrix} VV_0 & -VV_1 \\ VV_1 & VV_0 \end{pmatrix} \\ Rabbit_x \leftarrow (R_{rabbit} - R_{wolf})^T \cdot E_{wolf}^{\langle 0 \rangle} \\ Rabbit_y \leftarrow (R_{rabbit} - R_{wolf})^T \cdot E_{wolf}^{\langle 1 \rangle} \\ Point \leftarrow InterSection_1\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} Rabbit_x \\ Rabbit_y \end{pmatrix}, r\right] \\ Point \end{array} \right.$$

Функции $InterSection_2$ и $Direction\_Predator_2$ имеют точно такое же назначение, как функции $InterSection_1$ и $Direction\_Predator_1$.

$$InterSection_2(P_1,P_2,r) := \left[1 - \frac{r}{\sqrt{(P_{2_0} - P_{1_0})^2 + (P_{2_1} - P_{1_1})^2}}\right] \cdot P_1 + \frac{r}{\sqrt{(P_{2_0} - P_{1_0})^2 + (P_{2_1} - P_{1_1})^2}} \cdot P_2$$

$$Direction\_Predator_2(VV, R_{wolf}, R_{rabbit}, r) := \left| \begin{array}{l} E_{wolf} \leftarrow \frac{1}{\sqrt{(VV_0)^2 + (VV_1)^2}} \cdot \begin{pmatrix} VV_0 & -VV_1 \\ VV_1 & VV_0 \end{pmatrix} \\ Rabbit_x \leftarrow (R_{rabbit} - R_{wolf})^T \cdot E_{wolf}^{\langle 0 \rangle} \\ Rabbit_y \leftarrow (R_{rabbit} - R_{wolf})^T \cdot E_{wolf}^{\langle 1 \rangle} \\ Point \leftarrow InterSection_2\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} Rabbit_x \\ Rabbit_y \end{pmatrix}, r\right] \\ Point \end{array} \right.$$

В нашей тестовой программе нами написана функция, рассчитывающей точку касания прямой и окружности.

В локальной системе координат (Рисунок 3.20) (e_1, O, e_2) касательная прямая проходит через начало координат точку O. Она должна коснуться

окружности с центром в точке C. Для расчета точек касания (точки B) служит функция $Tangent\_Line$.

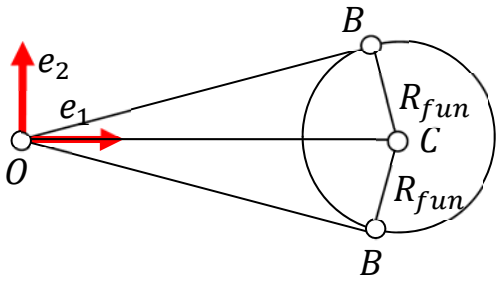


Рисунок 3.20. Касательная к окружности

Функция $Tangent\_Line$ рассчитывает две точки касания в локальной системе координат (e_1, O, e_2).

Given

$$(B_x - C_x)\cdot(0 - C_x) + (B_y - 0)\cdot(0 - 0) = R_{fun}^2$$

$$(B_x - C_x)\cdot(B_x - 0) + (B_y - 0)\cdot(B_y - 0) = 0$$

$$\text{Tangent\_Line}(C_x, R_{fun}) := \text{Find}(B_x, B_y) \rightarrow \begin{bmatrix} \frac{C_x^2 - R_{fun}^2}{C_x} & \frac{C_x^2 - R_{fun}^2}{C_x} \\ \frac{R_{fun}\cdot\sqrt{(C_x + R_{fun})\cdot(C_x - R_{fun})}}{C_x} & -\frac{R_{fun}\cdot\sqrt{(C_x + R_{fun})\cdot(C_x - R_{fun})}}{C_x} \end{bmatrix}$$

$$\text{Tangent\_Point}_{top}(C_x, R) := \text{Tangent\_Line}(C_x, R)^{\langle 0 \rangle}$$

$$\text{Tangent\_Point}_{bottom}(C_x, R) := \text{Tangent\_Line}(C_x, R)^{\langle 1 \rangle}$$

Given $B_x^2 + B_y^2 = r^2$ $(B_x - C_x)^2 + B_y^2 = R^2$

$$\text{Intersection\_Point}(C_x, R, r) := \text{Find}(B_x, B_y) \rightarrow \begin{bmatrix} \frac{C_x^2 - R^2 + r^2}{2\cdot C_x} \\ \frac{\sqrt{(C_x + R - r)\cdot(C_x - R + r)\cdot(R - C_x + r)\cdot(C_x + R + r)}}{2\cdot C_x} \end{bmatrix}$$

$$\begin{bmatrix} \frac{C_x^2 - R^2 + r^2}{2\cdot C_x} \\ -\frac{\sqrt{(C_x + R - r)\cdot(C_x - R + r)\cdot(R - C_x + r)\cdot(C_x + R + r)}}{2\cdot C_x} \end{bmatrix}$$

$$\text{Intersection\_Point}_{top}(C_x, R, r) := \text{Intersection\_Point}(C_x, R, r)^{\langle 0 \rangle}$$

$$\text{Intersection\_Point}_{bottom}(C_x, R, r) := \text{Intersection\_Point}(C_x, R, r)^{\langle 1 \rangle}$$

Функция *Inersection_Point* рассчитывает точки пересечения двух окружностей. Взаимное расположение этих окружностей изображено на рисунке 3.21.

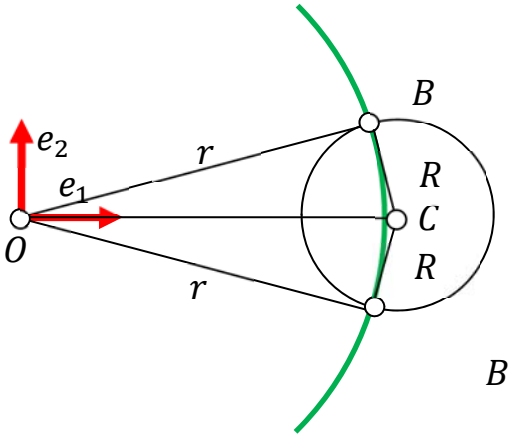


Рисунок 3.21. Точки пересечения окружностей

$$
\begin{aligned}
&\text{Point\_CS}(A,C,R,r) := \\
&\quad H_1 \leftarrow (1\ \ 0)^T \\
&\quad H_2 \leftarrow (0\ \ 1)^T \\
&\quad E_1 \leftarrow \frac{C - A}{|C - A|} \\
&\quad E_2 \leftarrow \left(-E1_1\ \ E1_0\right)^T \\
&\quad C_x \leftarrow (C - A)\cdot E_1 \\
&\quad B_1 \leftarrow \text{Tangent\_Point}_{top}(C_x,R) \ \text{ if } \ C_x > R \\
&\quad B_1 \leftarrow \text{Intersection\_Point}_{top}(C_x,R,r) \ \text{ otherwise} \\
&\quad h_1 \leftarrow \begin{pmatrix} H_1^T\cdot E_1 \\ H_1^T\cdot E_2 \end{pmatrix} \\
&\quad h_2 \leftarrow \begin{pmatrix} H_2^T\cdot E_1 \\ H_2^T\cdot E_2 \end{pmatrix} \\
&\quad B \leftarrow \begin{pmatrix} B_1^T\cdot h_1 \\ B_1^T\cdot h_2 \end{pmatrix} + A \\
&\quad B
\end{aligned}
$$

Функция $Point\_CS$ рассчитывает взаимное положение окружностей. Если расстояние между окружностями больше, чем R, то точка B – это точка касания, как на рисунке 3.20. Если расстояние между окружностями меньше, чем R, то точка B – это точка пересечения, как на рисунке 3.21. Входными параметрами функции $Point\_CS$ являются точки A, C и радиусы R и r.

Следует пояснить работу функции $Point\_CS$. Обратите внимание на рисунок 3.22.

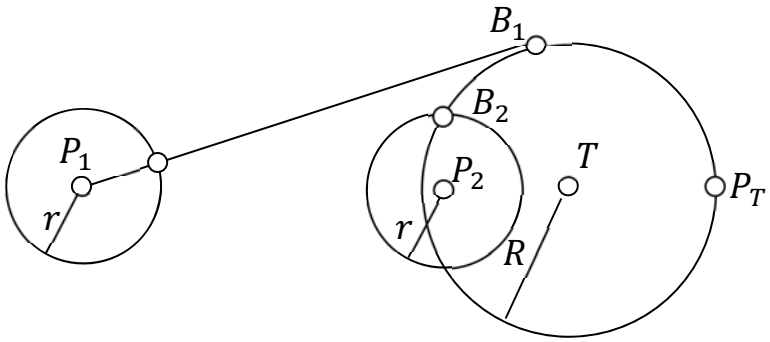


Рисунок 3.22. Выбор промежуточной точки, к которой надо стремиться для выполнения задачи

Допустим, преследователь может находиться в одном из двух положениях, P_1 или P_2. Задачей преследователя является достижение точки P_T.

Если расстояние между преследователем и целью ($[P_1T]$) больше чем радиус R окружности входа в точку P_T, то промежуточной целью будет являться достижение точки B_1. Точка B_1 - точка касания окружности с центром в точке T и радиуса R.

Если расстояние между преследователем и целью ($[P_2T]$) меньше или равно радиуса R окружности входа в точку P_T, то промежуточной целью будет являться достижение точки B_2. Точка B_2 - одна из точек пересечения окружностей (T, R) и (P_2, r). Под величиной радиуса r понимается шаг преследователя, движущегося со скоростью V_P, за дискретный промежуток времени ΔT.

Функция $Direction\_Predator_3$ рассчитывает в локальной системе координат преследователя (а именно, преследователя $Wolf\ 3$) точку промежуточной цели.

Входными параметрами функции являются: вектор скорости преследователя VV, угловая частота вращения преследователя ω, координаты преследователя R_{wolf} и шаг за дискретный промежуток времени r.

Выходным параметром является точка $Point$ в локальной системе координат преследователя, образованной вектором скорости преследователя.

$$\begin{aligned}
&\text{Direction_Predator}_3(VV, \omega, R_{wolf}, R_{rabbit}, r) := \\
&\quad E_{wolf} \leftarrow \frac{1}{\sqrt{(VV_0)^2 + (VV_1)^2}} \cdot \begin{pmatrix} VV_0 & -VV_1 \\ VV_1 & VV_0 \end{pmatrix} \\
&\quad R_{1.rabbit} \leftarrow \text{Point_CS}\left(R_{wolf}, R_{rabbit}, \frac{|VV|}{\omega}, r\right) \\
&\quad Rabbit_x \leftarrow \left(R_{1.rabbit} - R_{wolf}\right)^T \cdot E_{wolf}^{\langle 0 \rangle} \\
&\quad Rabbit_y \leftarrow \left(R_{1.rabbit} - R_{wolf}\right)^T \cdot E_{wolf}^{\langle 1 \rangle} \\
&\quad Point \leftarrow \text{InterSection}_2\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} Rabbit_x \\ Rabbit_y \end{pmatrix}, r\right] \\
&\quad Point
\end{aligned}$$

Функции $RabbitVector_1$ и $RabbitVector_2$ описывают поведение преследуемого объекта. Эти функции реализуют вращение по часовой стрелке и против часовой с шагом на расстояние за определенный момент времени.

$$\begin{aligned}
&\text{RabbitVector}_1(VV, RR, \omega) := \\
&\quad E \leftarrow (\cos(\omega \cdot \Delta T) \quad -\sin(\omega \cdot \Delta T)) \\
&\quad H \leftarrow \frac{1}{\sqrt{(VV_0)^2 + (VV_1)^2}} \cdot \begin{pmatrix} VV_0 & VV_1 \\ -VV_1 & VV_0 \end{pmatrix} \\
&\quad R \leftarrow \begin{bmatrix} \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \Delta T \cdot E \cdot H^{\langle 0 \rangle} + RR_0 \\ \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \Delta T \cdot E \cdot H^{\langle 1 \rangle} + RR_1 \end{bmatrix} \\
&\quad V \leftarrow \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \begin{pmatrix} E \cdot H^{\langle 0 \rangle} \\ E \cdot H^{\langle 1 \rangle} \end{pmatrix} \\
&\quad (E \quad H \quad R \quad V)
\end{aligned}$$

Входными параметрами функций $RabbitVector_1$ и $RabbitVector_2$ служат: вектор скорости преследуемой цели VV, координаты точки

преследуемой цели в мировой системе координат RR, угловая частота вращения преследуемой цели ω.

$$\text{RabbitVector}_2(VV, RR, \omega) := \left| \begin{array}{l} E \leftarrow (\cos(\omega \cdot \Delta T) \quad \sin(\omega \cdot \Delta T)) \\ H \leftarrow \dfrac{1}{\sqrt{(VV_0)^2 + (VV_1)^2}} \cdot \begin{pmatrix} VV_0 & VV_1 \\ -VV_1 & VV_0 \end{pmatrix} \\ R \leftarrow \begin{bmatrix} \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \Delta T \cdot E \cdot H^{\langle 0 \rangle} + RR_0 \\ \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \Delta T \cdot E \cdot H^{\langle 1 \rangle} + RR_1 \end{bmatrix} \\ V \leftarrow \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \begin{pmatrix} E \cdot H^{\langle 0 \rangle} \\ E \cdot H^{\langle 1 \rangle} \end{pmatrix} \\ (E \quad H \quad R \quad V) \end{array} \right.$$

Выходными параметрами служат единичный вектор поворота в локальной системе координат, образованной вектором скорости VV, векторы локального базиса H, координаты нового положения преследуемой цели R в мировой системе координат после поворота на угол $\omega \cdot \Delta T$ и после шага на расстояние $|VV| \cdot \Delta T$, новый вектор скорости преследуемого объекта V.

$$\text{RabbitVector}_0(VV, R_{fox}, R_{rabbit}) := \left| \begin{array}{l} E_{rabbit} \leftarrow \dfrac{1}{\sqrt{(VV_0)^2 + (VV_1)^2}} \cdot \begin{pmatrix} VV_0 & -VV_1 \\ VV_1 & VV_0 \end{pmatrix} \\ Fox_x \leftarrow (R_{fox} - R_{rabbit})^T \cdot E_{rabbit}{}^{\langle 0 \rangle} \\ Fox_y \leftarrow (R_{fox} - R_{rabbit})^T \cdot E_{rabbit}{}^{\langle 1 \rangle} \\ (Fox_x \quad Fox_y) \end{array} \right.$$

Функция $RabbitVector_0$ рассчитывает координаты преследователя R_{fox} в локальной системе координат, образованной вектором скорости VV преследуемого объекта. Входными параметрами являются: вектор скорости VV преследуемого объекта, координаты преследователя R_{fox}, координаты преследуемого объекта R_{rabbit}.

Функция $Situation$ является основной функцией нашей тестовой программы, где в итерационном цикле рассчитываются траектории всех участников задачи преследования.

Situation :=

$\left(v_{rabbit_0}\ v_{fox_0}\ r_{rabbit_0}\ r_{fox_0}\right) \leftarrow \left(V_{Rabbit0}\ V_{Fox0}\ R_{Rabbit0}\ R_{Fox0}\right)$

1

$\left(v_{wolf.1_0}\ r_{wolf.1_0}\right) \leftarrow \left(V_{wolf.1}\ R_{wolf.1}\right)$

$\left(v_{wolf.2_0}\ r_{wolf.2_0}\right) \leftarrow \left(V_{wolf.2}\ R_{wolf.2}\right)$

$\left(v_{wolf.3_0}\ r_{wolf.3_0}\right) \leftarrow \left(V_{wolf.2}\ R_{wolf.3}\right)$

2

$\left(target_{wolf.3_0}\ v_{target.wolf.3_0}\right) \leftarrow \left(R_{Rabbit0} - \frac{V_{wolf.3.0}}{\omega_{wolf.3}} \cdot \frac{V_{Rabbit0}}{\left|V_{Rabbit0}\right|}\ V_{Rabbit0}\right)$

3

$i \leftarrow 1$

$\Delta R_{animal} \leftarrow \left|r_{rabbit_0} - r_{fox_0}\right|$

4

while $(i \geq 1) \wedge (i \leq 500) \wedge \left(\Delta R_{animal} \geq \Delta R_0\right)$

5

$\left(fox_x\ fox_y\right) \leftarrow RabbitVector_0\left(v_{rabbit_{i-1}}, r_{fox_{i-1}}, r_{rabbit_{i-1}}\right)$

6

$\left(e_{xRabbit}\ h_{rabbit}\ r_{rabbit_i}\ v_{rabbit_i}\right)$

$\leftarrow RabbitVector_1\left(v_{rabbit_{i-1}}, r_{rabbit_{i-1}}, \omega_{rabbit}\right)$ if $fox_y \geq 0$

7

$\left(e_{xRabbit}\ h_{rabbit}\ r_{rabbit_i}\ v_{rabbit_i}\right)$

$\leftarrow RabbitVector_2\left(v_{rabbit_{i-1}}, r_{rabbit_{i-1}}, \omega_{rabbit}\right)$ otherwise

8

$\left(rabbit_x\ rabbit_y\right) \leftarrow Predator\_Vector_0\left(v_{fox_{i-1}}, r_{fox_{i-1}}, r_{rabbit_{i-1}}\right)$

9

$\left(e_{xFox}\ h_{fox}\ r_{fox_i}\ v_{fox_i}\right) \leftarrow Predator\_Vector_1\left(v_{fox_{i-1}}, r_{fox_{i-1}}, \omega_{fox}\right)$

if $rabbit_y \geq 0$

10

$\left(e_{xFox}\ h_{fox}\ r_{fox_i}\ v_{fox_i}\right) \leftarrow Predator\_Vector_2\left(v_{fox_{i-1}}, r_{fox_{i-1}}, \omega_{fox}\right)$

otherwise

11

if $\left|r_{wolf.1_{i-1}} - \left[r_{rabbit_{i-1}} + \frac{10}{\left|v_{rabbit_{i-1}}\right|} \cdot \begin{bmatrix} -\left(v_{rabbit_{i-1}}\right)_1 \\ \left(v_{rabbit_{i-1}}\right)_0 \end{bmatrix}\right]\right| > \Delta R_0$

12

$Victim_1 \leftarrow Direction\_Predator_1$

$$\left[v_{wolf.1_{i-1}}, r_{wolf.1_{i-1}}, r_{rabbit_{i-1}} + \frac{10}{\left| v_{rabbit_{i-1}} \right|} \cdot \begin{bmatrix} -\left(v_{rabbit_{i-1}} \right)_1 \\ \left(v_{rabbit_{i-1}} \right)_0 \end{bmatrix}, \omega_{wolf.1} \cdot \Delta T \right]$$

$\left(e_{x.wolf.1} \;\; h_{wolf.1} \;\; r_{wolf.1_i} \;\; v_{wolf.1_i} \right) \leftarrow Predator\_Vector_1$

13

$\left(v_{wolf.1_{i-1}}, r_{wolf.1_{i-1}}, \omega_{wolf.1} \right)$ if $Victim_{1_1} \geq 0$

$\left(e_{x.wolf.1} \;\; h_{wolf.1} \;\; r_{wolf.1_i} \;\; v_{wolf.1_i} \right) \leftarrow Predator\_Vector_2$

14

$\left(v_{wolf.1_{i-1}}, r_{wolf.1_{i-1}}, \omega_{wolf.1} \right)$ otherwise

otherwise

15

$$r_{wolf.1_i} \leftarrow r_{rabbit_{i-1}} + \frac{10}{\left| v_{rabbit_{i-1}} \right|} \cdot \begin{bmatrix} -\left(v_{rabbit_{i-1}} \right)_1 \\ \left(v_{rabbit_{i-1}} \right)_0 \end{bmatrix}$$

16

$v_{wolf.1_i} \leftarrow v_{rabbit_i}$

17

$$\text{if } \left| r_{wolf.2_{i-1}} - \left[r_{rabbit_{i-1}} + \frac{-10}{\left| v_{rabbit_{i-1}} \right|} \cdot \begin{bmatrix} -\left(v_{rabbit_{i-1}} \right)_1 \\ \left(v_{rabbit_{i-1}} \right)_0 \end{bmatrix} \right] \right| > \Delta R_0$$

18

$Victim_2 \leftarrow Direction\_Predator_1$

19

$$\left[v_{wolf.2_{i-1}}, r_{wolf.2_{i-1}}, r_{rabbit_{i-1}} + \frac{-10}{\left| v_{rabbit_{i-1}} \right|} \cdot \begin{bmatrix} -\left(v_{rabbit_{i-1}} \right)_1 \\ \left(v_{rabbit_{i-1}} \right)_0 \end{bmatrix}, \omega_{wolf.2} \cdot \Delta T \right]$$

$\left(e_{x.wolf.2} \;\; h_{wolf.2} \;\; r_{wolf.2_i} \;\; v_{wolf.2_i} \right) \leftarrow Predator\_Vector_1$

20

$\left(v_{wolf.2_{i-1}}, r_{wolf.2_{i-1}}, \omega_{wolf.2} \right)$ if $Victim_{2_1} \geq 0$

$\left(e_{x.wolf.2} \;\; h_{wolf.2} \;\; r_{wolf.2_i} \;\; v_{wolf.2_i} \right) \leftarrow Predator\_Vector_2$

21

$\left(v_{wolf.2_{i-1}}, r_{wolf.2_{i-1}}, \omega_{wolf.2} \right)$ otherwise

otherwise

22

$$r_{wolf.2_i} \leftarrow r_{rabbit_{i-1}} + \frac{-10}{\left| v_{rabbit_{i-1}} \right|} \cdot \begin{bmatrix} -\left(v_{rabbit_{i-1}} \right)_1 \\ \left(v_{rabbit_{i-1}} \right)_0 \end{bmatrix}$$

23

Для того, чтобы пояснить работу функции *Situation*, мы составили ниже таблицу, в которой представили работу каждого оператора из листинга функции *Situation*.

1.	Блок операторов, которые присваивают начальным значениям массивов локальных переменных стартовые соответствующие значения задачи преследования. Переменные стартовых значений задачи преследования для функции $Situation$ являются глобальными.
2.	Отдельного определения заслуживает цель для преследователя $Wolf\ 3$. Скорость этой точки равна скорости движения преследуемого объекта.
3.	Блок операторов подготовки к выполнению цикла по условию. Присвоение счетчику циклов i единичного значения. Задание стартового расстояния между объектами $Rabbit$ и Fox. Поскольку для преследуемого объекта $Rabbit$ выбрана стратегия убегания от объекта Fox.
4.	Цикл выполнения блока операторов по условию. Пока истинно условие, что $i \geq 1$ и $i \leq 500$ и расстояние между объектами-преследователями и преследуемым объектом больше порогового значения (глобальная переменная) выполнять блок операторов.
5.	Перевод текущих координат объекта Fox в систему координат, связанной с текущей скоростью объекта $Rabbit$, с центром в точке нахождения объекта $Rabbit$.
6.	Производится анализ положения объекта Fox в локальной системе координат. Если объект Fox находится в верхней полуплоскости, то объект $Rabbit$ производит вращение по часовой стрелке. Если объект Fox находится в нижней полуплоскости, то объект $Rabbit$ производит вращение против часовой стрелки.
7.	
8.	Перевод текущих координат объекта $Rabbit$ в систему координат, связанной с текущей скоростью объекта Fox, с центром в точке нахождения объекта Fox.

9.	Производится анализ положения объекта $Rabbit$ в локальной системе координат. Если объект $Rabbit$ находится в верхней полуплоскости, то объект Fox производит вращение против часовой стрелки. Если объект $Rabbit$ находится в нижней полуплоскости, то объект Fox производит вращение по часовой стрелке.
10.	
11.	Оператор условия для поведения преследователя $Wolf\ 1$. На рисунке 3.17 показана точка, достижение которой является задачей преследователя $Wolf\ 1$. Это точка T_1. Если текущее расстояние между объектом $Wolf\ 1$ и точкой T_1 больше порогового значения ΔR_0, то выполнять следующие операторы.
12.	Перевод координат точки T_1 в локальную систему координат, образованную текущим значением скорости объекта $Wolf\ 1$ с центром в месте положения $Wolf\ 1$.
13.	Выполняется анализ положения точки T_1 в локальной системе координат. Если точка T_1 в локальной системе координат находится в верхней полуплоскости, то объект $Wolf\ 1$ совершает вращение и шаг против часовой стрелки.
14.	Если точка T_1 в локальной системе координат находится в нижней полуплоскости, то объект $Wolf\ 1$ совершает вращение и шаг по часовой стрелке.
15.	Если текущее расстояние между объектом $Wolf\ 1$ и точкой T_1 меньше порогового значения ΔR_0, то выполнять следующие операторы.
16.	Присвоить объекту $Wolf\ 1$ координаты точки T_1.
17.	Скорость объекта $Wolf\ 1$ равна скорости T_1.
18.	Оператор условия для поведения преследователя $Wolf\ 2$. На рисунке 3.17 показана точка, достижение которой является задачей преследователя $Wolf\ 2$. Это точка T_2. Если текущее расстояние

	между объектом $Wolf\ 2$ и точкой T_2 больше порогового значения ΔR_0, то выполнять следующие операторы.
19.	Перевод координат точки T_2 в локальную систему координат, образованную текущим значением скорости объекта $Wolf\ 2$ с центром в месте положения $Wolf\ 2$.
20.	Выполняется анализ положения точки T_2 в локальной системе координат. Если точка T_2 в локальной системе координат находится в верхней полуплоскости, то объект $Wolf\ 2$ совершает вращение и шаг против часовой стрелки.
21.	Если точка T_2 в локальной системе координат находится в нижней полуплоскости, то объект $Wolf\ 2$ совершает вращение и шаг по часовой стрелке.
22.	Если текущее расстояние между объектом $Wolf\ 1$ и точкой T_1 меньше порогового значения ΔR_0, то выполнять следующие операторы.
23.	Присвоить объекту $Wolf\ 2$ координаты точки T_2.
24.	Скорость объекта $Wolf\ 2$ равна скорости T_2
25.	В этом операторе назначается точка, достижение которой является целью для объекта $Wolf\ 3$. Это точка T на рисунке 3.3. Текущая скорость точки T равна скорости объекта $Rabbit$.
26.	Условный оператор проверки текущего расстояния между преследующим объектом $Wolf\ 3$ и целью для этого объекта. Это будет текущее положение точки T, как на рисунке 3.3. Если расстояние меньше, чем радиус допустимой кривизны траектории объекта $Wolf\ 3$, то следует выполнять такой оператор.
27.	Оператор вращения вектора скорости объекта $Wolf\ 3$ против часовой стрелки. Фактически, объект $Wolf\ 3$ стремится вернуться на окружность радиуса $\lvert V_{wolf.3}\rvert/\omega_{wolf.3}$ с центром в точке T (Рисунок 3.3).

28.	Если текущее расстояние между объектом $Wolf\ 3$ и его целью T больше порогового значения ΔR_0 выполнять следующие операторы.
29.	Назначить целью для $Wolf\ 3$ точку B_1 (Рисунок 3.22), перевести ее координату в систему координат, связанную с текущей скоростью объекта $Wolf\ 3$.
30.	Если точка B_1 находится в локальной системе координат в верхней полуплоскости, то объект $Wolf\ 3$ совершает вращение против часовой стрелки.
31.	Если точка B_1 находится в локальной системе координат в нижней полуплоскости, то объект $Wolf\ 3$ совершает вращение по часовой стрелке.
32.	Если текущее расстояние между объектом $Wolf\ 3$ и текущей точкой преследуемого объекта меньше порогового значения ΔR_0 выполнять следующие операторы.
33.	Присвоить текущему положению объекта $Wolf\ 3$ текущее положение преследуемого объекта. Подчеркнем, что этот оператор во время тестовых прогонов программы не выполнялся. Необходим для создания алгоритма без незнакомых ситуаций. Так, на всякий случай. Присвоить скорости объекта $Wolf\ 3$ текущую скорость преследуемого объекта.
34.	Увеличения счетчика циклов на единицу.
35.	Вывод полученного значения количества совершенных циклов
36.	Формирований вектора выходных значений функции $Situation$.

Выходными параметрами функции $Situation$ служат: массив точек траектории объекта $Rabbit$, массив точек траектории объекта Fox, общее число рассчитанных циклов, массив точек траектории объекта $Wolf\ 1$, массив точек траектории объекта $Wolf\ 2$, массив точек траектории объекта $Wolf\ 3$.

$FrameNumber := (Situation^T)_2$	$FrameNumber = 244$

Общее число рассчитанных циклов.

$i := 0..FrameNumber$
Создание ранжированной переменной по общему количеству циклов.
$Rabbit_{x_i} := [[(Situation^T)_0]_i]_0$ $Rabbit_{y_i} := [[(Situation^T)_0]_i]_1$
Массив точек объекта Fox, сформированный покоординатно.
$X_{rabbit} := Rabbit_{x_{FRAME}}$ $Y_{rabbit} := Rabbit_{y_{FRAME}}$
Точки объекта объекта Fox, соответствующие кадру $FRAME$.
$Fox_{x_i} := [[(Situation^T)_1]_i]_0$ $Fox_{y_i} := [[(Situation^T)_1]_i]_1$
Массив точек объекта $Rabbit$, сформированный покоординатно.
$X_{fox} := Fox_{x_{FRAME}}$ $Y_{fox} := Fox_{y_{FRAME}}$
Точки объекта объекта $Rabbit$, соответствующие кадру $FRAME$.
$Wolf_{1.x_i} := [[(Situation^T)_3]_i]_0$ $Wolf_{1.y_i} := [[(Situation^T)_3]_i]_1$
Массив точек объекта $Wolf\ 1$, сформированный покоординатно.
$X_{wolf.1} := Wolf_{1.x_{FRAME}}$ $Y_{wolf.1} := Wolf_{1.y_{FRAME}}$
Точки объекта объекта $Wolf\ 1$ соответствующие кадру $FRAME$.
$Wolf_{2.x_i} := [[(Situation^T)_4]_i]_0$ $Wolf_{2.y_i} := [[(Situation^T)_4]_i]_1$
Массив точек объекта $Wolf\ 2$, сформированный покоординатно.
$X_{wolf.2} := Wolf_{2.x_{FRAME}}$ $Y_{wolf.2} := Wolf_{2.y_{FRAME}}$
Точки объекта объекта $Wolf\ 2$, соответствующие кадру $FRAME$.
$Wolf_{3.x_i} := [[(Situation^T)_5]_i]_0$ $Wolf_{3.y_i} := [[(Situation^T)_5]_i]_1$
Массив точек объекта $Wolf\ 3$, сформированный покоординатно.
$X_{wolf.3} := Wolf_{3.x_{FRAME}}$ $Y_{wolf.3} := Wolf_{3.y_{FRAME}}$
Точки объекта объекта $Wolf\ 3$, соответствующие кадру $FRAME$.
$Predators_x := stack(X_{wolf.1}, X_{wolf.2}, X_{wolf.3})$ $Predators_y := stack(Y_{wolf.1}, Y_{wolf.2}, Y_{wolf.3})$

Объединение в координатные массивы точек объектов $Wolf\ 1, Wolf\ 2,$ $Wolf\ 3$, соответствующие кадру $FRAME$.

Далее, для наглядной визуализации процесса группового преследования одиночной цели, мы хотим прорисовать траекторию каждого участника от начала процесса преследования до определенного момента времени.

```
Traectory(animal) := | Q ← diag(animal)
                     | for j ∈ 1..FrameNumber
                     |   for i ∈ 0..j
                     |     Q_{i,j} ← animal_i
                     | Q
```

Для этого функцией $Trajectory$ создается на основе вектор - столбца создается диагональная матрица. Работа функции $Trajectory$ продемонстрирована на рисунке 3.23.

Traectory(Rabbit_x) =

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0.387	0.387	0.387	0.387	0.387	0.387	0.387
2	0	0	0.768	0.768	0.768	0.768	0.768	0.768
3	0	0	0	1.144	1.144	1.144	1.144	1.144
4	0	0	0	0	1.513	1.513	1.513	1.513
5	0	0	0	0	0	1.876	1.876	1.876
6	0	0	0	0	0	0	2.231	2.231
7	0	0	0	0	0	0	0	2.578
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0

Рисунок 3.23. Работа функции $Trajectory$

Входным параметром служит вектор – столбец, а выходным служит параметром треугольная матрица, на главной диагонали которой стоят соответствующие значения вектора – столбца.

При помощи функции $Trajectory$ будет прорисовываться пройденная траектория участника задачи преследования от момента начала преследования до момента, соответствующего отдельному кадру.

$RabbitTraectory_x(Kadr) := submatrix(Traectory(Rabbit_x), 0, Kadr, Kadr, Kadr)$	Траектория объекта $Rabbit$
$RabbitTraectory_y(Kadr) := submatrix(Traectory(Rabbit_y), 0, Kadr, Kadr, Kadr)$	
$FoxTraectory_x(Kadr) := submatrix(Traectory(Fox_x), 0, Kadr, Kadr, Kadr)$	Траектория объекта Fox
$FoxTraectory_y(Kadr) := submatrix(Traectory(Fox_y), 0, Kadr, Kadr, Kadr)$	
$W_{1.y}(Kadr) := submatrix(Traectory(Wolf_{1.y}), 0, Kadr, Kadr, Kadr)$	Траектория объекта $Wolf\ 1$
$W_{1.x}(Kadr) := submatrix(Traectory(Wolf_{1.x}), 0, Kadr, Kadr, Kadr)$	
$W_{2.x}(Kadr) := submatrix(Traectory(Wolf_{2.x}), 0, Kadr, Kadr, Kadr)$	Траектория объекта $Wolf\ 2$
$W_{2.y}(Kadr) := submatrix(Traectory(Wolf_{2.y}), 0, Kadr, Kadr, Kadr)$	
$WolfTraectory_{3.x}(Kadr) := submatrix(Traectory(Wolf_{3.x}), 0, Kadr, Kadr, Kadr)$	Траектория объекта $Wolf\ 3$
$WolfTraectory_{3.y}(Kadr) := submatrix(Traectory(Wolf_{3.y}), 0, Kadr, Kadr, Kadr)$	

На рисунке 3.24 показан результат работы тестовой программы группового преследования с различными стратегиями.

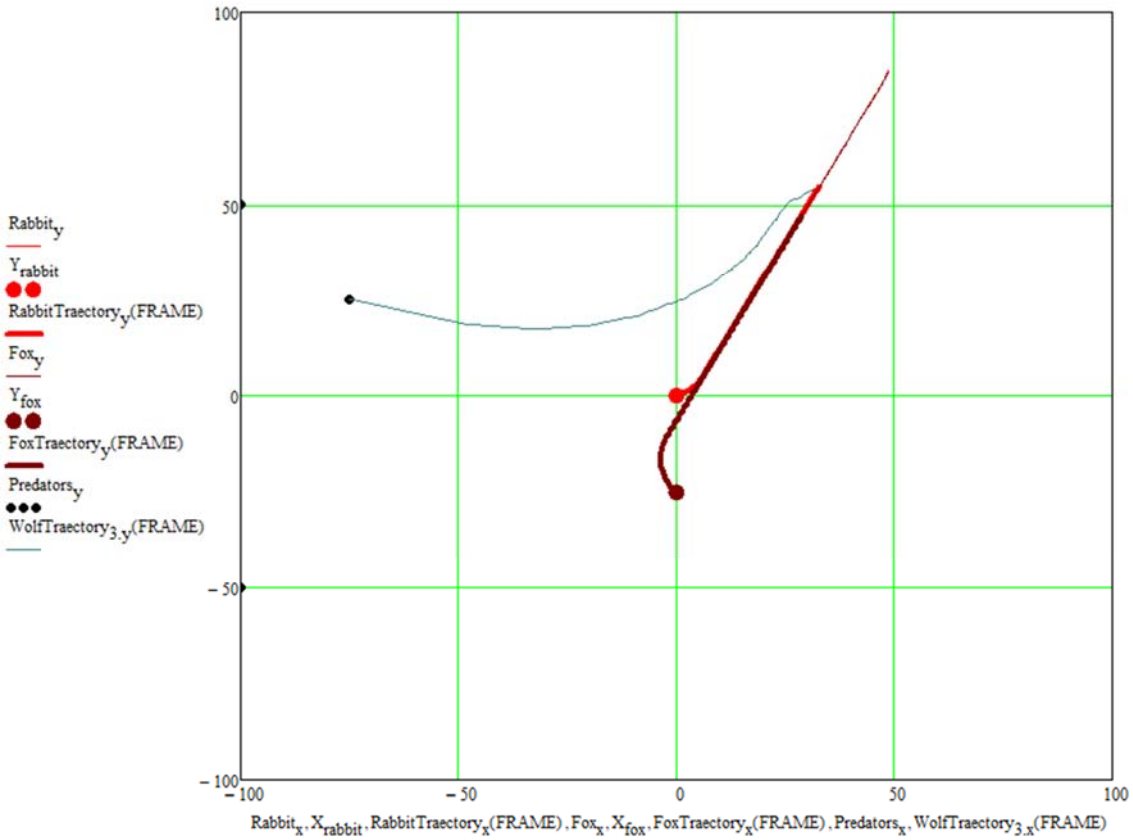


Рисунок 3.24. Групповое преследование с различными стратегиями

Рисунок 3.24 дополнен ссылкой на анимированное изображение.

3.5 КОММЕНТАРИИ К ПРОГРАММЕ ПРЕСЛЕДОВАНИЯ ГРУППОЙ ОДНОЙ ЦЕЛИ ЖЕСТКИМИ СВЯЗЯМИ

В программе преследования группой одной цели с жесткими связями (Рисунок 3.25) предполагается, что преследуемый объект ($Rabbit$) в поведении ориентируется на убегание только от объекта Fox, используя стратегию корректировки направления движения.

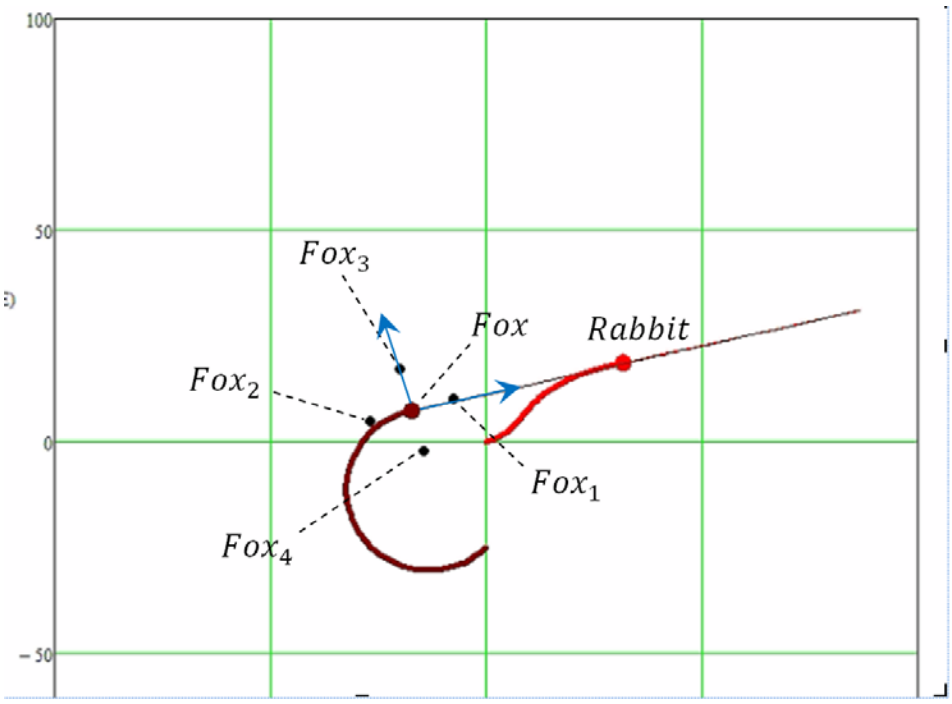


Рисунок 3.25. Групповое преследование с жесткими связями

Преследующий объект Fox в своем преследовании объекта $Rabbit$ использует стратегию корректировки направления движения, которая заключается в стремлении совмещения вектора скорости V_{fox} с линией визирования $(Fox, Rabbit)$.

Преследующие объекты Fox_1, Fox_2, Fox_3 и Fox_4 имеют постоянные фиксированные координаты в системе координат, образованной вектором скорости V_{fox}, и центром в точке положения Fox.

$V_{rabbit} := 8$ — Модуль скорости объекта $Rabbit$

$\lambda_{rabbit} := \frac{\pi}{12}$ — Угол наклона скорости объекта $Rabbit$ к оси абсцисс в мировой системе координат

$V_{fox} := 15$ — Модуль скорости объекта Fox

$\lambda_{fox} := \frac{-3\pi}{4}$ — Угол наклона скорости объекта Fox к оси абсцисс в мировой системе координат

$\Delta T := \frac{25.3}{500}$ $\Delta T = 0.051$ — Назначение дискретного промежутка времени. Назначение произведено после многочисленных прогонов тестовой программы

$T_{rabbit} := 12$ Период вращения объекта $Rabbit$

$\omega_{rabbit} := \frac{2\pi}{T_{rabbit}}$ Частота вращения объекта $Rabbit$. Параметр необходим для определения минимального радиуса кривизны траектории

$T_{fox} := 8$ Период вращения объекта Fox.

$\omega_{fox} := \frac{2\pi}{T_{fox}}$ Частота вращения объекта Fox. Параметр необходим для определения минимального радиуса кривизны траектории

$\beta_0 := \omega_{fox} \cdot \Delta T$ Пороговое значение вращения вектора скорости объекта Fox, при достижении которого считается, что скорость совпадает с линией визирования

$\Delta R_0 := V_{fox} \cdot \Delta T$ Пороговое расстояния между объектами $Rabbit$ и Fox, при достижении которого считается, что объект $Rabbit$ настиг объект Fox

$\Delta R_0 = 0.759$

$R_{Rabbit0} := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ Стартовая позиция объекта $Rabbit$

$V_{Rabbit0} := V_{rabbit} \cdot \begin{pmatrix} \cos(\lambda_{rabbit}) \\ \sin(\lambda_{rabbit}) \end{pmatrix}$ Стартовый вектор скорости объекта $Rabbit$

$R_{Fox0} := \begin{pmatrix} 0 \\ -25 \end{pmatrix}$ Стартовая позиция объекта Fox

$V_{Fox0} := V_{fox} \cdot \begin{pmatrix} \cos(\lambda_{fox}) \\ \sin(\lambda_{fox}) \end{pmatrix}$ Стартовый вектор скорости объекта Fox

Функция $FoxVector_0$ переводит точку R_{rabbit} в локальную систему координат.

$$\text{FoxVector}_0(VV, R_{fox}, R_{rabbit}) := \left| \begin{array}{l} E_{fox} \leftarrow \dfrac{1}{\sqrt{(VV_0)^2 + (VV_1)^2}} \cdot \begin{pmatrix} VV_0 & -VV_1 \\ VV_1 & VV_0 \end{pmatrix} \\ Rabbit_x \leftarrow (R_{rabbit} - R_{fox})^T \cdot E_{fox}^{\langle 0 \rangle} \\ Rabbit_y \leftarrow (R_{rabbit} - R_{fox})^T \cdot E_{fox}^{\langle 1 \rangle} \\ (Rabbit_x \;\; Rabbit_y) \end{array} \right.$$

Локальная система координат образована вектором VV и перпендикулярным ему вектором. Центр координат находится в точке R_{fox}.

Входными параметрами являются вектор VV, точки R_{fox} и R_{rabbit}. Выходным параметром является преобразованный в локальную систему координат вектор $Rabbit$.

$$\text{FoxVector}_1(VV, RR) := \left| \begin{array}{l} E \leftarrow (\cos(\omega_{fox} \cdot \Delta T) \;\; \sin(\omega_{fox} \cdot \Delta T)) \\ H \leftarrow \dfrac{1}{\sqrt{(VV_0)^2 + (VV_1)^2}} \cdot \begin{pmatrix} VV_0 & VV_1 \\ -VV_1 & VV_0 \end{pmatrix} \\ R \leftarrow \begin{bmatrix} \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \Delta T \cdot E \cdot H^{\langle 0 \rangle} + RR_0 \\ \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \Delta T \cdot E \cdot H^{\langle 1 \rangle} + RR_1 \end{bmatrix} \\ V \leftarrow \sqrt{(VV_0)^2 + (VV_1)^2} \cdot \begin{pmatrix} E \cdot H^{\langle 0 \rangle} \\ E \cdot H^{\langle 1 \rangle} \end{pmatrix} \\ (E \;\; H \;\; R \;\; V) \end{array} \right.$$

Функция $FoxVector_1$ реализует вращение в локальной системе координат, связанной с вектором VV, единичного вектора оси абсцисс на угол $\omega \cdot \Delta T$ против часовой стрелки. Совершает перемещение точки вдоль вектора вращения на расстояние $VV \cdot \Delta T$. Полученная точка переводится в мировую систему координат.

Входными параметрами являются вектор скорости VV и вектор точки RR, выходными параметрами являются: вращаемый вектор E, локальный базис H, сформированный на основе вектора VV, точка R после поворота и линейного шага, переведенная в мировую систему координат, скорость V после поворота в мировой системе координат.

У функции $FoxVector_2$ действие аналогичное как у функции $FoxVector_1$, только вращение происходит по часовой стрелке.

$$\text{FoxVector}_2(VV, RR) := \left| \begin{array}{l} E \leftarrow \left(\cos\left(\omega_{fox} \cdot \Delta T\right) \ -\sin\left(\omega_{fox} \cdot \Delta T\right)\right) \\ H \leftarrow \dfrac{1}{\sqrt{\left(VV_0\right)^2 + \left(VV_1\right)^2}} \cdot \begin{pmatrix} VV_0 & VV_1 \\ -VV_1 & VV_0 \end{pmatrix} \\ R \leftarrow \begin{bmatrix} \sqrt{\left(VV_0\right)^2 + \left(VV_1\right)^2} \cdot \Delta T \cdot E \cdot H^{\langle 0 \rangle} + RR_0 \\ \sqrt{\left(VV_0\right)^2 + \left(VV_1\right)^2} \cdot \Delta T \cdot E \cdot H^{\langle 1 \rangle} + RR_1 \end{bmatrix} \\ V \leftarrow \sqrt{\left(VV_0\right)^2 + \left(VV_1\right)^2} \cdot \begin{pmatrix} E \cdot H^{\langle 0 \rangle} \\ E \cdot H^{\langle 1 \rangle} \end{pmatrix} \\ (E \ H \ R \ V) \end{array} \right.$$

Функция $RabbitVector_0$ переводит точку R_{fox} в локальную систему координат.

$$\text{RabbitVector}_0\left(VV, R_{fox}, R_{rabbit}\right) := \left| \begin{array}{l} E_{rabbit} \leftarrow \dfrac{1}{\sqrt{\left(VV_0\right)^2 + \left(VV_1\right)^2}} \cdot \begin{pmatrix} VV_0 & -VV_1 \\ VV_1 & VV_0 \end{pmatrix} \\ Fox_x \leftarrow \left(R_{fox} - R_{rabbit}\right)^T \cdot E_{rabbit}{}^{\langle 0 \rangle} \\ Fox_y \leftarrow \left(R_{fox} - R_{rabbit}\right)^T \cdot E_{rabbit}{}^{\langle 1 \rangle} \\ \left(Fox_x \ Fox_y\right) \end{array} \right.$$

Локальная система координат образована вектором VV и перпендикулярным ему вектором. Центр координат находится в точке R_{rabbit}.

Функция $RabbitVector_1$ реализует вращение в локальной системе координат, связанной с вектором VV, единичного вектора оси абсцисс на угол $\omega \cdot \Delta T$ по часовой стрелке. Совершает перемещение точки вдоль вектора вращения на расстояние $VV \cdot \Delta T$. Полученная точка переводится в мировую систему координат.

$$\text{RabbitVector}_1(VV, RR) := \left| \begin{array}{l} E \leftarrow \left(\cos\left(\omega_{rabbit}\cdot\Delta T\right) \ -\sin\left(\omega_{rabbit}\cdot\Delta T\right)\right) \\ H \leftarrow \dfrac{1}{\sqrt{\left(VV_0\right)^2+\left(VV_1\right)^2}}\cdot\begin{pmatrix} VV_0 & VV_1 \\ -VV_1 & VV_0 \end{pmatrix} \\ R \leftarrow \begin{bmatrix} \sqrt{\left(VV_0\right)^2+\left(VV_1\right)^2}\cdot\Delta T\cdot E\cdot H^{\langle 0\rangle}+RR_0 \\ \sqrt{\left(VV_0\right)^2+\left(VV_1\right)^2}\cdot\Delta T\cdot E\cdot H^{\langle 1\rangle}+RR_1 \end{bmatrix} \\ V \leftarrow \sqrt{\left(VV_0\right)^2+\left(VV_1\right)^2}\cdot\begin{pmatrix} E\cdot H^{\langle 0\rangle} \\ E\cdot H^{\langle 1\rangle} \end{pmatrix} \\ (E \ H \ R \ V) \end{array} \right.$$

Входными параметрами являются вектор скорости VV и вектор точки RR, выходными параметрами являются: вращаемый вектор E, локальный базис H, сформированный на основе вектора VV, точка R после поворота и линейного шага, переведенная в мировую систему координат, скорость V после поворота в мировой системе координат.

$$\text{RabbitVector}_2(VV, RR) := \left| \begin{array}{l} E \leftarrow \left(\cos\left(\omega_{rabbit}\cdot\Delta T\right) \ \sin\left(\omega_{rabbit}\cdot\Delta T\right)\right) \\ H \leftarrow \dfrac{1}{\sqrt{\left(VV_0\right)^2+\left(VV_1\right)^2}}\cdot\begin{pmatrix} VV_0 & VV_1 \\ -VV_1 & VV_0 \end{pmatrix} \\ R \leftarrow \begin{bmatrix} \sqrt{\left(VV_0\right)^2+\left(VV_1\right)^2}\cdot\Delta T\cdot E\cdot H^{\langle 0\rangle}+RR_0 \\ \sqrt{\left(VV_0\right)^2+\left(VV_1\right)^2}\cdot\Delta T\cdot E\cdot H^{\langle 1\rangle}+RR_1 \end{bmatrix} \\ V \leftarrow \sqrt{\left(VV_0\right)^2+\left(VV_1\right)^2}\cdot\begin{pmatrix} E\cdot H^{\langle 0\rangle} \\ E\cdot H^{\langle 1\rangle} \end{pmatrix} \\ (E \ H \ R \ V) \end{array} \right.$$

У функции $RabbitVector_2$ действие аналогичное как у функции $RabbitVector_1$, только вращение происходит против часовой стрелки.

Функция $Situation$ является основной функцией нашей тестовой программы, где в итерационном цикле рассчитываются траектории всех участников задачи преследования.

Situation :=	$\left(v_{rabbit_0} \ v_{fox_0} \ r_{rabbit_0} \ r_{fox_0}\right) \leftarrow \left(V_{Rabbit0} \ V_{Fox0} \ R_{Rabbit0} \ R_{Fox0}\right)$	1
	$Vector_1 \leftarrow 10\cdot\dfrac{V_{Fox0}}{\left\|V_{Fox0}\right\|}$	2

	$\text{Vector}_2 \leftarrow -\text{Vector}_1$	3
	$\text{Vector}_3 \leftarrow 10 \cdot \frac{\left(-V_{Fox0_1} \quad V_{Fox0_0}\right)^T}{\left\|V_{Fox0}\right\|}$	4
	$\text{Vector}_4 \leftarrow -\text{Vector}_3$	5
	$r_{fox.1_0} \leftarrow r_{fox_0} + \text{Vector}_1$	6
	$r_{fox.2_0} \leftarrow r_{fox_0} + \text{Vector}_2$	7
	$r_{fox.3_0} \leftarrow r_{fox_0} + \text{Vector}_3$	8
	$r_{fox.4_0} \leftarrow r_{fox_0} + \text{Vector}_4$	9
	$i \leftarrow 1$	10
	$\Delta R_{animal} \leftarrow \left\|r_{rabbit_0} - r_{fox_0}\right\|$	11
	while $(i \geq 1) \wedge (i \leq 500) \wedge \left(\Delta R_{animal} \geq \Delta R_0\right)$	12
	$\left(fox_x \quad fox_y\right) \leftarrow \text{RabbitVector}_0\left(v_{rabbit_{i-1}}, r_{fox_{i-1}}, r_{rabbit_{i-1}}\right)$	13
	$\left(e_{xRabbit} \quad h_{rabbit} \quad r_{rabbit_i} \quad v_{rabbit_i}\right) \leftarrow \text{RabbitVector}_1\left(v_{rabbit_{i-1}}, r_{rabbit_{i-1}}\right)$	14
	if $fox_y \geq 0$	
	$\left(e_{xRabbit} \quad h_{rabbit} \quad r_{rabbit_i} \quad v_{rabbit_i}\right) \leftarrow \text{RabbitVector}_2\left(v_{rabbit_{i-1}}, r_{rabbit_{i-1}}\right)$	15
	otherwise	
	$\left(rabbit_x \quad rabbit_y\right) \leftarrow \text{FoxVector}_0\left(v_{fox_{i-1}}, r_{fox_{i-1}}, r_{rabbit_{i-1}}\right)$	16
	$\left(e_{xFox} \quad h_{fox} \quad r_{fox_i} \quad v_{fox_i}\right) \leftarrow \text{FoxVector}_1\left(v_{fox_{i-1}}, r_{fox_{i-1}}\right)$	17
	if $rabbit_y \geq 0$	
	$\left(e_{xFox} \quad h_{fox} \quad r_{fox_i} \quad v_{fox_i}\right) \leftarrow \text{FoxVector}_2\left(v_{fox_{i-1}}, r_{fox_{i-1}}\right)$	18
	otherwise	
	$v_{buf} \leftarrow v_{fox_{i-1}}$	19

	$r_{fox.1_i} \leftarrow r_{fox_i} + 10 \cdot \frac{v_{buf}}{\lvert v_{buf} \rvert}$	20
	$r_{fox.2_i} \leftarrow r_{fox_i} + 10 \cdot \frac{-v_{buf}}{\lvert v_{fox_{i-1}} \rvert}$	21
	$r_{fox.3_i} \leftarrow r_{fox_i} + 10 \cdot \frac{-\left(-v_{buf_1} \quad v_{buf_0}\right)^T}{\lvert v_{buf} \rvert}$	22
	$r_{fox.4_i} \leftarrow r_{fox_i} + 10 \cdot \frac{\left(-v_{buf_1} \quad v_{buf_0}\right)^T}{\lvert v_{buf} \rvert}$	23
	$\Delta R_{animal} \leftarrow \lvert r_{rabbit_i} - r_{fox_i} \rvert$	24
	$i \leftarrow i + 1$	25
	$Total \leftarrow i - 1$	26
	$\left(r_{rabbit} \quad r_{fox} \quad Total \quad r_{fox.1} \quad r_{fox.2} \quad r_{fox.3} \quad r_{fox.4}\right)$	27

Для того, чтобы пояснить работу функции $Situation$, мы составили ниже таблицу, в которой представили работу каждого оператора из листинга функции $Situation$.

1	Инициализация массивов векторов скоростей и положений объектов $Rabbit$ и Fox
2	Установка вектора смещения для объекта Fox_1
3	Установка вектора смещения для объекта Fox_2
4	Установка вектора смещения для объекта Fox_3
5	Установка вектора смещения для объекта Fox_4
6	Установка стартового положения объекта Fox_1
7	Установка стартового положения объекта Fox_2
8	Установка стартового положения объекта Fox_3
9	Установка стартового положения объекта Fox_4
10	Инициализация счетчика циклов
11	Задание начального расстояния между объектами $Rabbit$ и Fox

12	Выполнение цикла по условию: выполнять, если счетчик циклов в диапазоне от 0 до 500 и расстояние между объектами $Rabbit$ и Fox больше порогового значения ΔR_0
13	Присвоить в локальном базисе, образованного текущей скоростью объекта $Rabbit$, и центром в точке положения объекта $Rabbit$, локальные координаты объекта Fox
14	Если объект Fox находится в верхней полуплоскости локального базиса, то вектор скорости объекта $Rabbit$ необходимо вращать по часовой стрелке с шагом на расстояние, соответствующее временному промежутку
15	Если объект Fox находится в нижней полуплоскости локального базиса, то вектор скорости объекта $Rabbit$ необходимо вращать против часовой стрелки с шагом на расстояние, соответствующее временному промежутку
16	Присвоить в локальном базисе, образованного текущей скоростью объекта Fox, и центром в точке положения объекта Fox, локальные координаты объекта $Rabbit$
17	Если объект $Rabbit$ находится в верхней полуплоскости локального базиса, то вектор скорости объекта Fox необходимо вращать против часовой стрелки с шагом на расстояние, соответствующее временному промежутку
18	Если объект $Rabbit$ находится в нижней полуплоскости локального базиса, то вектор скорости объекта Fox необходимо вращать по часовой стрелке с шагом на расстояние, соответствующее временному промежутку
20	Присвоение объекту Fox_1 текущих координат
21	Присвоение объекту Fox_2 текущих координат
22	Присвоение объекту Fox_3 текущих координат

23	Присвоение объекту Fox_4 текущих координат
24	Вычисление текущего расстояния между объектами $Rabbit$ и Fox
25	Увеличение счетчика циклов на единицу
26	Полное число циклов
27	Формирование матрицы выходных параметров

$$\left(r_{rabbit}\ \ r_{fox}\ \ Total\ \ r_{fox.1}\ \ r_{fox.2}\ \ r_{fox.3}\ \ r_{fox.4}\right)$$

У функции $Situation$ выходными параметрами служат: массив точек траектории объекта $Rabbit$, массив точек траектории объекта Fox, общее число циклов, в дальнейшем это будет число кадров при изготовлении анимированного изображения, массив точек траектории объекта Fox_1, массив точек траектории объекта Fox_2, массив точек траектории объекта Fox_3 и массив точек траектории объекта Fox_4.

$$FrameNumber := \left(Situation^{T}\right)_2 \quad FrameNumber = 233$$

Общее число кадров.

$$i := 0..\ FrameNumber$$

Ранжированная переменная по числу кадров.

$$Rabbit_{x_i} := \left[\left[\left(Situation^{T}\right)_0\right]_i\right]_0 \quad Rabbit_{y_i} := \left[\left[\left(Situation^{T}\right)_0\right]_i\right]_1$$

Разделение по координатам массива точек траектории объекта $Rabbit$.

$$Fox_{x_i} := \left[\left[\left(Situation^{T}\right)_1\right]_i\right]_0 \quad Fox_{y_i} := \left[\left[\left(Situation^{T}\right)_1\right]_i\right]_1$$

Разделение по координатам массива точек траектории объекта Fox.

$$Fox_{1.x_i} := \left[\left[\left(Situation^{T}\right)_3\right]_i\right]_0 \quad Fox_{1.y_i} := \left[\left[\left(Situation^{T}\right)_3\right]_i\right]_1$$

Разделение по координатам массива точек траектории объекта Fox_1.

$$Fox_{2.x_i} := \left[\left[\left(Situation^{T}\right)_4\right]_i\right]_0 \quad Fox_{2.y_i} := \left[\left[\left(Situation^{T}\right)_4\right]_i\right]_1$$

Разделение по координатам массива точек траектории объекта Fox_2.

$$Fox_{3.x_i} := \left[\left[\left(Situation^{T}\right)_5\right]_i\right]_0 \quad Fox_{3.y_i} := \left[\left[\left(Situation^{T}\right)_5\right]_i\right]_1$$

Разделение по координатам массива точек траектории объекта Fox_3.

$$Fox_{4.x_i} := \left[\left[\left(Situation^T\right)_6\right]_i\right]_0 \quad Fox_{4.y_i} := \left[\left[\left(Situation^T\right)_6\right]_i\right]_1$$

Разделение по координатам массива точек траектории объекта Fox_4.

$$X_{rabbit} := Rabbit_{x_{FRAME}} \quad Y_{rabbit} := Rabbit_{y_{FRAME}}$$

Координаты объекта $Rabbit$, соответствующие кадру $FRAME$.

$$X_{fox} := Fox_{x_{FRAME}} \quad Y_{fox} := Fox_{y_{FRAME}}$$

Координаты объекта Fox, соответствующие кадру $FRAME$.

$$X_{1.fox} := Fox_{1.x_{FRAME}} \quad Y_{1.fox} := Fox_{1.y_{FRAME}}$$

Координаты объекта Fox_1, соответствующие кадру $FRAME$.

$$X_{2.fox} := Fox_{2.x_{FRAME}} \quad Y_{2.fox} := Fox_{2.y_{FRAME}}$$

Координаты объекта Fox_2, соответствующие кадру $FRAME$.

$$X_{3.fox} := Fox_{3.x_{FRAME}} \quad Y_{3.fox} := Fox_{3.y_{FRAME}}$$

Координаты объекта Fox_3, соответствующие кадру $FRAME$.

$$X_{4.fox} := Fox_{4.x_{FRAME}} \quad Y_{4.fox} := Fox_{4.y_{FRAME}}$$

Координаты объекта Fox_4, соответствующие кадру $FRAME$.

Функцией $Trajectory$ создается на основе вектор - столбца создается диагональная матрица. Пример работы функции $Trajectory$ описывается в параграфе 3.4.

$$\begin{aligned} &Traectory(animal) := \\ &\quad Q \leftarrow diag(animal) \\ &\quad \text{for } j \in 1..FrameNumber \\ &\quad\quad \text{for } i \in 0..j \\ &\quad\quad\quad Q_{i,j} \leftarrow animal_i \\ &\quad Q \end{aligned}$$

Далее, в программе следует операторы, выводящие сегменты траектории объекта $Rabbit$ от начала итерационного процесса до значения времени, соответствующего значению переменной $Kadr$.

$$RabbitTraectory_x(Kadr) := submatrix\left(Traectory\left(Rabbit_x\right), 0, Kadr, Kadr, Kadr\right)$$

$$RabbitTraectory_y(Kadr) := submatrix\left(Traectory\left(Rabbit_y\right), 0, Kadr, Kadr, Kadr\right)$$

Операторы, выводящие сегменты траектории объекта Fox от начала итерационного процесса до значения времени, соответствующего значению переменной $Kadr$.

$$FoxTraectory_x(Kadr) := submatrix\left(Traectory\left(Fox_x\right), 0, Kadr, Kadr, Kadr\right)$$

$$FoxTraectory_y(Kadr) := submatrix\left(Traectory\left(Fox_y\right), 0, Kadr, Kadr, Kadr\right)$$

Вывод на экран значения общего числа кадров.

$$FrameNumber = 233$$

Вывод на экран модуля скорости объекта $Rabbit$.

$$V_{rabbit} = 8$$

Вывод на экран модуля скорости объекта Fox.

$$V_{fox} = 15$$

Вывод на экран угловой частоты вращения объекта $Rabbit$.

$$\omega_{rabbit} = 0.524$$

Вывод на экран угловой частоты вращения объекта Fox.

$$\omega_{fox} = 0.785$$

Вывод на экран вектора скорости объекта $Rabbit$.

$$V_{Rabbit0} = \begin{pmatrix} 7.727 \\ 2.071 \end{pmatrix}$$

Вывод на экран модуля скорости объекта Fox.

$$V_{Fox0} = \begin{pmatrix} -10.607 \\ -10.607 \end{pmatrix}$$

Стартовая позиция объекта $Rabbit$.

$$R_{Rabbit0} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Стартовая позиция объекта Fox.

$$R_{Fox0} = \begin{pmatrix} 0 \\ -25 \end{pmatrix}$$

На рисунке 3.26 приведен итоговый результат работы программы преследования группой одной цели с жесткими связями. Под жесткими связями мы понимаем постоянную ориентацию объектов Fox_1, Fox_2, Fox_3 и Fox_4 в локальной системе координат, образованной вектором скорости объекта Fox.

Вектор скорости объекта Fox сонаправлен вектору абсцисс локальной системы координат. Вектор ординат локальной системы координат, соответственно, перпендикулярен вектору скорости объекта Fox.

Рисунок 3.26 дополнен ссылкой на анимированное изображение [54], где в динамике можно будет посмотреть результаты работы программы.

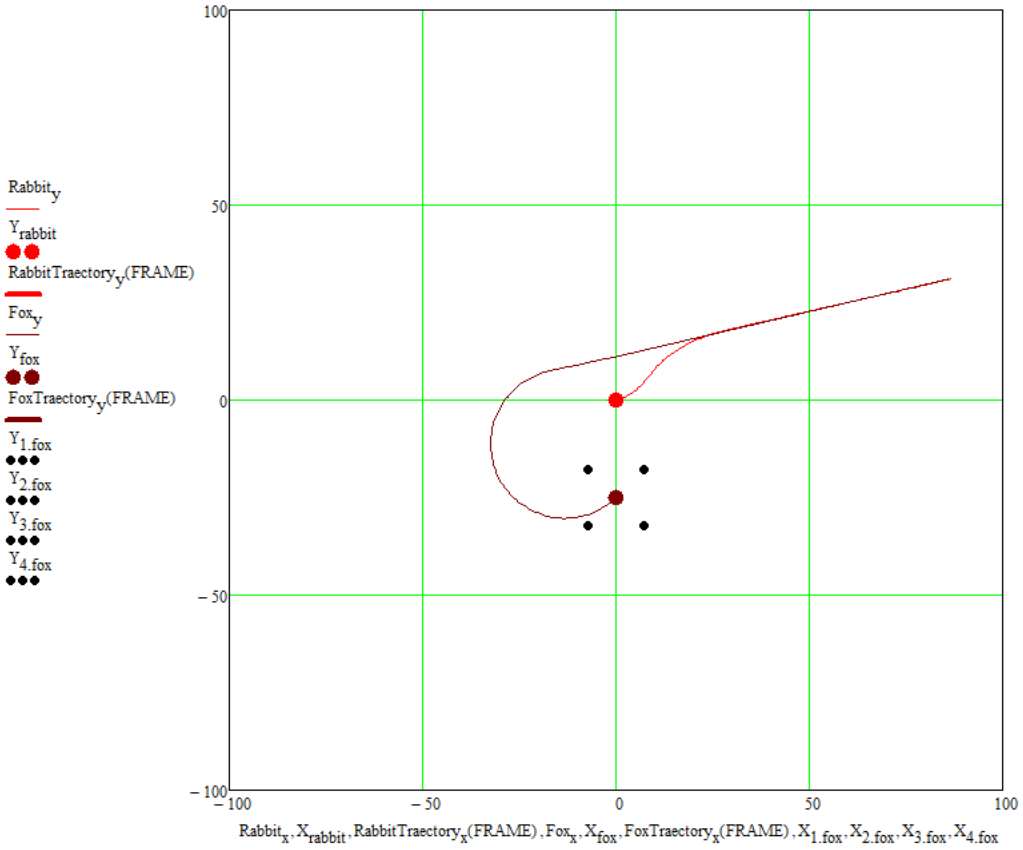


Рисунок 3.26. Результаты работы программы преследования группой одиночной цели с фиксированными связями

3.6 КОММЕНТАРИИ К ПРОГРАММЕ ОДНОВРЕМЕННОГО ДОСТИЖЕНИЯ ГРУППОЙ ПРЕСЛЕДОВАТЕЛЕЙ ГРУППЫ ЦЕЛЕЙ

В данной программе решалась задача одновременного достижения двух целей тремя преследователями. Для каждого из преследователей строился свой расчетный алгоритм для выполнения своих задач.

На рисунке 3.27 показано, что преследователи $Persuer\ 1$ и $Persuer\ 2$ преследуют цель $Target\ 1$, а преследователь $Persuer\ 3$ преследует цель $Target\ 2$.

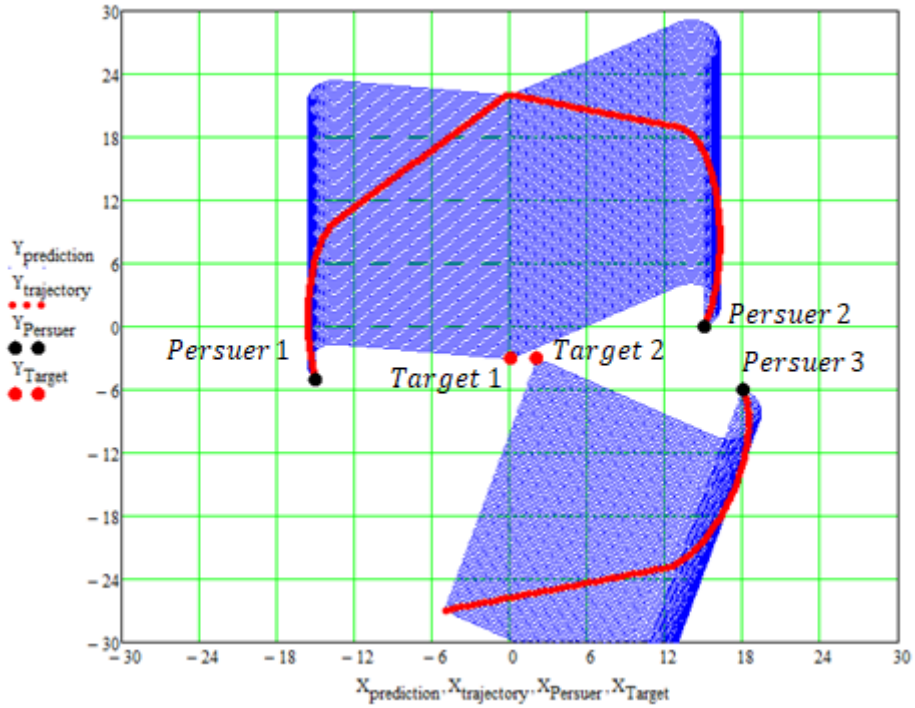


Рисунок 3.27. Преследование множества целей множеством преследователей

Каждый расчетный алгоритм подсчитывает количество дискретных временных промежутков, за которое каждый преследователь выполняет свою задачу.

Далее, выбирается наибольшее время, кто из трех преследователей достиг своей цели. Это можно назвать предварительным прогоном программы. Остальные два участника модифицируют свой расчетный алгоритм, что достичь своих целей за время, выбранное эталонным.

Поскольку в программе для преследователей была выбрана стратегия следования прогнозируемым траекториям, то в качестве модификации расчетных алгоритмов нами предложен способ увеличивающий радиус кривизны первоначального участка прогнозируемых траекторий.

Как второй способ рассматривалось просто увеличение модуля скорости движения преследователя при сохранении ограничений на кривизну прогнозируемых траекторий.

Программа является тестовой и демонстрационной, поэтому в ней много чего нуждается в улучшении и оптимизации. Мы в ней хотели реализовать математическую модель, изложенную в параграфе 3.3.

Инициализация стартовых значений участников задачи преследования.

$$Persuer_{1.start} := \begin{pmatrix} -15 \\ -5 \end{pmatrix}$$ Стартовая позиция преследователя $Persuer$ 1

$$Persuer_{2.start} := \begin{pmatrix} 15 \\ 0 \end{pmatrix}$$ Стартовая позиция преследователя $Persuer$ 2

$$Persuer_{3.start} := \begin{pmatrix} 18 \\ -6 \end{pmatrix}$$ Стартовая позиция преследователя $Persuer$ 3

$$Target_{1.start} := \begin{pmatrix} 0 \\ -3 \end{pmatrix}$$ Стартовая позиция преследуемого объекта $Target$ 1

$$Target_{2.start} := \begin{pmatrix} 2 \\ -3 \end{pmatrix}$$ Стартовая позиция преследуемого объекта $Target$ 2

$$n_{1.p.start} := \frac{1}{\sqrt{1+1}} \cdot \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$ Направление движения преследователя $Persuer$ 1

$$n_{2.p.start} := \frac{1}{\sqrt{2^2+1^2}} \cdot \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$ Направление движения преследователя $Persuer$ 2

$$n_{3.p.start} := \frac{1}{\sqrt{2^2+1^2}} \cdot \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$ Направление движения преследователя $Persuer$ 3

$V_{1.Persuer} := 20$ Модуль скорости преследователя $Persuer$ 1

$V_{2.Persuer} := 20$ Модуль скорости преследователя $Persuer$ 2

$V_{3.Persuer} := 22$ Модуль скорости преследователя $Persuer$ 3

$$n_{1.t.start} := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$ Направление движения преследователя $Target$ 1

$$v_2 := \begin{pmatrix} -0.3 \\ -1 \end{pmatrix} \quad n_{2.t.start} := \frac{v_2}{|v_2|}$$ Направление движения преследователя $Target$ 2

$V_{1.Target} := 15$ Модуль скорости преследователя $Target$ 1

$V_{2.Target} := 15$ Модуль скорости преследователя $Target$ 2

Векторы направления движений для вывода на экран. Данная визуализация носит тестовый характер. На наш взгляд программирование в любом пакете компьютерной математики является не только решением задачи, но и исследовательской работой.

$i := 0..25$

Ранжированная переменная для формирования массива точек

$$vec_{n.1.x_i} := \frac{i}{25} \cdot n_{1.p.start_0} + Persuer_{1.start_0}$$

$$vec_{n.1.y_i} := \frac{i}{25} \cdot n_{1.p.start_1} + Persuer_{1.start_1}$$

Массив точек вдоль направления движения объекта $Persuer\ 1$

$$vec_{n.2.x_i} := \frac{i}{25} \cdot n_{2.p.start_0} + Persuer_{2.start_0}$$

$$vec_{n.2.y_i} := \frac{i}{25} \cdot n_{2.p.start_1} + Persuer_{2.start_1}$$

Массив точек вдоль направления движения объекта $Persuer\ 2$

$$vec_{n.3.x_i} := \frac{i}{25} \cdot n_{3.p.start_0} + Persuer_{3.start_0}$$

$$vec_{n.3.y_i} := \frac{i}{25} \cdot n_{3.p.start_1} + Persuer_{3.start_1}$$

Массив точек вдоль направления движения объекта $Persuer\ 3$

Первоначальный радиус допустимой кривизны траекторий.

$$R_{curvature} := 2$$

Далее, нам понадобится функция расчета центров окружностей.

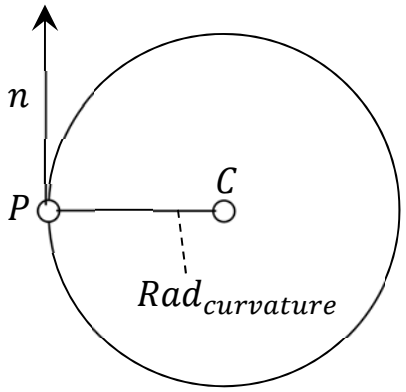


Рисунок 3.28. Расчет центров окружностей

При заданных значениях точки P, вектора n и радиуса $Rad_{curvature}$ следует найти координаты точки C при соблюдении следующих условий:

$$|P - C| = Rad_{curvature}$$
$$(P - C) \cdot n = 0$$

Модуль вектора $P - C$ равен радиусу окружности $Rad_{curvature}$ и вектор $P - C$ перпендикулярен вектору n.

Given

$$\left[\begin{pmatrix} C_x \\ C_y \end{pmatrix} - \begin{pmatrix} P_x \\ P_y \end{pmatrix}\right]^T \cdot \left[\begin{pmatrix} C_x \\ C_y \end{pmatrix} - \begin{pmatrix} P_x \\ P_y \end{pmatrix}\right] = Rad_{curvature}^{\ 2}$$

$$\left[\begin{pmatrix} C_x \\ C_y \end{pmatrix} - \begin{pmatrix} P_x \\ P_y \end{pmatrix}\right]^T \cdot \begin{pmatrix} n_x \\ n_y \end{pmatrix} = 0$$

$$Center_{cicle}(P_x, P_y, n_x, n_y, Rad_{curvature}) := Find(C_x, C_y) \rightarrow$$

$$\begin{bmatrix} \dfrac{P_x \cdot n_x - n_y \cdot \left(P_y + Rad_{curvature} \cdot n_x \cdot \sqrt{\dfrac{1}{n_x^{\ 2} + n_y^{\ 2}}}\right) + P_y \cdot n_y}{n_x} & \dfrac{P_x \cdot n_x - n_y \cdot \left(P_y - Rad_{curvature} \cdot n_x \cdot \sqrt{\dfrac{1}{n_x^{\ 2} + n_y^{\ 2}}}\right) + P_y \cdot n_y}{n_x} \\ P_y + Rad_{curvature} \cdot n_x \cdot \sqrt{\dfrac{1}{n_x^{\ 2} + n_y^{\ 2}}} & P_y - Rad_{curvature} \cdot n_x \cdot \sqrt{\dfrac{1}{n_x^{\ 2} + n_y^{\ 2}}} \end{bmatrix}$$

Как видим, система MathCAD выдает символьный результат.

Создадим функцию расчета точки P_t касания прямой (PP_t) и окружности (C, rad) (Рисунок 3.29).

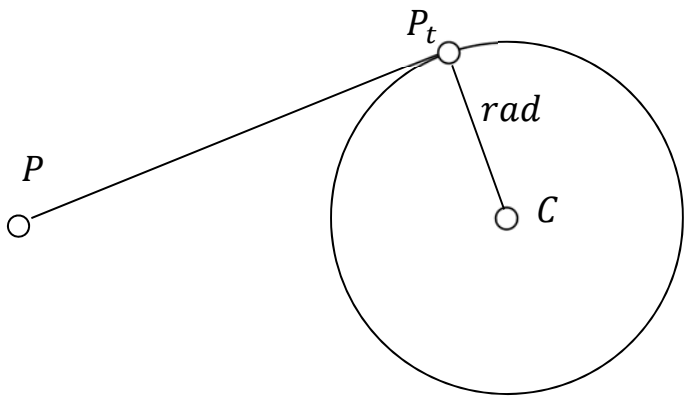


Рисунок 3.29. Расчет точки касания

Расчет точки касания P_t производится из следующих условий:

$$(P - P_t) \cdot (C - P_t) = 0$$
$$|P_t - C| = rad$$

Модуль вектора $P_t - C$ равен радиусу окружности rad и вектор $P - P_t$ перпендикулярен вектору $C - P_t$.

В этом случае система MathCAD не смогла выдать символьный результат, что не мешает использовать созданную функцию.

Given

$$\left[\begin{pmatrix} P_{t.x} \\ P_{t.y} \end{pmatrix} - \begin{pmatrix} C_0 \\ C_1 \end{pmatrix}\right]^T \cdot \left[\begin{pmatrix} P_{t.x} \\ P_{t.y} \end{pmatrix} - \begin{pmatrix} C_0 \\ C_1 \end{pmatrix}\right] = rad^2$$

$$\left[\begin{pmatrix} P_{t.x} \\ P_{t.y} \end{pmatrix} - \begin{pmatrix} C_0 \\ C_1 \end{pmatrix}\right]^T \cdot \left[\begin{pmatrix} P_0 \\ P_1 \end{pmatrix} - \begin{pmatrix} P_{t.x} \\ P_{t.y} \end{pmatrix}\right] = 0$$

$$Tangent(P, C, rad) := Find(P_{t.x}, P_{t.y}) \rightarrow$$

Расчет точек касания для объектов $Persuer\ 1$, $Persuer\ 2$ и $Persuer\ 3$.

$$P_{tan.1} := Tangent$$

$$\left[Target_{1.start}, \left(Center_{cicle}\left(Persuer_{1.start_0}, Persuer_{1.start_1}, n_{1.p.start_0}, n_{1.p.start_1}, R_{curvature}\right)^{\langle 1 \rangle}\right), R_{curvature}\right]^{\langle 1 \rangle} = \begin{pmatrix} -13.377 \\ -1.597 \end{pmatrix}$$

$$P_{tan.2} := Tangent$$

$$\left[Target_{1.start}, \left(Center_{cicle}\left(Persuer_{2.start_0}, Persuer_{2.start_1}, n_{2.p.start_0}, n_{2.p.start_1}, R_{curvature}\right)^{\langle 0 \rangle}\right), R_{curvature}\right]^{\langle 0 \rangle} = \begin{pmatrix} 13.214 \\ 3.579 \end{pmatrix}$$

$$P_{tan.3} := Tangent$$

$$\left(Target_{2.start}, Center_{cicle}\left(Persuer_{3.start_0}, Persuer_{3.start_1}, n_{3.p.start_0}, n_{3.p.start_1}, R_{curvature}\right)^{\langle 1 \rangle}, R_{curvature}\right)^{\langle 1 \rangle} = \begin{pmatrix} 16.265 \\ -9.604 \end{pmatrix}$$

Задание касательных прямых.

$$L_{1.tan}(t) := (1 - t) \cdot P_{tan.1} + t \cdot Target_{1.start}$$
$$L_{2.tan}(t) := (1 - t) \cdot P_{tan.2} + t \cdot Target_{1.start}$$

$$L_{3.tan}(t) := (1 - t)\cdot P_{tan.3} + t\cdot Target_{2.start}$$

Для объектов $Persuer\ 1$ и $Persuer\ 2$ расчет точки касания ведется из условия, что касательная прямая проходит через точку $Target\ 1$ (Рисунок 3.30). Для объекта $Persuer\ 3$ расчет точки касания ведется из условия, что касательная прямая проходит через точку $Target\ 2$.

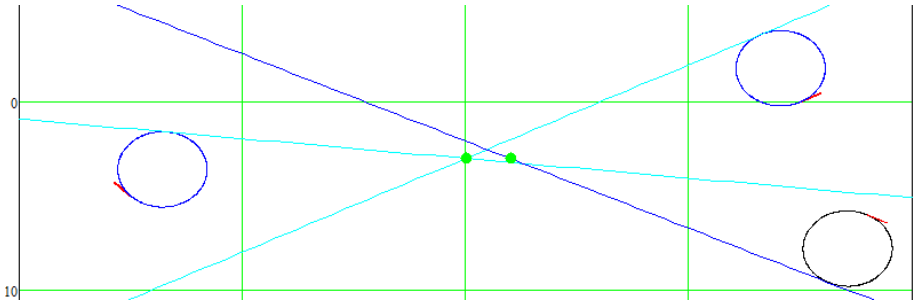


Рисунок 3.30. Расчет касательных

Мы имеем сейчас для отдельно взятого преследователя окружность. Для дальнейшей работы нам необходимо сегмент окружности и прямолинейный сегмент представить в виде единой составной кривой (Рисунок 3.31).

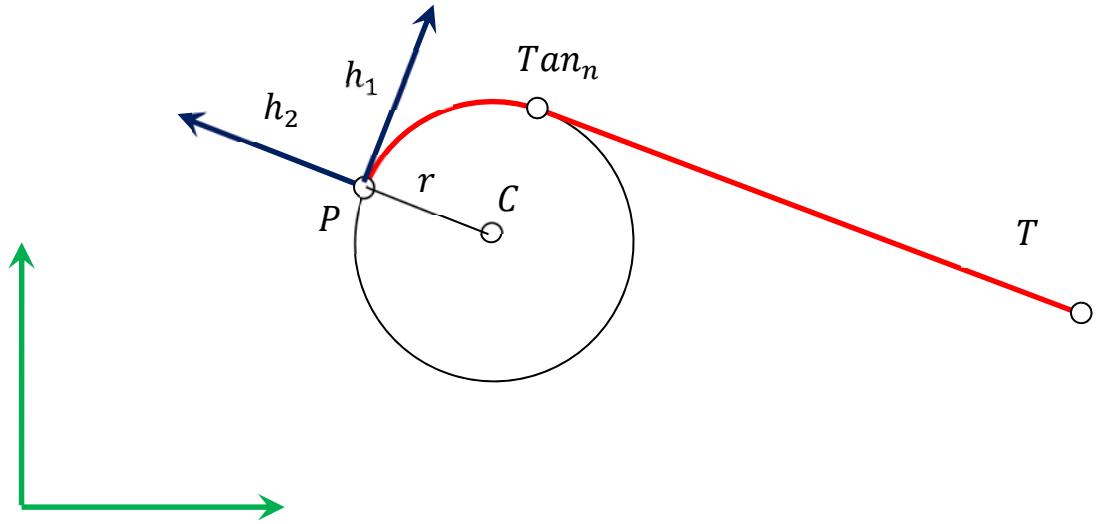


Рисунок 3.31. Получение единого массива точек составной кривой

Функция $Syst_1$ написана для того, чтобы получить упорядоченный массив точек

$\text{Syst}_1(n, r, P, T) :=$

$h_1 \leftarrow n$

$h_2 \leftarrow \begin{pmatrix} -n_1 \\ n_0 \end{pmatrix}$

Формирование локального базиса. Вектор h_1 сонаправлен с направлением движения точки P.

$T_n \leftarrow \begin{bmatrix} (T - P)^T \cdot h_1 \\ (T - P)^T \cdot h_2 \end{bmatrix}$

Перевод точки T в систему координат (h_1, P, h_2).

$C \leftarrow \begin{pmatrix} 0 \\ -r \end{pmatrix}$

В системе координат (h_1, P, h_2) центр окружности имеет такие координаты.

$\text{Tan}_n \leftarrow \text{Tangent}(T_n, C, r)^{\langle 1 \rangle}$

Расчет касательной точки Tan_n в системе координат (h_1, P, h_2).

$f(t) \leftarrow r \cdot \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix} + C$

Определение окружности радиуса r с центром в точке C в системе координат (h_1, P, h_2).

$\alpha_f \leftarrow \text{acos}\left[\frac{(-C)^T \cdot (\text{Tan}_n - C)}{|(-C)| \cdot |\text{Tan}_n - C|} \right]$

Определение угла $\widehat{PCTan}_n$

for $i \in 0..25$

$\alpha_i \leftarrow \frac{i}{25} \cdot \alpha_f$

$x1_i \leftarrow r \cdot \cos\left(\frac{\pi}{2} - \alpha_i\right) + C_0$

$y1_i \leftarrow r \cdot \sin\left(\frac{\pi}{2} - \alpha_i\right) + C_1$

Получение в системе координат (h_1, P, h_2) массива точек в растворе угла от 0 до α_f.

$s_1 \leftarrow r \cdot \alpha_f$

Длина дуги от точки P до точки Tan_n.

$$\begin{aligned} &t_i \leftarrow \frac{i}{50} \\ &x2_i \leftarrow (1 - t_i)\cdot Tan_{n_0} + t_i \cdot T_{n_0} \\ &y2_i \leftarrow (1 - t_i)\cdot Tan_{n_1} + t_i \cdot T_{n_1} \end{aligned}$$

Получение в системе координат (h_1, P, h_2) массива точек прямолинейного сегмента от точки Tan_n до точки T.

$$s_2 \leftarrow |Tan_n - T_n|$$

Длина отрезка $[Tan_n T]$.

$$s \leftarrow s_1 + s_2$$

Суммарная длина составной кривой.

$$\begin{aligned} &x \leftarrow \text{stack}(x_1, \text{submatrix}(x_2, 1, 50, 0, 0)) \\ &y \leftarrow \text{stack}(y_1, \text{submatrix}(y_2, 1, 50, 0, 0)) \end{aligned}$$

Покоординатное объединение массивов точек.

$$\begin{aligned} &E_1 \leftarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &E_2 \leftarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{aligned}$$

Базисные векторы мировой системы координат.

$$\begin{aligned} &e_1 \leftarrow \begin{pmatrix} E_1^T \cdot h_1 \\ E_1^T \cdot h_2 \end{pmatrix} \\ &e_2 \leftarrow \begin{pmatrix} E_2^T \cdot h_1 \\ E_2^T \cdot h_2 \end{pmatrix} \end{aligned}$$

Преобразование базисных векторов мировой системы координат в систему координат (h_1, P, h_2).

$$\begin{aligned} &\text{for } i \in 0..\text{rows}(x) - 1 \\ &\quad x_{new_i} \leftarrow \begin{pmatrix} x_i \\ y_i \end{pmatrix}^T \cdot e_1 + P_0 \\ &\quad y_{new_i} \leftarrow \begin{pmatrix} x_i \\ y_i \end{pmatrix}^T \cdot e_2 + P_1 \end{aligned}$$

Перевод массива точек составной кривой в мировую систему координат.

$$(x_{new} \;\; y_{new} \;\; s)$$

Формирование матрицы выходных значений

У функции $Syst_1$ входными переменными являются единичный вектор n направления движения точки P, радиус допустимой кривизны r, точка преследователя P и точка преследуемого объекта T.

Функция $Syst_2$ выполняет аналогичные действия, только точка C располагается зеркально относительно вектора n.

$\text{Syst}_2(n, r, P, T) :=$

$h_1 \leftarrow n$

$h_2 \leftarrow \begin{pmatrix} -n_1 \\ n_0 \end{pmatrix}$

Формирование локального базиса. Вектор h_1 сонаправлен с направлением движения точки P.

$T_n \leftarrow \begin{bmatrix} (T - P)^T \cdot h_1 \\ (T - P)^T \cdot h_2 \end{bmatrix}$

Перевод точки T в систему координат (h_1, P, h_2).

$C \leftarrow \begin{pmatrix} 0 \\ r \end{pmatrix}$

В системе координат (h_1, P, h_2) центр окружности имеет такие координаты.

$\text{Tan}_n \leftarrow \text{Tangent}(T_n, C, r)^{\langle 1 \rangle}$

Расчет касательной точки Tan_n в системе координат (h_1, P, h_2).

$f(t) \leftarrow r \cdot \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix} + C$

Определение окружности радиуса r с центром в точке C в системе координат (h_1, P, h_2).

$\alpha_f \leftarrow \text{acos}\left[\frac{(-C)^T \cdot (\text{Tan}_n - C)}{|(-C)| \cdot |\text{Tan}_n - C|} \right]$

Определение угла $\widehat{PCTan_n}$

for $i \in 0..25$

$\alpha_i \leftarrow \frac{i}{25} \cdot \alpha_f$

$x1_i \leftarrow r \cdot \cos\left(\frac{\pi}{2} - \alpha_i\right) + C_0$

$y1_i \leftarrow r \cdot \sin\left(\frac{\pi}{2} - \alpha_i\right) + C_1$

Получение в системе координат (h_1, P, h_2) массива точек в растворе угла от 0 до α_f.

$s_1 \leftarrow r \cdot \alpha_f$

Длина дуги от точки P до точки Tan_n.

$$t_i \leftarrow \frac{i}{50}$$
$$x2_i \leftarrow (1 - t_i)\cdot Tan_{n_0} + t_i \cdot T_{n_0}$$
$$y2_i \leftarrow (1 - t_i)\cdot Tan_{n_1} + t_i \cdot T_{n_1}$$

Получение в системе координат (h_1, P, h_2) массива точек прямолинейного сегмента от точки Tan_n до точки T.

$$s_2 \leftarrow |Tan_n - T_n|$$

Длина отрезка $[Tan_n T]$.

$$s \leftarrow s_1 + s_2$$

Суммарная длина составной кривой.

$$x \leftarrow stack(x_1, submatrix(x_2, 1, 50, 0, 0))$$
$$y \leftarrow stack(y_1, submatrix(y_2, 1, 50, 0, 0))$$

Покоординатное объединение массивов точек.

$$E_1 \leftarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$E_2 \leftarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Базисные векторы мировой системы координат.

$$e_1 \leftarrow \begin{pmatrix} E_1^T \cdot h_1 \\ E_1^T \cdot h_2 \end{pmatrix}$$
$$e_2 \leftarrow \begin{pmatrix} E_2^T \cdot h_1 \\ E_2^T \cdot h_2 \end{pmatrix}$$

Преобразование базисных векторов мировой системы координат в систему координат (h_1, P, h_2).

for $i \in 0..rows(x) - 1$

$$x_{new_i} \leftarrow \begin{pmatrix} x_i \\ y_i \end{pmatrix}^T \cdot e_1 + P_0$$
$$y_{new_i} \leftarrow \begin{pmatrix} x_i \\ y_i \end{pmatrix}^T \cdot e_2 + P_1$$

Перевод массива точек составной кривой в мировую систему координат.

$$(x_{new} \quad y_{new} \quad s)$$

Формирование матрицы выходных значений

Расчет стартовой прогнозируемой линии для преследователя $Persuer\ 1$.

$$dr_1 := 0$$
$$X_1 := Syst_1(n_{1.p.start}, R_{curvature} + dr_1, Persuer_{1.start}, Target_{1.start})_{0,0}$$

$$Y_1 := Syst_1\left(n_{1.p.start}, R_{curvature} + dr_1, Persuer_{1.start}, Target_{1.start}\right)_{0,1}$$

Расчет стартовой прогнозируемой линии для преследователя $Persuer\ 2$.

$$dr_2 := 0.2$$

$$X_2 := Syst_2\left(n_{2.p.start}, R_{curvature} + dr_2, Persuer_{2.start}, Target_{1.start}\right)_{0,0}$$

$$Y_2 := Syst_2\left(n_{2.p.start}, R_{curvature} + dr_2, Persuer_{2.start}, Target_{1.start}\right)_{0,1}$$

Расчет стартовой прогнозируемой линии для преследователя $Persuer\ 3$.

$$dr_3 := 0.5$$

$$X_3 := Syst_1\left(n_{3.p.start}, R_{curvature} + dr_3, Persuer_{3.start}, Target_{2.start}\right)_{0,0}$$

$$Y_3 := Syst_1\left(n_{3.p.start}, R_{curvature} + dr_3, Persuer_{3.start}, Target_{2.start}\right)_{0,1}$$

Следует обратить внимание на значение переменных dr_1, dr_2 и dr_3. Это переменные модификации допустимой кривизны траекторий преследователей. Эти переменные после пробного прогона были настроены на время достижения цели $Target\ 1$ преследователем $Persuer\ 1$. В следствии чего, все три преследователя достигают своих целей одновременно.

На рисунке 3.32 представлены стартовые линии прогнозируемых траекторий движения преследователей, настроенных на достижение своих целей одновременно.

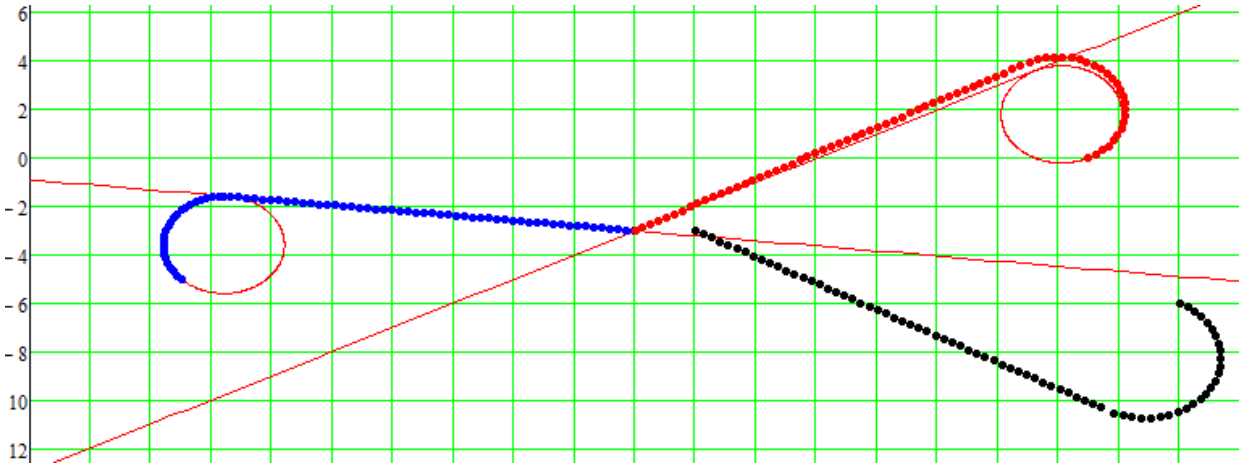


Рисунок 3.32. Стартовые прогнозируемые линии движения, настроенные на достижение своих целей одновременно

Далее, создаем функцию $Diff_{Point}$ расчета узловых производных от формального параметра.

Создание ранжированной переменной по числу элементов массивов. Число элементов во всех массивах траекторий одинаково. Мы взяли X_1

$$i := 0.. \text{rows}(X_1) - 1$$

Определение формального параметра

$$\text{Formal}_{\text{Parameter}_i} := i$$

Шаг формального параметра

$$\text{Formal}_{\text{Parameter}_1} - \text{Formal}_{\text{Parameter}_0} = 1$$

Число элементов в массиве X_1

$$\text{rows}(X_1) = 76$$

Число элементов массива формального параметра

$$\text{rows}(\text{Formal}_{\text{Parameter}}) = 76$$

Функция численного дифференцирования от формального параметра

$$\begin{aligned}
&\text{Diff}_{\text{Point}}(x,y) := \left|\begin{array}{l}
D \leftarrow (0 \;\; 0) \\
\text{for } i \in 0.. \text{rows}(x) - 1 \\
\quad \left|\begin{array}{l}
\text{if } i = 0 \\
\quad \left|\begin{array}{l} Dx \leftarrow x_{i+1} - x_i \\ Dy \leftarrow y_{i+1} - y_i \end{array}\right. \\
\text{if } i = \text{rows}(\text{Formal}_{\text{Parameter}}) - 1 \\
\quad \left|\begin{array}{l} Dx \leftarrow x_i - x_{i-1} \\ Dy \leftarrow y_i - y_{i-1} \end{array}\right. \\
\text{if } (i \neq 0) \wedge (i \neq \text{rows}(X_1) - 1) \\
\quad \left|\begin{array}{l} Dx \leftarrow \dfrac{x_{i+1} - x_{i-1}}{2} \\ Dy \leftarrow \dfrac{y_{i+1} - y_{i-1}}{2} \end{array}\right. \\
D \leftarrow \text{stack}[D, (Dx \;\; Dy)]
\end{array}\right. \\
D \leftarrow \text{submatrix}(D, 1, \text{rows}(\text{Formal}_{\text{Parameter}}), 0, 1)
\end{array}\right.
\end{aligned}$$

Сплайн – интерполяция массивов траекторий преследователей для получения непрерывных функций от формального параметра t.

$x_1(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, X_1\right), \text{Formal}_{\text{Parameter}}, X_1, t\right)$
$y_1(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, Y_1\right), \text{Formal}_{\text{Parameter}}, Y_1, t\right)$

Координатные функции преследователя *Persuer* 1

$x_2(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, X_2\right), \text{Formal}_{\text{Parameter}}, X_2, t\right)$
$y_2(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, Y_2\right), \text{Formal}_{\text{Parameter}}, Y_2, t\right)$

Координатные функции преследователя *Persuer* 2

$x_3(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, X_3\right), \text{Formal}_{\text{Parameter}}, X_3, t\right)$
$y_3(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, Y_3\right), \text{Formal}_{\text{Parameter}}, Y_3, t\right)$

Координатные функции преследователя *Persuer* 3

На рисунке 3.33 представлены стартовые траектории в зависимости от t.

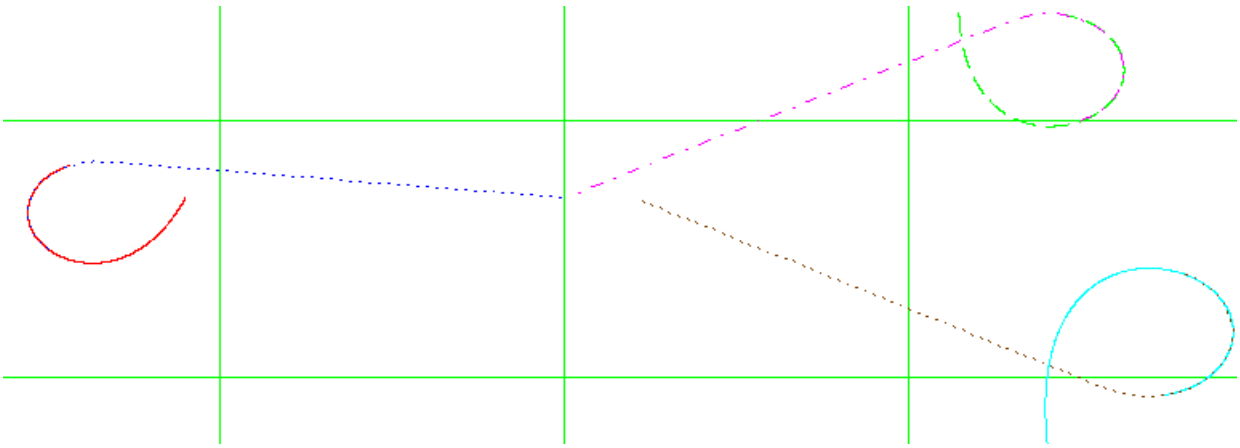


Рисунок 3.33. Функции стартовых траекторий

Координатные функции первых производных после сплайн-интерполяции от непрерывного формального параметра t.

Координатные функции первой производной преследователя *Persuer* 1 от формального параметра t

$dx_1(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_1, Y_1\right)^{\langle 0 \rangle}\right), \text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_1, Y_1\right)^{\langle 0 \rangle}, t\right)$
$dy_1(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_1, Y_1\right)^{\langle 1 \rangle}\right), \text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_1, Y_1\right)^{\langle 1 \rangle}, t\right)$

Координатные функции первой производной преследователя *Persuer* 2 от формального параметра t

$dx_2(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_2, Y_2\right)^{\langle 0 \rangle}\right), \text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_2, Y_2\right)^{\langle 0 \rangle}, t\right)$
$dy_2(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_2, Y_2\right)^{\langle 1 \rangle}\right), \text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_2, Y_2\right)^{\langle 1 \rangle}, t\right)$

Координатные функции первой производной преследователя *Persuer* 3 от формального параметра t

$$dx_3(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_3, Y_3\right)^{\langle 0 \rangle}\right), \text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_3, Y_3\right)^{\langle 0 \rangle}, t\right)$$

$$dy_3(t) := \text{interp}\left(\text{cspline}\left(\text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_3, Y_3\right)^{\langle 1 \rangle}\right), \text{Formal}_{\text{Parameter}}, \text{Diff}_{\text{Point}}\left(X_3, Y_3\right)^{\langle 1 \rangle}, t\right)$$

Для того, чтобы ввести в программу параметр реального времени мы перейдем от формального параметра к параметру длины дуги. Для этого составим якобианы для каждого из преследователей.

$$Jc_1\left(s_1, t\right) := \frac{1}{\sqrt{\left(dx_1(t)\right)^2 + \left(dy_1(t)\right)^2}}$$

Якобиан преследователя $Persuer$ 1 от формального параметра и параметра своей длины дуги s_1

$$Jc_2\left(s_2, t\right) := \frac{1}{\sqrt{\left(dx_2(t)\right)^2 + \left(dy_2(t)\right)^2}}$$

Якобиан преследователя $Persuer$ 2 от формального параметра и параметра своей длины дуги s_2

$$Jc_3\left(s_3, t\right) := \frac{1}{\sqrt{\left(dx_3(t)\right)^2 + \left(dy_3(t)\right)^2}}$$

Якобиан преследователя $Persuer$ 3 от формального параметра и параметра своей длины дуги s_3

Стартовое значение параметра длины дуги для всех преследователей одинаково и равно 0. Число разбиений отрезков длин дуг.

$$\text{Start}_0 := 0 \quad \text{NumberOfKnot} := 200$$

Значения длин дуг от преследователя до цели.

$$S_1 := \text{Syst}_1\left(n_{1.p.start}, R_{curvature}, \text{Persuer}_{1.start}, \text{Target}_{1.start}\right)_{0,2} = 18.372$$

$$S_2 := \text{Syst}_2\left(n_{2.p.start}, R_{curvature} + dr_2, \text{Persuer}_{2.start}, \text{Target}_{1.start}\right)_{0,2} = 21.562$$

$$S_3 := \text{Syst}_1\left(n_{3.p.start}, R_{curvature} + dr_3, \text{Persuer}_{3.start}, \text{Target}_{2.start}\right)_{0,2} = 23.346$$

Далее, следует обращение к встроенной процедуре решения дифференциальных уравнений методом Рунге-Кутты 4 порядка с фиксированным шагом.

$$ARC_1 := \text{rkfixed}\left(\text{Start}_0, 0, S_1, \text{NumberOfKnot}, Jc_1\right)$$

В матрице ARC_1 в первом столбце хранится разбиение на 200 участков

параметра s_1, во втором – соответствующие значения параметра t

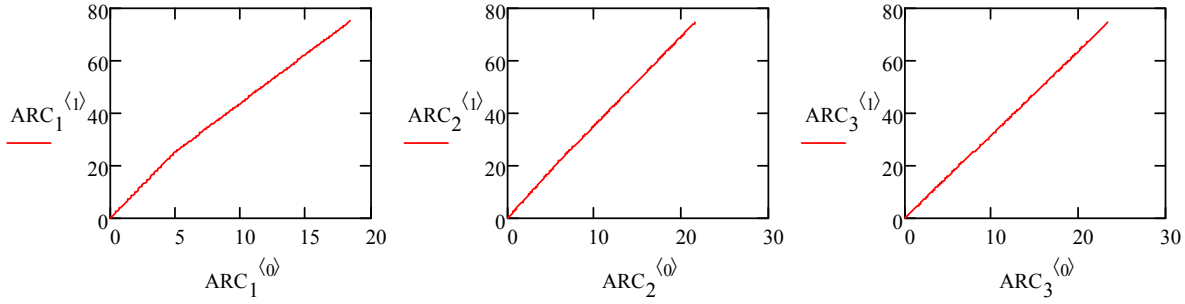
В матрице ARC_2 в первом столбце хранится разбиение на 200 участков параметра s_2, во втором – соответствующие значения параметра t

$$ARC_2 := rkfixed(Start_0, 0, S_2, NumberOfKnot, Jc_2)$$

В матрице ARC_3 в первом столбце хранится разбиение на 200 участков параметра s_3, во втором – соответствующие значения параметра t

$$ARC_3 := rkfixed(Start_0, 0, S_3, NumberOfKnot, Jc_3)$$

Для контроля над процессом вычисления мы вывели значения матриц ARC_1, ARC_2 и ARC_3 на графики.



Сплайн – интерполяция массива точек на основе матрицы ARC_1, чтобы получить непрерывную функцию $t(s_1)$, формального параметра t от параметра длины дуги траектории объекта $Persuer$ 1.

$$F_{t.1}(dist) := interp(cspline(ARC_1^{\langle 0 \rangle}, ARC_1^{\langle 1 \rangle}), ARC_1^{\langle 0 \rangle}, ARC_1^{\langle 1 \rangle}, dist)$$

Сплайн – интерполяция массива точек на основе матрицы ARC_2, чтобы получить непрерывную функцию $t(s_2)$, формального параметра t от параметра длины дуги траектории объекта $Persuer$ 2.

$$F_{t.2}(dist) := interp\left(cspline\left(ARC_2^{\langle 0\rangle}, ARC_2^{\langle 1\rangle}\right), ARC_2^{\langle 0\rangle}, ARC_2^{\langle 1\rangle}, dist\right)$$

Сплайн – интерполяция массива точек на основе матрицы ARC_3, чтобы получить непрерывную функцию $t(s_3)$, формального параметра t от параметра длины дуги траектории объекта $Persuer$ 3.

$$F_{t.3}(dist) := interp\left(cspline\left(ARC_3^{\langle 0\rangle}, ARC_3^{\langle 1\rangle}\right), ARC_3^{\langle 0\rangle}, ARC_3^{\langle 1\rangle}, dist\right)$$

Координатные функции траекторий объектов преследования от своих длин дуг.

$Pred_{1.x}(dist) := x_1\left(F_{t.1}(dist)\right)$ $Pred_{1.y}(dist) := y_1\left(F_{t.1}(dist)\right)$ Траектория $Persuer$ 1

$Pred_{2.x}(dist) := x_2\left(F_{t.2}(dist)\right)$ $Pred_{2.y}(dist) := y_2\left(F_{t.2}(dist)\right)$ Траектория $Persuer$ 2

$Pred_{3.x}(dist) := x_3\left(F_{t.3}(dist)\right)$ $Pred_{3.y}(dist) := y_3\left(F_{t.3}(dist)\right)$ Траектория $Persuer$ 3

На рисунке 3.33 представлены графики траекторий преследователей. Выведены для контроля за процессом вычислений.

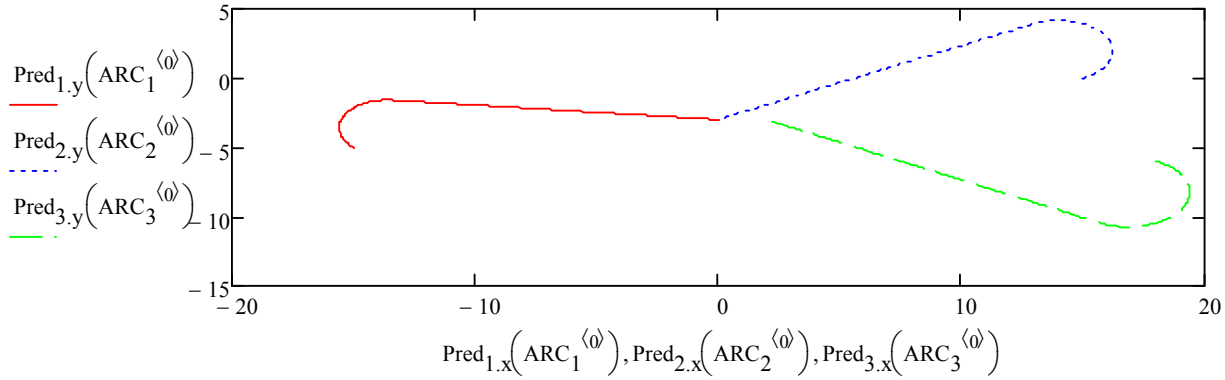


Рисунок 3.33. Графики траекторий преследователей

В нашей программе цели $Target$ 1 и $Target$ 2 движутся прямолинейно, что не имеет принципиального значения. Что является важным - это необходимо, чтобы траектории были непрерывными.

$Target1_0 := Target1_{1.start}$

$X_{1.Target_0} := \left(Target1_0\right)_0$ $Y_{1.Target_0} := \left(Target1_0\right)_1$

Инициализация траектории цели $Target$ 1

$Target2_0 := Target2_{2.start}$

$X_{2.Target_0} := \left(Target2_0\right)_0$ $Y_{2.Target_0} := \left(Target2_0\right)_1$

Инициализация траектории цели $Target$ 2

Для того, чтобы для каждого преследователя построить сеть из однопараметрического множества прогнозируемых траекторий движения как на рисунке 3.27, введем для базовых траекторий вектора смещения.

$$\text{Vector}_{1.\text{bias}}(\text{TT}) := \begin{pmatrix} \text{TT}_0 \\ \text{TT}_1 \end{pmatrix} - \text{Target}_{1.\text{start}}$$

Функция вектора смещения для преследователей $Persuer\ 1$ и $Persuer\ 2$

$$\text{Vector}_{2.\text{bias}}(\text{TT}) := \begin{pmatrix} \text{TT}_0 \\ \text{TT}_1 \end{pmatrix} - \text{Target}_{2.\text{start}}$$

Функция вектора смещения для преследователя $Persuer\ 3$

Семейство однопараметрического множества прогнозируемых траекторий движения для каждого из преследователей.

$$L_{B.1}(\text{dist}, \text{TT}) := \begin{pmatrix} \text{Pred}_{1.x}(\text{dist}) \\ \text{Pred}_{1.y}(\text{dist}) \end{pmatrix} + \text{Vector}_{1.\text{bias}}(\text{TT})$$

Однопараметрическое множество линий для объекта $Persuer\ 1$, векторная функция

$$L_{B.2}(\text{dist}, \text{TT}) := \begin{pmatrix} \text{Pred}_{2.x}(\text{dist}) \\ \text{Pred}_{2.y}(\text{dist}) \end{pmatrix} + \text{Vector}_{1.\text{bias}}(\text{TT})$$

Однопараметрическое множество линий для объекта $Persuer\ 2$, векторная функция

$$L_{B.3}(\text{dist}, \text{TT}) := \begin{pmatrix} \text{Pred}_{3.x}(\text{dist}) \\ \text{Pred}_{3.y}(\text{dist}) \end{pmatrix} + \text{Vector}_{2.\text{bias}}(\text{TT})$$

Однопараметрическое множество линий для объекта $Persuer\ 3$, векторная функция

Координатные массивы семейства линий для преследователей для значения времени, соответствующие значению переменной $FRAME$.

$$XX_1 := L_{B.1}\left(ARC_1^{\langle 0 \rangle}, \text{Target}1_{\text{FRAME}}\right)_0$$

$$YY_1 := L_{B.1}\left(ARC_1^{\langle 0 \rangle}, \text{Target}1_{\text{FRAME}}\right)_1$$

Массив точек координатных линий в определенный кадр объекта $Persuer\ 1$

$$XX_2 := L_{B.2}\left(ARC_2^{\langle 0 \rangle}, \text{Target}1_{\text{FRAME}}\right)_0$$

$$YY_2 := L_{B.2}\left(ARC_2^{\langle 0 \rangle}, \text{Target}1_{\text{FRAME}}\right)_1$$

Массив точек координатных линий в определенный кадр объекта $Persuer\ 2$

$XX_3 := L_{B.3}\left(ARC_3^{\langle 0 \rangle}, Target2_{FRAME}\right)_0$

$YY_3 := L_{B.3}\left(ARC_3^{\langle 0 \rangle}, Target2_{FRAME}\right)_1$

Массив точек координатных линий в определенный кадр объекта $Persuer\ 3$

Подготовка к запуску основных расчетных операторов, для всех преследователей установлено максимальное количество циклов.

$NN_1 := 100 \quad NN_2 := 100 \quad NN_3 := 100$

Функция XY_1 выполняет расчет траектории преследователя $Persuer\ 1$.

$XY_1 :=$ | $p_0 \leftarrow Persuer1_{.start}$ — Инициализация расчетного массива стартовым значением преследователя $Persuer\ 1$

| $x \leftarrow (p_0)_0$
| $y \leftarrow (p_0)_1$ — Покоординатное присвоение буферным переменным стартового значения

| $l_x \leftarrow L_{B.1}\left(ARC_1^{\langle 0 \rangle}, Target1_0\right)_0$
| $l_y \leftarrow L_{B.1}\left(ARC_1^{\langle 0 \rangle}, Target1_0\right)_1$ — Покоординатное присвоение буферным массивам массивов базовой линии преследователя

| $d \leftarrow 0$ — Начальное значение длины дуги

| $k \leftarrow 0$ — Присвоение счетчику циклов значения 1

| for $i \in 1..NN_1$ — Организация вычислительного цикла

| | if $\left|p_{i-1} - Target1_{i-1}\right| \geq V_{1.Persuer} \cdot \Delta T$ — Если расстояние между преследователем и

целью больше расстояния одного шага, то выполнять следующие операторы

$d_b \leftarrow d$ — Присвоить первому значения диапазона текущего значения длины дуги

$d \leftarrow root\left(\left|L_{B.1}\left(d, Target_{1_i}\right) - p_{i-1}\right| - V_{1.Persuer} \cdot \Delta T, d, d_b, S_1\right)$ — Нахождение параметра длины дуги, соответствующему точке пересечения окружности с центром в точке p_{i-1} и текущей линией. Отсекание ненужного корня происходит выбором диапазона $[d_b; s_1]$

$p_i \leftarrow L_{B.1}\left(d, Target_{1_i}\right)$ — Определение следующего шага траектории

$x \leftarrow stack\left[x, \left(p_i\right)_0\right]$
$y \leftarrow stack\left[y, \left(p_i\right)_1\right]$ — Покоординатное заполнение массива траектории преследователя $Persuer\ 1$

$l_x \leftarrow stack\left(l_x, L_{B.1}\left(ARC_1^{\langle 0 \rangle}, Target_{1_i}\right)_0\right)$
$l_y \leftarrow stack\left(l_y, L_{B.1}\left(ARC_1^{\langle 0 \rangle}, Target_{1_i}\right)_1\right)$ — Покоординатное упорядоченное заполнение

двухпараметрического множества точек сети прогнозируемых траекторий

| | | $k \leftarrow i$ | Присвоение текущего значения счетчика циклов

| | | break otherwise | Если условие не выполнятся, то расчетный цикл прервать

$(x\ y\ l_x\ l_y\ k)$ Формирование матрицы выходных параметров

Выходными параметрами функции XY_1 служат координатные массивы траектории преследователя $Persuer\ 1$, покоординатные массивы множества прогнозируемых траекторий, общее количество расчетных циклов.

Функция XY_2 выполняет расчет траектории преследователя $Persuer\ 2$.

$XY_2 :=$ $p_0 \leftarrow Persuer_{2.start}$ Инициализация расчетного массива стартовым значением преследователя $Persuer\ 2$

$x \leftarrow (p_0)_0$
$y \leftarrow (p_0)_1$ Покоординатное присвоение буферным переменным стартового значения

$l_x \leftarrow L_{B.2}(ARC_2^{\langle 0 \rangle}, Target1_0)_0$
$l_y \leftarrow L_{B.2}(ARC_2^{\langle 0 \rangle}, Target1_0)_1$ Покоординатное присвоение буферным массивам массивов

базовой линии преследователя

$d \leftarrow 0$ Начальное значение длины дуги

$k \leftarrow 0$ Присвоение счетчику циклов значения 1

for $i \in 1..NN_2$ Организация вычислительного цикла

if $\left|p_{i-1} - Target1_{i-1}\right| \geq V_{2.Persuer} \cdot \Delta T$ Если расстояние между преследователем и целью больше расстояния одного шага, то выполнять следующие операторы

$d_b \leftarrow d$ Присвоить первому значения диапазона текущего значения длины дуги

$d \leftarrow root\left(\left|L_{B.2}\left(d, Target1_i\right) - p_{i-1}\right| - V_{2.Persuer} \cdot \Delta T, d, d_b, S_2\right)$ Нахождение параметра длины дуги, соответствующему точке пересечения окружности с центром в точке p_{i-1} и текущей линией. Отсекание ненужного корня происходит выбором диапазона $[d_b; s_2]$

$p_i \leftarrow L_{B.2}(d, Target1_i)$	Определение следующего шага траектории
$x \leftarrow stack[x, (p_i)_0]$ $y \leftarrow stack[y, (p_i)_1]$	Покоординатное заполнение массива траектории преследователя $Persuer\ 2$
$1_x \leftarrow stack(1_x, L_{B.2}(ARC_2^{\langle 0 \rangle}, Target1_i)_0)$ $1_y \leftarrow stack(1_y, L_{B.2}(ARC_2^{\langle 0 \rangle}, Target1_i)_1)$	Покоординатное упорядоченное заполнение двухпараметрического множества точек сети прогнозируемых траекторий
$k \leftarrow i$	Присвоение текущего значения счетчика циклов
break otherwise	Если условие не выполнятся, то расчетный цикл прервать
$(x\ y\ 1_x\ 1_y\ k)$	Формирование матрицы выходных параметров

Выходными параметрами функции XY_2 служат координатные массивы траектории преследователя $Persuer\ 2$, покоординатные массивы множества прогнозируемых траекторий, общее количество расчетных циклов.

Функция XY_3 выполняет расчет траектории преследователя $Persuer\ 3$.

$XY_3 :=$ $p_0 \leftarrow Persuer_{3.start}$ Инициализация расчетного массива стартовым значением преследователя $Persuer\ 3$

$x \leftarrow (p_0)_0$
$y \leftarrow (p_0)_1$ Покоординатное присвоение буферным переменным стартового значения

$l_x \leftarrow L_{B.3}(ARC_3^{\langle 0 \rangle}, Target2_0)_0$
$l_y \leftarrow L_{B.3}(ARC_3^{\langle 0 \rangle}, Target2_0)_1$ Покоординатное присвоение буферным массивам массивов базовой линии преследователя

$d \leftarrow 0$ Начальное значение длины дуги

$k \leftarrow 0$ Присвоение счетчику циклов значения 1

for $i \in 1..NN_3$ Организация вычислительного цикла

if $|p_{i-1} - Target2_{i-1}| \geq V_{3.Persuer} \cdot \Delta T$ Если расстояние между преследователем и целью больше расстояния одного шага, то выполнять следующие операторы

$d_b \leftarrow d$ Присвоить первому значения диапазона

текущего значения длины дуги

$d \leftarrow \text{root}\left(\left|L_{B.3}\left(d, \text{Target}_{2_i}\right) - p_{i-1}\right| - V_{3.Persuer} \cdot \Delta T, d, d_b, S_3\right)$

Нахождение параметра длины дуги, соответствующему точке пересечения окружности с центром в точке p_{i-1} и текущей линией. Отсекание ненужного корня происходит выбором диапазона $[d_b; s_3]$

$p_i \leftarrow L_{B.3}\left(d, \text{Target}_{2_i}\right)$

Определение следующего шага траектории

$x \leftarrow \text{stack}\left[x, \left(p_i\right)_0\right]$

$y \leftarrow \text{stack}\left[y, \left(p_i\right)_1\right]$

Покоординатное заполнение массива траектории преследователя $Persuer\ 3$

$l_x \leftarrow \text{stack}\left(l_x, L_{B.3}\left(ARC_3^{\langle 0 \rangle}, \text{Target}_{2_i}\right)_0\right)$

$l_y \leftarrow \text{stack}\left(l_y, L_{B.3}\left(ARC_3^{\langle 0 \rangle}, \text{Target}_{2_i}\right)_1\right)$

Покоординатное упорядоченное заполнение двухпараметрического множества точек сети прогнозируемых траекторий

		$k \leftarrow i$	Присвоение текущего значения счетчика циклов
		break otherwise	Если условие не выполнятся, то расчетный цикл прервать
		$(x\ y\ l_x\ l_y\ k)$	Формирование матрицы выходных параметров

Выходными параметрами функции XY_3 служат координатные массивы траектории преследователя $Persuer\ 3$, покоординатные массивы множества прогнозируемых траекторий, общее количество расчетных циклов.

Далее, выводим общее число расчетных циклов после выполнения каждой функции.

$$XY1_{0,4} = 83 \quad XY2_{0,4} = 83 \quad XY3_{0,4} = 83$$

Как видим, значения числа расчетных циклов для всех функций равно одному числу, Это означает, цели достигнуты одновременно, что мы и хотели показать.

Формирование объединенного массива координат целей, соответствующих переменной $FRAME$, то есть кадру.

$$X_{Target} := \text{stack}\left(X_{1.Target_{FRAME}}, X_{2.Target_{FRAME}}\right) \quad Y_{Target} := \text{stack}\left(Y_{1.Target_{FRAME}}, Y_{2.Target_{FRAME}}\right)$$

В системе MathCAD 15 есть ограничение на вывод количества переменных при визуализации.

Формирование объединенного массива координат преследователей, соответствующих переменной $FRAME$.

$$Y_{Persuer} := \text{stack}\left[\left(XY1_{0,1}\right)_{FRAME}, \left(XY2_{0,1}\right)_{FRAME}, \left(XY3_{0,1}\right)_{FRAME}\right]$$
$$X_{Persuer} := \text{stack}\left[\left(XY1_{0,0}\right)_{FRAME}, \left(XY2_{0,0}\right)_{FRAME}, \left(XY3_{0,0}\right)_{FRAME}\right]$$

Формирование объединенного массива координат сети прогнозируемых траекторий преследователей.

$$X_{prediction} := stack\left[\left(XY1_{0,2}\right),\left(XY2_{0,2}\right),\left(XY3_{0,2}\right)\right] Y_{prediction} := stack\left[\left(XY1_{0,3}\right),\left(XY2_{0,3}\right),\left(XY3_{0,3}\right)\right]$$

Формирование траекторий движения преследователей.

$$Y_{trajectory} := stack\left[\left(XY1_{0,1}\right),\left(XY2_{0,1}\right),\left(XY3_{0,1}\right)\right] X_{trajectory} := stack\left[\left(XY1_{0,0}\right),\left(XY2_{0,0}\right),\left(XY3_{0,0}\right)\right]$$

На рисунке 3.34 показан результат работы тестовой программы одновременного достижения группой преследователей группы целей с сетью прогнозируемых траекторий. Рисунок 3.34 дополнен ссылкой на анимированное изображение [56].

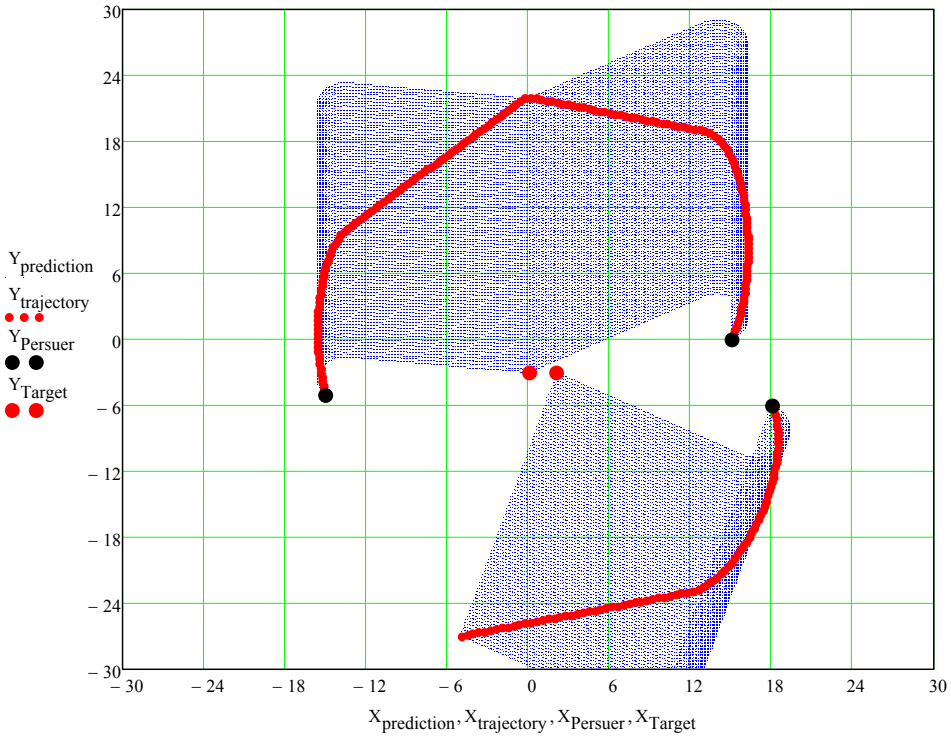


Рисунок 3.34. Достижение целей с сетью линий прогнозирования

На рисунке 3.35 показана работа группы преследователей по группе целей уже без сети прогнозируемых траекторий. Рисунок 3.35 дополнен ссылкой на анимированное изображение [57].

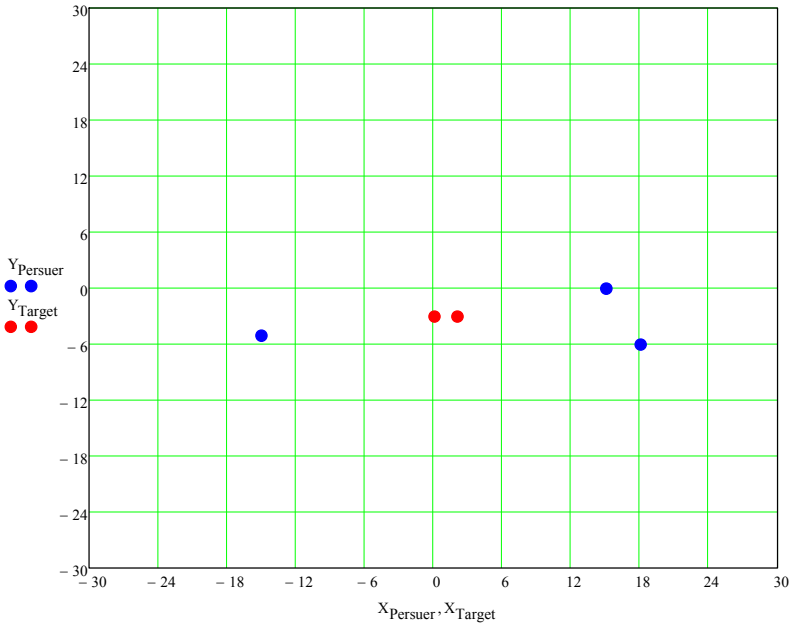


Рисунок 3.35. Достижение целей без визуализации сети прогнозируемых траекторий

ЗАКЛЮЧЕНИЕ

Современное состояние вычислительной техники и темпы развития робототехники, беспилотных летательных аппаратов уже требуют разработки моделей и алгоритмов, в которых участниками принимаются решения.

В этой связи очень актуальными являются задачи преследования, уклонения, группового преследования, уклонения от группы объектов.

В данном исследовании получены следующие результаты:

1. Произведено моделирование траекторий движения преследующего и преследуемого объектов по поверхности, заданной точечным базисом
2. Произведено моделирование траектории движения преследующего объекта по геодезической линии.
3. Произведено моделирование траектории преследующего объекта с учетом опережения
4. Разработаны модели поведения преследующего и преследуемого объектов на пересеченной местности. Разработаны методы анализа фазовых координат оппонента по задаче преследования. Разработана методика принятия решения участником задачи преследования
5. Разработаны модели поведения объектов, участников задачи преследования, когда они находятся на разных поверхностях. Предложен метод построения поверхности, когда объект преследования находится сверху над объектом преследования
6. Разработан алгоритм автоматизированного построения траекторий, предполагающих обход движущихся препятствий
7. Разработан алгоритм автоматизированного построения траекторий с заданными дифференциальными ограничениями

Проведенные исследования показали, что геометрическое моделирование, проводимое в процессе разработки робототехнических комплексов, оснащенных системами искусственного интеллекта для принятия

решений, является необходимой частью для прогнозирования результатов квазидискретных задач преследования.

Компьютерное моделирование дает широкие возможности для прогнозирования результатов разнообразных игр преследования, как непрерывных, так и квазидискретных.

Так же и визуализация процессов преследования, убегания, уклонения и т.д. оказывает огромную помощь в работе разработчика.

Теоретические результаты исследования, методы и алгоритмы геометрического моделирования целесообразно использовать при разработке робототехнических комплексов, оснащенных системами искусственного интеллекта, которые в своей работе выполняют различные задачи преследования.

ЛИТЕРАТУРА

1. Дубанов А.А. Геометрическое моделирование задач преследования в среде Mathcad: монография. М.: РИОР, 2020.
2. Дубанов А.А. Моделирование траекторий от преследователя до цели с ограничениями на кривизну и с заданными краевыми условиями. Свидетельство о регистрации программы для ЭВМ № 2020614336, от 20.03.2020.
3. Дубанов А.А., Аюшеев Т.В. Геометрическая модель преследования группой одиночной цели // В сборнике: проблемы машиноведения. Материалы IV международной научно-технической конференции / Научный редактор П.Д. Балакин. 2020. С. 432–437.
4. Дубанов А.А., Аюшеев Т.В. Алгоритмы следования по предполагаемым траекториям от преследователя до динамической цели // В сборнике: проблемы машиноведения. Материалы IV международной научно-технической конференции / Научный редактор П.Д. Балакин. 2020. С. 437–442.
5. Нефедов Ю.Ю, Дубанов А.А. О построении траектории обхода препятствия // В сборнике: проблемы машиноведения. Материалы IV международной научно-технической конференции / Научный редактор П.Д. Балакин. 2020. С. 443–446.
6. Дубанов А.А. Модель группового преследования одиночной цели на основе следования ранее прогнозируемым траекториям // Информационные технологии. 2020. Т. 26. № 6. С. 334–341.
7. Визуализация окружностей Аполлония при геометрическом моделировании метода параллельного сближения на плоскости / А.А. Дубанов, А.В. Урбаханов, Н.Б. Цыренжапов, А.Э. Севээн // Научно-технический вестник Поволжья. 2020. № 6. С. 105–109.
8. Trajectory modelling when bypassing obstacles. A.A. Dubanov, T.V. Ausheev. Journal of physics: conference series. XIII international scientific

and technical conference “Applied Mechanics and Systems Dynamics”. 2020. P. 012057.

9. Dubanov A.A., Ausheev T.V. Geometric model of persecution by a group of one goal. Journal of physics: conference series. IV international scientific and technical conference “Mechanical Science and Technology Update”, MSTU 2020. 2020. P. 012036.
10. Dubanov A.A., Ausheev T.V. A geometric model for following the intended trajectories from the pursuer to the target. Journal of physics: conference series. IV international scientific and technical conference “Mechanical Science and Technology Update”, MSTU 2020. 2020. P. 012035.
11. Нефедова В.А., Дубанов А.А. Визуализация окружностей Аполлония при геометрическом моделировании метода параллельного сближения на плоскости // Научное обозрение. Технические науки. 2020. № 4. С. 70–75.
12. Геометрическая квазидискретная модель группового преследования одиночной цели/ Дубанов А.А., Аюшеев Т.В., Севеэн А.О., Вестник Южно-Уральского Государственного Университета. Серия: Строительство и Архитектура. 2020. Т. 20. № 4. С. 65–72.
13. Дубанов А.А., Аюшеев Т.В. Кинематическая модель задачи преследования на плоскости методом погони // Динамика систем, механизмов и машин. 2020. Т. 8. № 1. С. 154–160.
14. Дубанов А.А., Аюшеев Т.В. Кинематическая модель метода параллельного сближения // Динамика систем, механизмов и машин. 2020. Т. 8. № 1. С. 160–165.
15. Геометрическая модель метода параллельного сближения / А.А. Дубанов, Б.В. Заятуев, А.В. Бадеев, А.Э. Севеэн // Научно-технический вестник Поволжья. 2020. № 12. С. 187–189.

16. Геометрическая модель задачи преследования на плоскости методом погони / А.А. Дубанов, Б.В. Заятуев, А.В. Бадеев, А.Э. Свээн // Научно-технический вестник Поволжья. 2020. № 12. С. 190–192.
17. Дубанов А.А., Севээн А.Э. Кинематическая модель метода параллельного сближения // Свидетельство о регистрации программы для ЭВМ № 2020664886 от 20.11.2020.
18. Дубанов А.А., Севээн А.Э. Моделирование траектории преследователя на поверхности методом параллельного сближения // Свидетельство о регистрации программы для ЭВМ № 2020664893 от 20.11.2020.
19. Dubanov A.A. Model of group pursuit of a single target based on following previously predicted trajectories. Advances in intelligent systems and computing, 2020. Vol. 1295. Pp. 36–49.
20. Dubanov A.A., Seveen A.E., Tsyrenzhapov N.B. Geometric modeling of the parallel approach method in some transport problems. IOP conference series: materials science and engineering. 8 T. “VIII International Scientific Conference Transport of Siberia 2020”. 2020. P. 012088.
21. Dubanov A.A., Nefedova V.A., Tashkane A.S. Numerical and analytical building surface crossing lines in some transport tasks. IOP conference series: materials science and engineering. 8. T. “VIII International Scientific Conference Transport of Siberia 2020”. 2020. P. 012018.
22. Dubanov A.A. Trajectory modeling in a pursuit problem with curvature restrictions. Advances in intelligent systems and computing, 2020. Vol. 1224. Pp. 226–232.
23. Дубанов А.А. Построение моделей движения объектов в задаче преследования. решение в системе вычислительной математики Mathcad. Cloud of Science. 2019. Т. 6. № 1. С. 48–62.
24. Дубанов А.А. Задача преследования объекта с поверхности, расположенной над поверхностью преследуемого // Вестник Южно-

Уральского государственного университета. Серия: Строительство и Архитектура. 2019. Т. 19. № 2. С. 67–72.

25. Построение моделей движения объектов в задаче преследования. решение в системе вычислительной математики “MathCAD” / И.Т. Бубеев, А.А. Дубанов, Т.В. Аюшеев, П.В. Мотошкин // Программные системы и вычислительные методы. 2019. № 1. С. 1–11.
26. Дубанов А.А. Задача преследования объектов, передвигающихся по разным поверхностям // Вестник кибернетики. 2019. № 1(33). С. 100–105.
27. Dubanov A.A. Modeling the behavior of objects in the pursuit problem. Advances in intelligent systems and computing 2019. Vol. 984. Pp. 259–274.
28. Дубанов А.А. Модели поведения объектов в задаче преследования на поверхности // Свидетельство о регистрации программы для ЭВМ № 2019618234 от 08.07.2019.
29. Дубанов А.А., Аюшеев Т.В. Моделирование траекторий при обходе препятствий // Динамика систем, механизмов и машин. 2019. Т. 7. № 4. С. 110–117.
30. Dubanov A.A., Semenov D.A., Ausheev T.V. The task of pursuing objects moving on different surfaces. Journal of physics: Conference series. 2019. P. 072004.
31. Нефедова В.А., Дубанов А.А. Моделирование траектории с ограничениями на кривизну // Научное обозрение. Технические науки. 2019. № 6. С. 38–43.
32. Дубанов А.А., Аюшеев Т.В., Урбаханов А.В. Конструирование траекторий с заданными ограничениями по кривизне // Прикладная математика и фундаментальная информатика. 2019. Т. 6. № 2. С. 12–21.
33. Дубанов А.А., Аюшеев Т.В., Урбаханов А.В. Моделирование траекторий при обходе препятствий // Прикладная математика и фундаментальная информатика. 2019. Т. 6. № 2. С. 22–33.

34. Дубанов А.А., Нефедов Ю.Ю. Конструирование траекторий обхода нескольких препятствий // Прикладная математика и фундаментальная информатика. 2019. Т. 6. № 4. С. 18–22.
35. Дубанов А.А. Визуализация окружностей Аполлония при геометрическом моделировании метода параллельного сближения на плоскости // Прикладная математика и фундаментальная информатика. 2019. Т. 6. № 4. С. 23–31.
36. Дубанов А.А. Задача преследования. решение в системе вычислительной математики MathCAD // Информационные технологии. 2018. Т. 24. № 4. С. 251–255.
37. Дубанов А.А., Билдушкина М.Н. Построение геодезических линий в системе компьютерной математики MathCAD // T-Comm: Телекоммуникации и транспорт. 2018. Т. 12. № 7. С. 37–41.
38. Дубанов А.А., Билдушкина М.Н. Построение геодезических линий применительно к задаче преследования в системе компьютерной математики MathCAD // Моделирование, оптимизация и информационные технологии. 2018. Т. 6. № 3(22). С. 121–131.
39. Дубанов А.А., Билдушкина М.Н. Построение геодезических линий в системе компьютерной математики MathCAD // Cloud of Science. 2018. Т. 5. № 4. С. 599–607.
40. Дубанов А.А., Билдушкина М.Н. Построение геодезических линий в системе компьютерной математики MathCAD // Нейрокомпьютеры: разработка, применение. 2018. № 9. С. 61–66.
41. Дубанов А.А., Эрдынеева Л.И. Задача преследования в системе вычислительной математики MathCAD // В сборнике: Современные проблемы телекоммуникаций. материалы конференции. 2016. С. 129–133.

42. Дубанов А.А., Эрдынеева Л.И. Задача преследования в системе вычислительной математики MathCAD // Международный журнал прикладных и фундаментальных исследований. 2016. № 9-1. С. 7–11.
43. Анимированное изображение, Перехват цели. URL: https://www.youtube.com/watch?v=rsMGA1ICo7M (дата обращения: 20.04.2021).
44. Анимированное изображение, Моделирование убегания цели от преследователя. URL: https://www.youtube.com/watch?v=hGieKXNiuz8 (дата обращения: 20.04.2021).
45. Анимированное изображение, Кинематическая модель параллельного сближения. URL: https://www.youtube.com/watch?v=qNXdykK21Z8 (дата обращения: 20.04.2021).
46. Анимированное изображение, Проекция прямой линии на поверхность. URL: https://www.youtube.com/watch?v=06qgINE4j8U (дата обращения: 20.04.2021).
47. Анимированное изображение, Сфера на поверхности. URL: https://www.youtube.com/watch?v=xszwIyTHUec (дата обращения: 20.04.2021).
48. Анимированное изображение, Визуализация окружностей Аполлония. URL: https://youtu.be/rsMGA1ICo7M (дата обращения: 20.04.2021).
49. Анимированное изображение, Визуализация задачи преследования методом параллельного сближения на плоскости. URL: https://youtu.be/qNXdykK21Z8 (дата обращения: 20.04.2021).
50. Анимированное изображение, Визуализация метода погони, когда скорость преследователя всегда направлена на цель. URL: https://youtu.be/PAu9Qg1dySM (дата обращения: 20.04.2021).
51. Анимированное изображение, Итерационный процесс задачи преследования методом погони. URL: https://youtu.be/UQ5bVKjVqZ4 (дата обращения: 20.04.2021).

52. Анимированное изображение, Корректировка направления движения преследователя. URL: https://youtu.be/XubnxJfk-Lc (дата обращения: 20.04.2021).
53. Анимированное изображение, Результат моделирования группового преследования одиночной цели. URL: https://www.youtube.com/watch?v=aC4PuXTgVS0&feature=youtu.be (дата обращения: 20.04.2021).
54. Анимированное изображение, Групповое преследование с жесткими связями. URL: https://youtu.be/sLy7Jvppf4A (дата обращения: 20.04.2021).
55. Анимированное изображение, Преследование одной цели двумя преследователями. URL: https://www.youtube.com/watch?v=7VNHNwCbWrg (дата обращения: 20.04.2021).
56. Анимированное изображение, Преследование двух целей группой из трех преследователей. URL: https://youtu.be/NNJDJOJT34I (дата обращения: 20.04.2021).
57. Анимированное изображение, Преследование двух целей группой из трех преследователей без вспомогательных линий. URL: https://youtu.be/tdbgoNoby3A (дата обращения: 20.04.2021).
58. Вагин Д.А., Петров Н.Н. Задача преследования жестко скоординированных убегающих // Известия РАН. Теория и системы управления. 2001. № 5. С. 75–79.
59. Вагин Д.А., Петров Н.Н. Об одной задаче группового преследования с фазовыми ограничениями // Прикладная математика и механика. 2002. Т. 66. Вып. 2. С. 234–241.
60. Банников А.С. Некоторые нестационарные задачи группового преследования, Известия Института математики и информатики УдГУ. 2013. Вып. 1(41). С. 3–46.

61. Банников А.С. Нестационарная задача группового преследования // Труды Математического центра имени Н.И. Лобачевского. Казань: Изд-во Казанского математического общества, 2006. Т. 34. С. 26–28.
62. Банников А.С. Нестационарная задача группового преследования // Проблемы теоретической и прикладной математики: тр. 39-й Всерос. молодеж. конф., 28 янв. — 1 фев. 2008 г. Екатеринбург: УрО РАН, 2008. С. 221–223.
63. Бардадым Т.А. Задача преследования с простым движением и разнотипными ограничениями на управления // Кибернетика. 1982. № 2. С. 80–84.
64. Благодатских В.И. Введение в оптимальное управление (линейная теория). М.: Высшая школа, 2001. 240 с.
65. Благодатских А.И., Петров Н.Н. Конфликтное взаимодействие групп управляемых объектов. Ижевск: Изд-во Удмурт. ун-та, 2009. 266 с.
66. Губарев Е.В. Убегание от группы преследователей // Автоматика. 1992. № 5. С. 66–70.
67. Изместьев И.В., Ухоботов В.И. Задача преследования маломаневренных объектов с терминальным множеством в форме кольца. Материалы международной конференции «Геометрические методы в теории управления и математической физике: дифференциальные уравнения, интегрируемость, качественная теория» Рязань, 15–18 сентября 2016 г., Итоги науки и техн. Сер. Соврем. мат. и ее прил. Темат. обз., 148, ВИНИТИ РАН, М., 2018. С. 25–31.
68. Ковшов А.М. Параллельные стратегии в играх преследования на сфере: автореф. дис. канд. физ.-матем. наук. СПб., 1996. 12 с.
69. Константинов Р.В. О квазилинейной дифференциальной игре с простой динамикой при наличии фазового ограничения // Математические заметки. 2001. Т. 69. Вып. 4. С. 581–590.

70. Котов И.И. Новый метод построения поверхностей, удовлетворяющих некоторым наперед заданным требованиям // Вопросы теории, приложений и методики преподавания начертательной геометрии (труды Рижской научно-методической конференции, июнь 1957). Рига: Рижский институт инженеров гражданского воздушного флота, 1960. С. 143–161.
71. Красовский Н.Н. Игровые задачи о встрече движений. М.: Наука, 1970.
72. Красовский Н.Н., Субботин А.И. Позиционные дифференциальные игры. М.: Наука, 1974. 456 с.
73. Кумков С.И., Пацко В.С. Задача преследования с неполной информацией: Препринт. Екатеринбург; ИММ УрО РАН, 1993. 64 с.
74. Макаров И.М., Лохин В.М. Манько С.В. Искусственный интеллект и интеллектуальные системы управления. М.: Наука, 2006. 333 с.
75. Панкратова Я.Б. Решение кооперативной дифференциальной игры группового преследования // Дискретный анализ и исследование операций. 2010. Т. 17. № 2. С. 57–78.
76. Петросян Л.А., Рихсиев Б.Б. Преследование на плоскости. М.: Наука, 1961.
77. Петросян Л.А., Зенкевич Н.А., Шевкопляс Е.В. Теория Игр. Изд-во «БХВ-Петербург», 2012. 424 с.
78. Петросян Л.А., Зенкевич Н.А., Семина Е.А. Теория игр. М., 1998. 300 с.
79. Петросян Л.А., Захаров В.В. Математические модели в экологии. Изд-во СПбГУ, 1997. 254 с.
80. Петросян Л.А., Рихсиев Б.Б. Преследование на плоскости. М.: Наука, 1991. 94 с.
81. Петросян Л.А., Томский Г.В. Геометрия простого преследования. М.: Наука, 1983. 143 с.
82. Петросян Л.А., Зубов В.И. Математические методы в планировании. Изд-во ЛГУ, 1982. 96 с.

83. Петросян Л.А. Дифференциальные игры преследования. Изд-во ЛГУ, 1977. 222 с.
84. Понтрягии Л.С. К теории дифференциальных игр // Успехи математических наук. Т. XXI. Вып. 4, 1966.
85. Понтрягин Л.С., Болтянский В.Г., Гамкрелидзе Р.В., Мищенко Е.Ф. Математическая теория оптимальных процессов. М.: Наука, 1976.
86. Понтрягин Л.С. Избранные научные труды. Т. 2. М.: Наука, 1988. 576 с.
87. Понтрягин Л.С. Линейная дифференциальная игра убегания // Труды Математического института АН СССР. 1971. Т. 112. С. 30–63.
88. Понтрягин Л.С., Мищенко Е.Ф. Задача об убегании одного управляемого объекта от другого // ДАН СССР. 1969. Т. 189. № 4. С. 721–723.
89. Понтрягин Л.С., Мищенко Е.Ф. Задача об уклонении от встречи в линейных дифференциальных играх // Дифференциальные уравнения. 1971. Т. 7. № 3. С. 436–445.
90. Пшеничный Б.Н., Чикрий А.А. Задача об уклонении от встречи в дифференциальных играх // Журнал вычислительной математики и математической физики. 1974. Т. 14. № 6. С. 416–427.
91. Пшеничный Б.Н. О задаче убегания // Кибернетика. 1975. № 4. С. 120–127.
92. Пшеничный Б.Н. Простое преследование несколькими объектами // Кибернетика. 1976. № 3. С. 145–146.
93. Пшеничный Б.Н., Чикрий А.А. Дифференциальная игра уклонения // Известия АН СССР. Техническая кибернетика. 1977. № 1. С. 3–9.
94. Пшеничный Б.Н., Остапенко В.В. Дифференциальные игры. Киев: Наук. думка, 1992. 260 с.
95. Савелов А.А. Плоские кривые. М.: Книжный дом «Либ-роком», 2009. 296 с.

96. Саматов Б.Т. Задача преследования-убегания при интегрально-геометрических ограничениях на управления преследователя // Автомат. и телемех. 2013. № 7. С. 17–28.
97. Хачумов М.В. Решение задачи следования за целью автономным летательным аппаратом // Искусственный интеллект и принятие решений. 2015. № 2. С. 45–52.
98. Хачумов М.В. Задачи группового преследования цели в условиях возмущений // Искусственный интеллект и принятие решений. 2016. № 2. С. 46–54.
99. Isaacs R. Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization. New York: John Wiley & Sons, 1965.
100. Torrence D. Parsons. Pursuit-evasion in a graph. Theory and Applications of Graphs. Springer-Verlag, 1976. С. 426–441.
101. Richard Borie, Craig Tovey, Sven Koenig. Algorithms and Complexity Results for Pursuit-Evasion Problems. 2009.
102. Ellis J., Sudborough I., Turner J. The vertex separation and search number of a graph. Information and Computation. 1994. Vol. 113 (1). С. 50–79.
103. Fomin F.V., Thilikos D. An annotated bibliography on guaranteed graph searching. Theoretical Computer Science. 2008. Vol. 399 (3). С. 236–245.
104. Kirousis M. Papadimitriou C. Searching and pebbling. Theoretical Computer Science. 1986. Vol. 42 (2). Pp. 205–218.
105. Nowakowski R., Winkler P. Vertex-to-vertex pursuit in a graph. Discrete Mathematics. 1983. Vol. 43 (2–3). Pp. 235–239.
106. Petrosjan L.A. Differential Games of Pursuit. World Scientific Pub Co Inc., 1993. Vol. 2. (Series on Optimization).

107. Petrosyan L.A. Yeung D.W.K. Subgame-consistent Economic Optimization. Springer, 2012. 396 p.
108. Petrosjan L.A., Zenkevich N.A. Game Theory. World Scientific Publisher, 1996. 350 p.
109. Petrosjan L.A. Differential Games of Pursuit. World Scientific Publisher, 1993. 326 p.
110. Yeung D.W.K., Petrosyan L.A. Cooperative Stochastic Differential Games. Springer, 2006. 242 p.
111. Seymour P., Thomas R. Graph searching, and a min-max theorem for tree-width. Journal of Combinatorial Theory, Series B. 1993. Vol. 58 (1). Pp. 22–33.
112. Rene Vidal, Omid Shakernia, H. Jin Kim, David Hyunchul Shim, Shankar Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. IEEE Transactions on Robotics and Automation. 2002. Vol. 18 (5).
113. Marcos A.M. Vieira, Ramesh Govindan, Gaurav S. Sukhatme. Scalable and Practical Pursuit-Evasion with Networked Robots. Journal of Intelligent Service Robotics Special Issue on Networked Robots. 2009.
114. Chern F. Chung, Tomonari Furukawa. A Reachability-Based Strategy for the Time-Optimal Control of Autonomous Pursuers. Engineering Optimization. 2008. Vol. 40 (1).
115. Joao P. Hespanha, Hyoun Jin Kim, Shankar Sastry. Multiple-agent probabilistic pursuit-evasion games. 1999.
116. Ашкенази В.О. Применение теории игр в военном деле// Сборник переводов с английского под редакцией Ашкенази В.О.// Издательство «Советское Радио», М., 1961.
117. Кузьмина Л.И., Осипов Ю.В. Расчет длины траектории для задачи преследования // Вестник МГСУ. Рецензируемый научно-технический

журнал по строительству и архитектуре. НИУ МГСУ. 2013. № 12. С. 20–26.

118. Маматов М.Ш. Игровая задача преследования и убегания с управлением, заданным разностными уравнениями второго порядка // Известия Института математики и информатики Удмуртского государственного университета. 2006. С. 95–96.

119. Пацко В.С., Турова В.Л. Игра «шофер-убийца» и ее модификации // Вестник Удмуртского университета. Математика. 2008. Вып. 2.

120. Романников Д.О. Пример решения минимаксной задачи преследования с использованием нейронных сетей // Сборник научных трудов НГТУ. 2018. № (92). С. 108–116.

121. Келенджеридзе Д.Л. Об одной задаче оптимального преследования // Автоматика и телемеханика. 1962. Т. 23. Вып. 8. С. 1008–1013.

122. Breakwell J.V., Merz A.W. Toward a complete solution of the homicidal chaueur game. Proc. of the 1st Int. Conf. on the Theory and Application of Dierential Games, Amherst, Massachusetts, 1969. P. III-1III-5.

123. Lewin J. Decoy in pursuit-evasion games: PhD Thesis. Stanford University, 1973. 17.

124. Lewin J., Olsder G.J. Conic surveillance evasion // J. Opt. Theory Appl. 1979. Vol. 27, no, 1. Pp. 107–125.

125. Cardaliaguet P., Quincampoix M., Saint-Pierre P. Numerical methods for optimal control and dierential games. Ceremade CNRS URA 749. University of Paris Dauphine, 1995.

126. Cardaliaguet P., Quincampoix M., Saint-Pierre P. Set-valued numerical analysis for optimal control and dierential games. In M. Bardi, T.E.S. Raghavan and T. Parthasarathy (eds.), Stochastic and Dierential Games:

Theory and Numerical Methods, Annals of the Int. Soc. of Dynamic Games. Boston: Birkhauser. 1999. Vol. 4. Pp. 177–247.

127. Dubins L.E. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. Amer. J. Math. 1957. Vol. 79. Pp. 497–516.

128. Reeds J.A., Shepp L.A. Optimal paths for a car that goes both forwards and backwards. Pacic J. Math. 1990. Vol. 145, no. 2. Pp. 367–393.

129. Геометрическое моделирование в MathCAD. URL: http://dubanov.exponenta.ru.

130. Lewin J., Breakwell J.V. The surveillance-evasion game of degree. J. Opt. Theory Appl. 1975. Vol. 16, no. 34. Pp. 339–353.