# Описание программного модуля

Целью данной работы является разработка программного интерфейса (API - application programming interface), который упростит разработку программ моделирования систем радиоуправления с использованием языка программирования Python.

## Описание структуры базового класса и класса движения тела с нормальным ускорением

Программный интерфейс для моделирования систем радиоуправления представляет собой набор классов. Методы этих классов реализуют внутри себя вычисление основных параметров движения объекта. Реализован базовый класс *Object* в файле *universal\_model.py*, который вычисляет основные кинематические параметры объекта, такие как скорость, угол вектора скорости относительно горизонтальной оси. Также ведет слежение за историей перемещения и имеет некоторые дополнительные функции.

На основе базового класса формируется класс, для целей перемещающийся по следующему закону:

(1)

(2)

Непосредственно закон перемещения объекта *ObjectConstVelocity*, данного тела выглядит следующим образом:

(3)

(4)

Характер движения цели формируется конфигурационной структурой, передаваемой в конструктор класса. Рассмотрим пример такой структуры на рисунке 1. Поля *initialX* и *initialY* задают начальное положение тела или историй перемещения, если передать в список набор точек, а не одну. Поле *direction* задает угол направления вектора скорости тела. Соответственно поля *velocity* и *acceleration* передают модуль вектора скорости и ускорения, соответственно. А описание цели, которое можно вынести на график для различения траекторий вноситься в роле *description*.



Рисунок – Пример конфигураций движения тела

Проверим корректность траектории передвижения объекта следующим образом, будем поочерёдно формировать различные типы траекторий и сопоставлять ожидаемый характер движения и фактический. Типы траекторий указаны на рисунке 1:

1. Начальная координата x = 30 [м]; начальная координат y = 30[м]; направление вектора скорости вверх (90 градусов относительно оси x); скорость 1 [м/с]; ускорение 0 [м\с]2; ожидаемая траектория объекта – вертикальная прямая.
2. Начальная координата x = 30 [м]; начальная координат y = 30[м]; направление вектора скорости 0 градусов относительно оси x(прямо); скорость 1 [м/с]; ускорение 0 [м\с]2; ожидаемая траектория объекта – прямая вдоль оси x.
3. Начальная координата x = 30 [м]; начальная координат y = 30[м]; направление вектора скорости 45 градусов относительно оси x; скорость 1 [м/с]; ускорение 0 [м\с]2; ожидаемая траектория объекта – прямая направленная под углом 45 градусов относительно оси Х.
4. Начальная координата x = 30 [м]; начальная координат y = 30[м]; направление вектора скорости 180 градусов относительно оси x (назад); скорость 1 [м/с]; ускорение 0 [м\с]2; ожидаемая траектория объекта – движение назад (против оси х).

Дальше в примерах движения не будет приводиться структура конфигурации за примерами можно обратиться к программе с исходным кодом.

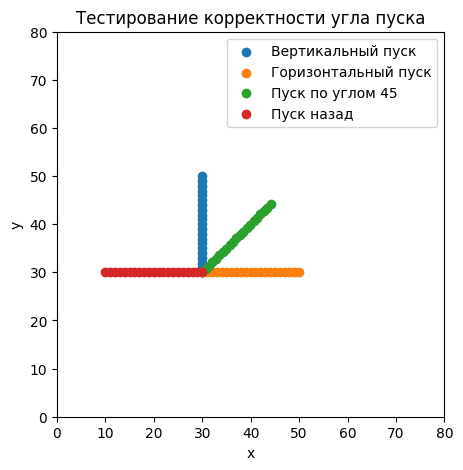


Рисунок – Тестовые прямолинейные траектории движения тела

Как можно видеть на рисунке 2 полученные траектории соответствуют ожидаемым. В данном тестовом примере было продемонстрированно корректность установки вектора скорости. Можно проверить, как влияет изменение скорости движения тел (поле *velocity*).

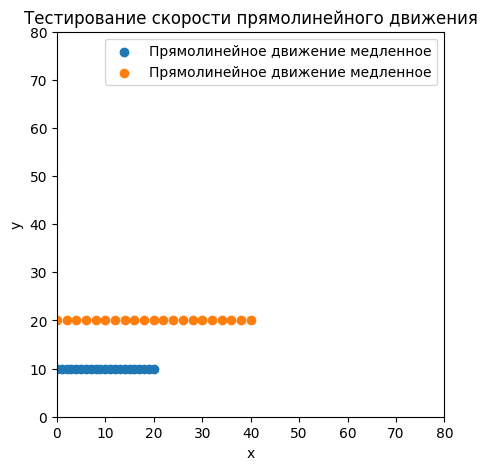


Рисунок – Тестовые траектории движения тела  
 при различных скоростях цели

А что будет если добавить нормальное ускорение (положительное значение в поле конфигурации *acceleration*)? Траектория тела примет вид окружности, что демонстрирует рисунок 3.

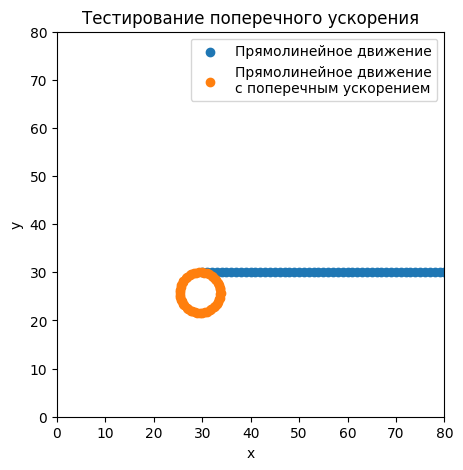


Рисунок – Тестовые прямолинейные траектории движения тела

Также можно наблюдать, как изменяется радиус окружности при изменении либо ускорения (рисунок 5), либо скорости (рисунок 6). В общем можно наблюдать верную тенденцию, при увеличении этих параметров радиус окружности увеличивается.



Рисунок – Тестовые траектории движения тела  
 при различных ускорения цели

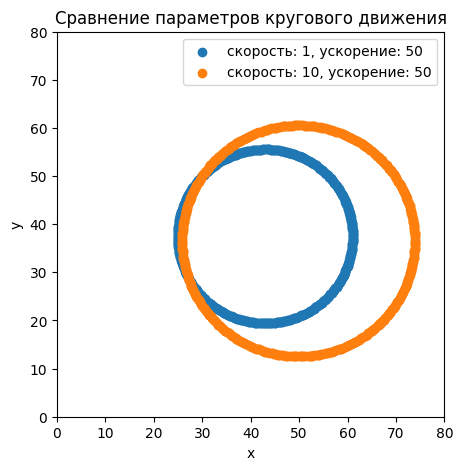


Рисунок – Тестовые траектории движения тела при различных  
 скоростях цели при наличии бокового ускорения

## Описание вспомогательного класса для линии визирования

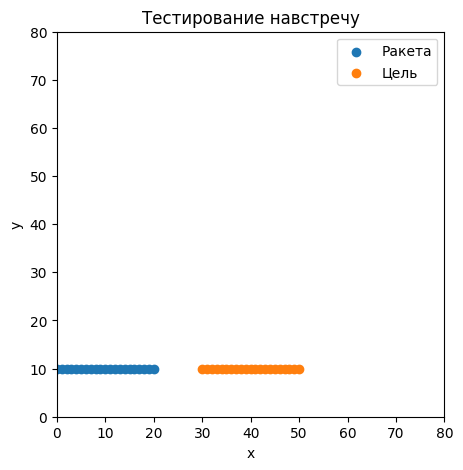
Так как в алгоритмах радиоуправления довольно часто применяется понятие линии визирования, то реализовать такую абстракцию имеет смысл.

Конструктор данного "соединяет" два объекта, которые реализованы наследованием от класса Object. С помощью приватных методов производиться измерение различных параметров линии визирования. Стоит отметить, что приватные методы класса предназначены только для внутреннего использования, к которым не должно быть доступа из экземпляра класса. При использовании данного экземпляра класса после его инициализации нужно вызывать метод getAllParams, в нем буду возвращаться все необходимые параметры такие как: приращение координат, дистанция между объектами, угол визирования, скорость сближения тел, скорость вращения линии визирования.

Будем формировать последовательно различные типы траекторий, и измерять параметры линии визирования в финальных точках. Будем рассматривать следующие траектории:

1. Движение тел навстречу
2. Движение тел параллельно разнонаправленными курсами
3. Движение под углом навстречу

Рассмотри первый случай. Тела двигаются навстречу друг другу с равными скоростями. Результаты приведены на рисунке 7.



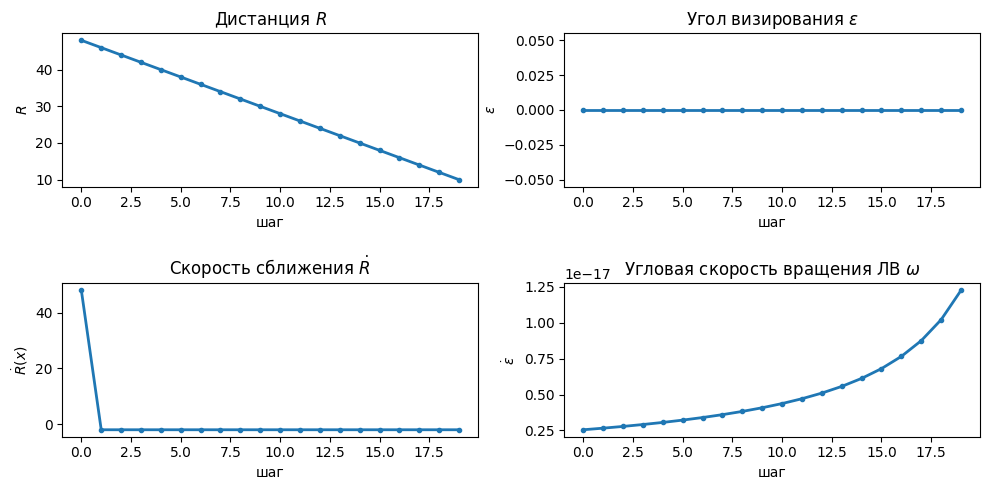


Рисунок – Результаты исследования линии  
 визирования навстречу

Можно обратить внимание на следующее. Угол визирования и скорость вращения линии визирования не меняются (форма графика может смутить, однако стоит обратить внимание на размерность оси абсцисс). Дистанция между объектами уменьшается так, как они сближаться. Скорость сближения остается постоянной. Отрицательный знак скорости сближения также объясняется математически. Это производная функции расстояния между объектами, которая уменьшается со "временем", следовательно, скорость сближения отрицательна. Хотя физически это не совсем верно. Таким образом, данный тестовый случай соответствует ожиданиям.

Проведем аналитический расчет случая для сравнения результатов.

(5)

(6)

Найдем расстояние между целями через 20 шагов с временным интервалом равным 1.

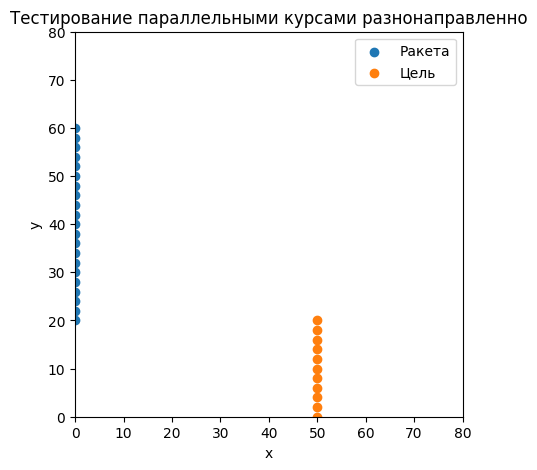
(7)

Нетрудно также найти и скорость сближения цели и ОУ.

(8)

Модель показала аналогичный результат.

Рассмотри другой пример, тела движутся в противоположных направлениях с равными скоростями в противоположных направлениях, на расстоянии 50 метров друг от друга. Результат моделирования приведен на рисунке 8.



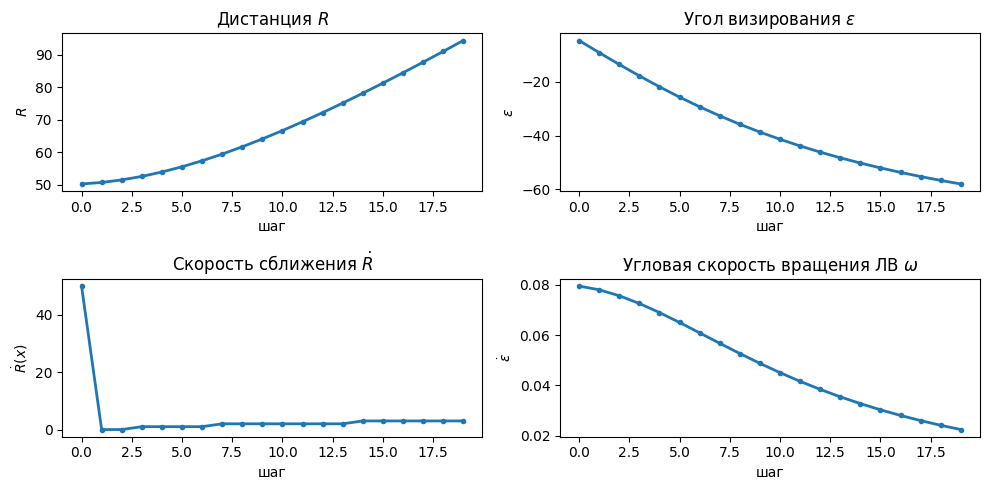


Рисунок – Результаты исследования линии  
параллельными курсами

Как можно заметить, параметры в данном тестовом сценарии меняются предсказуемо. Дистанция между объектами растет, с непостоянным значением, что логично. Угол визирования также растет. В данном примере хотелось бы сравнить корректность расчета угла визирования.

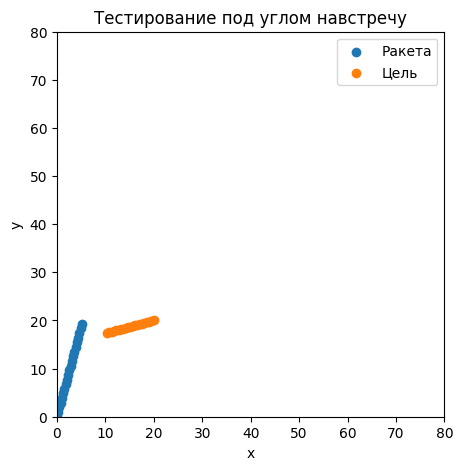
Траектория представляет собой одинаковые прямоугольники со сторонами 20, 25 и .

Таким образом, угол визирования можно вычислить:

(9)

Можно обратить внимание, что модель дает аналогичные значения.

В финальном тесте будет интересовать, прежде всего, корректность измерения скорости вращения угла линии визирования.



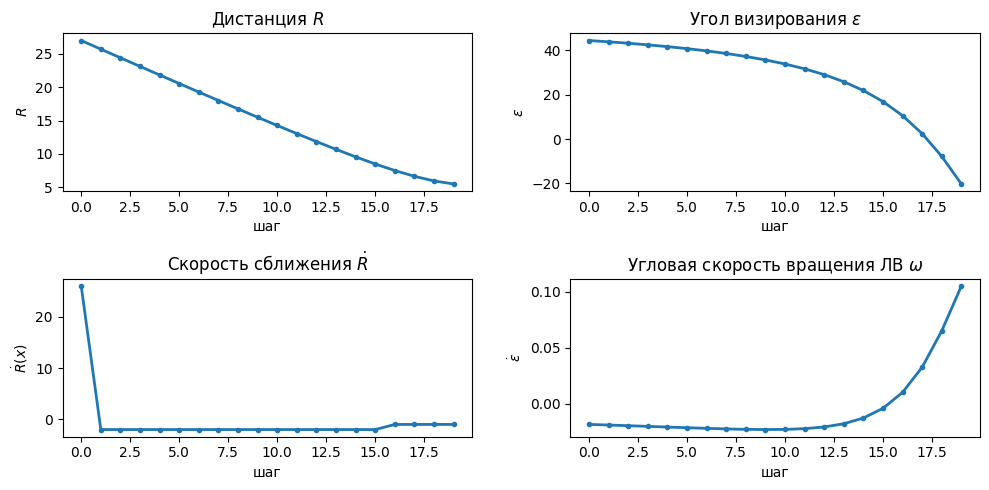


Рисунок – Результаты исследования линии  
навстречу под углом

В данном примере хотелось продемонстрировать корректность расчета угловой скорости вращения ЛВ:

(10)

(11)

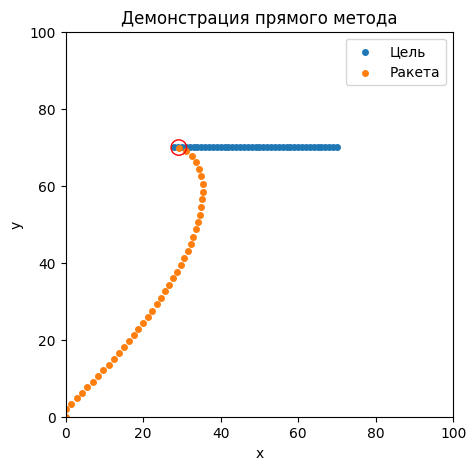
(12)

Если посмотреть численные значения, то можно увидеть, что в начальный момент времени угловая скорость вращения ЛВ равна 0.19, что соответствует рассчитанному значению. Для более детального рассмотрения примеров рекомендуется обратиться к исходному коду.

# Описание примеров реализации методов управления

## Описание примеров реализации двухточечных методов

Для более детального изучения программного кода рекомендуется обратиться к ноутбуку с исходным кодом. В данном описании приводняться, лишь конечные результаты моделирования. В примере ракета наводиться на цель, которая движется прямолинейно. Ракета пускается вертикально, со скоростью в 2 раза большей, чем скорость цели.

****

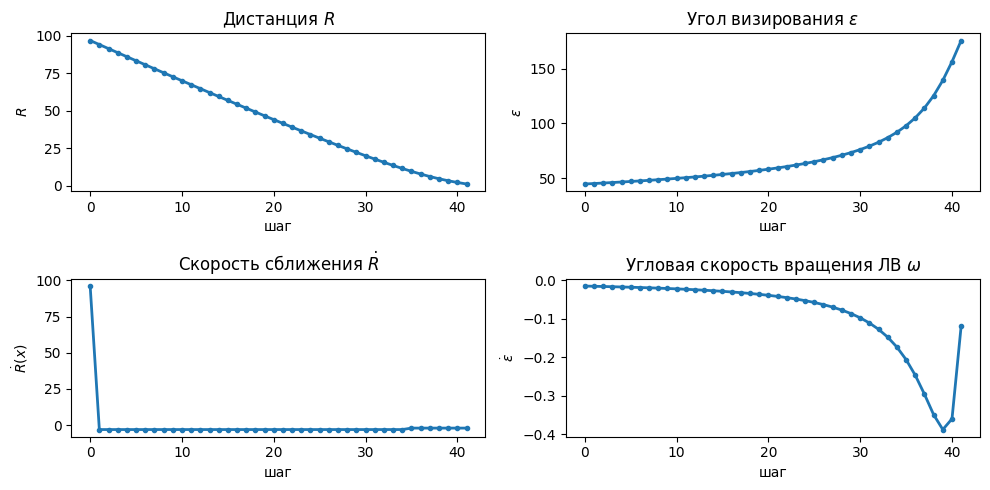
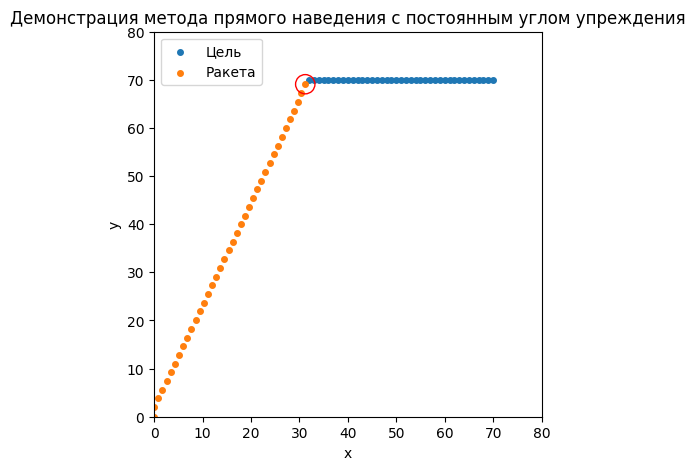


Рисунок – Результаты моделирования прямого метода

Как можно видеть на рисунке 9, метод успешно отработал. Теперь продемонстрируем прямой метод с упреждающим углом.



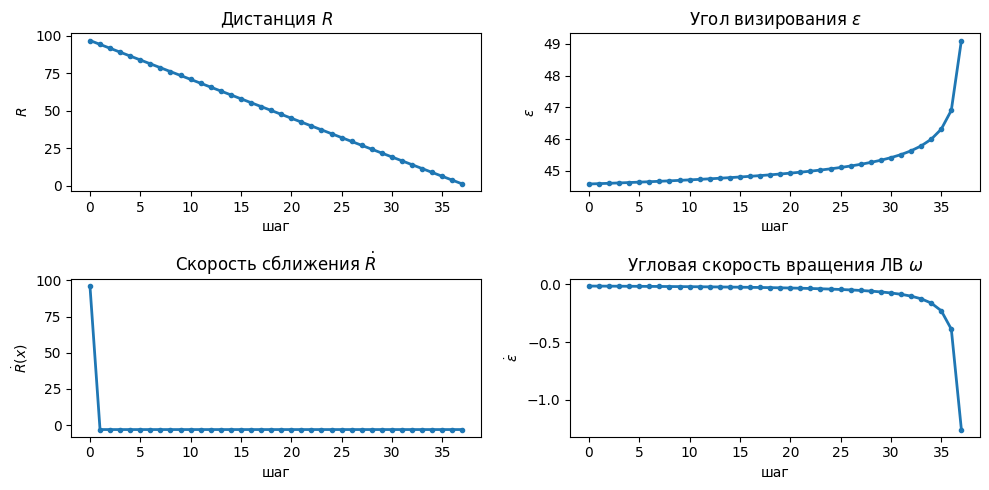


Рисунок – Результаты моделирования прямого метода  
с постоянным углом упреждения

Можно заметить на рисунке 10 характерные черты метода. Угол визирования изменяется незначительно. И траектория представляет из себя прямую.

## Описание примеров реализации трехточечных методов

Продемонстрируем пример реализации трёхточечного метода. Носитель и цель движутся навстречу друг другу с равными скоростями. А ракета пускаемая с носителя имеет скорость в 3 раза большее чем цель. Результат моделирования представлен на рисунке .

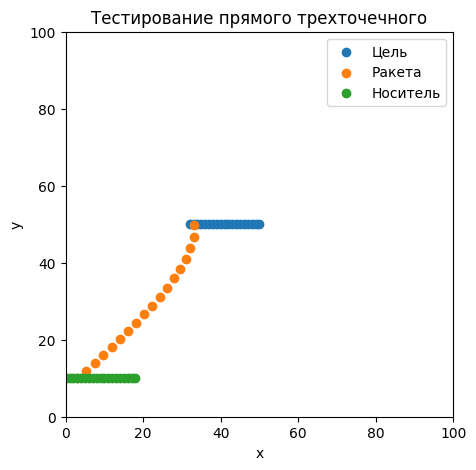


Рисунок – Результаты моделирования прямого трехточечного метода

Можно увидеть, что ракета попала в цель. В данном примере специально не добавлялась линия визирования.

## Модернизация прямого метода

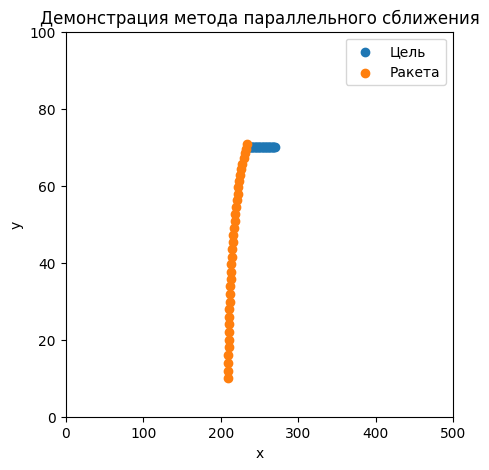
Попробуем переписать метод прямого наведения в следующий вид:

(13)

(14)

(15)

Однако при попытке моделировать данную вариацию метода не удаётся получить результат.



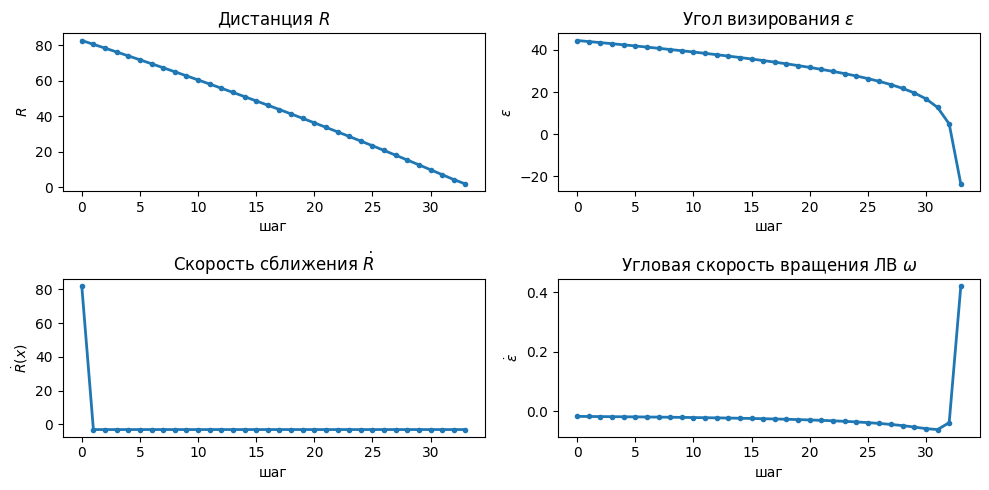


Рисунок – Результаты моделирования модернизации  
 прямого метода

Метод сработал только при следующих параметрах моделирования:

начальная координата x ракеты: 210; начальная координата y ракеты: 10; угол пуска ракеты: 90; скорость ракеты 2м\с; ускорение ракеты 0;

начальная координата x цели: 270; начальная координата y цели: 70; угол пуска цели: -180; скорость цели 1м\с; ускорение цели 0.

При изменении любого из параметров происходи срыв сопровождения. Стоит обратить внимание, что разработанная модель показала адекватное поведение в других методах наведения, что говорит о возможных теоретических ошибка в данном методе, однако причину этих ошибок не удалось установить.

# Вывод

Таким образом, в результате проделанной работы был разработан программный интерфейс (API) для дальнейшего удобного моделирования различных систем автоматического радиоуправления.

Однако стоит обратить внимание на реализацию расчета угловой скорости вращения линии визирования. Возможно, стоит изменить способ его расчета.