



Министерство науки и высшего образования Российской
Федерации
Федеральное государственное бюджетное образовательное
учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА СИСТЕМЫ УПРАВЛЕНИЯ ЛЕТАТЕЛЬНЫМИ АППАРАТАМИ (ИУ1)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 24.05.06 «Системы управления летательными
аппаратами»

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

По теме

Моделирование и анализ рабочих зон многопозиционных
радиолокационных систем с разработкой графического
интерфейса

Дисциплина

Радиолокационные и информационно-измерительные комплексы

Студенты	ИУ1-93	20/12/2024	И.И. Машков
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
	ИУ1-92	20/12/2024	Г.О. Шелуханов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
	ИУ1-91	20/12/2024	Т.С. Разомазов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель		20/12/2024	А.Н. Савельев
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2024

1. Постановка задачи

Развитие современных радиолокационных систем (РЛС) играет ключевую роль в обеспечении безопасности и управления на различных уровнях — от гражданской авиации до оборонных задач. Важным аспектом совершенствования таких систем является исследование и моделирование их рабочих зон, что позволяет оценить эффективность размещения и совместной работы РЛС.

Цель данной работы — разработка графического интерфейса для моделирования и анализа рабочих зон многопозиционных радиолокационных систем. Для достижения поставленной цели необходимо выполнение следующих задач:

- Разработка алгоритма программного модуля;
- Разработка графического интерфейса;
- Проведение моделирования частных случаев для подтверждения работоспособности разработанного модуля.

Разрабатываемое приложение должно предоставлять следующие функциональные возможности:

- Визуализация основных лепестков антенн РЛС;
- Построение фигур, для определения положения объекта, с учетом погрешностей;
- Вычисление площадей пересечения рабочих зон

В рамках данного задания предпочтительным языком программирования для выполнения поставленных задач является Python.

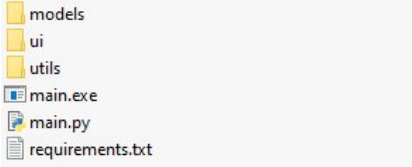
2. Практическая часть индивидуального задания

В рамках данного индивидуального задания было разработано специальное приложение на языке программирования Python. Данное приложение предназначено для моделирования двух угломерно-дальномерных систем РЛС. В данном отчете описаны порядок работы с разработанным приложением, а также некоторые важные теоретические аспекты работы, демонстрирующие работоспособность разработанного программного модуля.

2.1. Порядок установки разработанного приложения

Данное приложение было написано для использования на ОС Windows, поддерживающее работу данного приложения.

Для установки данной разработанной программы необходим любой архиватор, например 7z или WinRar. Архив с приложением имеет следующий вид (рис. 1).



models	1 763	?	Папка с файлами	17.12.2024 20:42
ui	141 885	?	Папка с файлами	17.12.2024 20:42
utils	8 715	?	Папка с файлами	17.12.2024 20:42
main.exe	101 878 450	100 649 413	Приложение	17.12.2024 19:52
main.py	255	22 243	Python File	16.12.2024 22:44
requirements.txt	315	?	Текстовый докум...	16.12.2024 23:00

Рисунок 1 — Архив с файлами для установки

Представленный архив необходимо распаковать в любую папку, после чего просто запустить файл main.exe.

Дополнительно в архиве представлен код программы, используемые компоненты и необходимые библиотеки для работы с исходным кодом.

Для запуска приложения с помощью приложенного кода необходимо:

1. Создать виртуальное окружение:

```
python -m venv .venv.venv\Scripts\activate
```

2. Установить необходимые библиотеки:

```
python -m pip install -r requirements.txt
```

3. Запустить приложение:

```
python main.py
```

2.2. Инструкция эксплуатации

При запуске программы, пользователя встречает окно, представленное ниже (рис. 2).

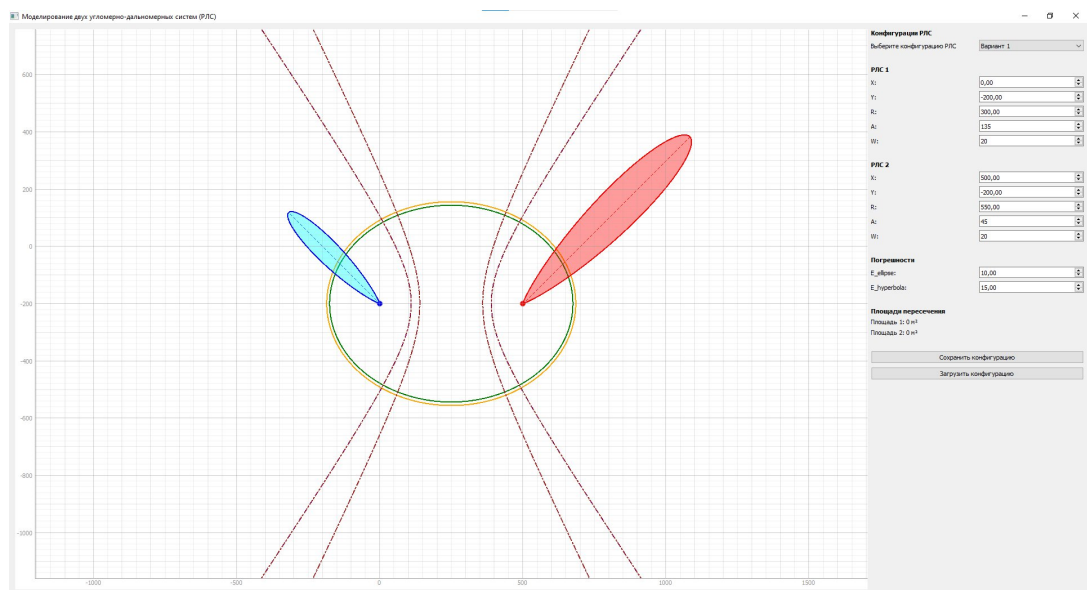


Рисунок 2 — Общий вид разработанного приложения

В главном окне представлена конфигурация РЛС, включающая в себе две РЛС станции, главные лепестки их антенн, а также соответствующие рабочие зоны. Данная конфигурация представлена на координатном полотне, которое размечено в условных единицах (у.е.)

Программа включает в себя блок управления конфигурацией (рис. 3).

Конфигурации РЛС

Выберите конфигурацию РЛС

Вариант 1

▼

РЛС 1

X:

Y:

R:

A:

W:

РЛС 2

X:

Y:

R:

A:

W:

Погрешности

E_ellipse:

E_hyperbola:

Площади пересечения

Площадь 1: 0 м²

Площадь 2: 0 м²

Сохранить конфигурацию

Загрузить конфигурацию

Рисунок 3 — Блок управления конфигурацией

В данном блоке задаются первичные параметры моделируемой системы. Для каждой РЛС существует свой набор параметров:

- Координаты РЛС – параметры X и Y ;
- Дальность действия РЛС – параметр *Radius* (R);
- Угол направления главного лепестка – параметр A (*Angle*);
- Ширина главного лепестка – параметр W (*Wide*).

В блоке Конфигурация РЛС можно выбрать несколько вариантов, которые рассматривались в рамках данной индивидуальной работы. Всего существует три конфигурации, которые имеют названия «Вариант 1», «Вариант 2», «Вариант 3» для различных комбинаций угломерных и дальномерных конфигураций РЛС. Рассмотрим каждую из них подробнее.

2.2.1. Конфигурация «Вариант 1».

Рассмотрим первый вариант конфигурации (рис. 4).

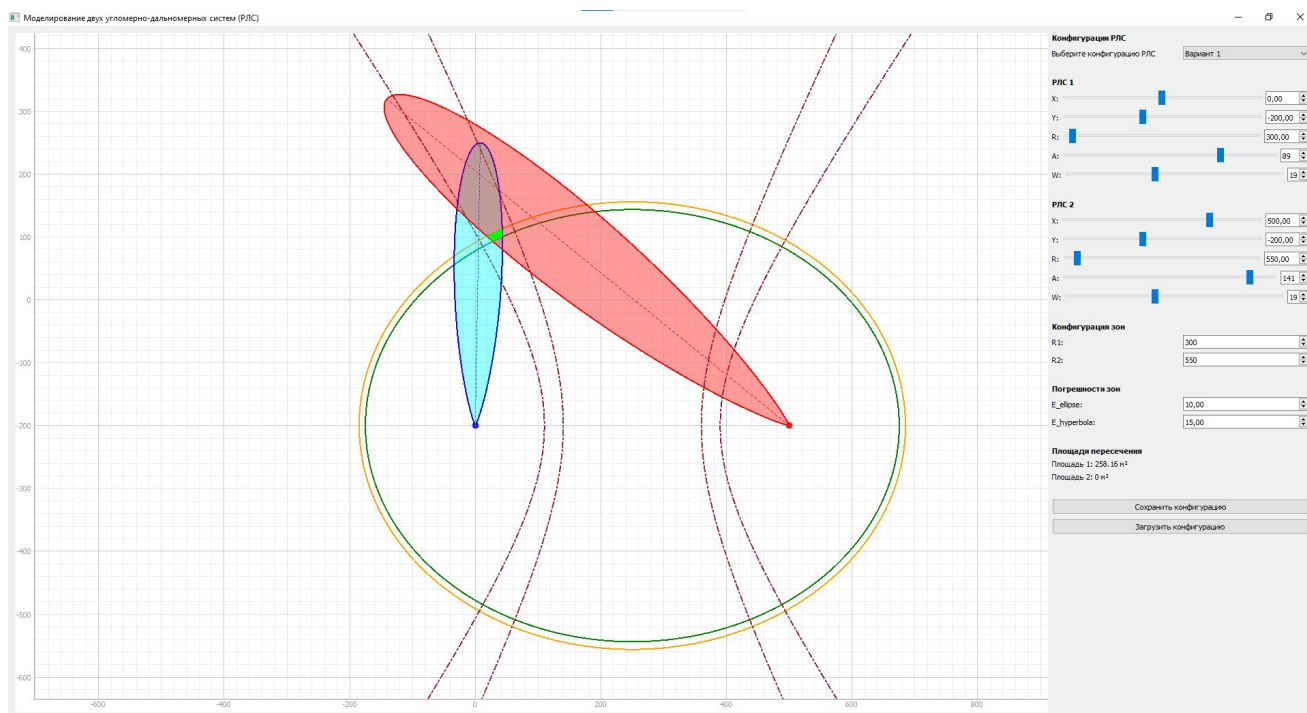


Рисунок 4 — Пример конфигурации №1

Каждая конфигурация характеризуется своим дополнительным набором параметров. Так, в случае первой конфигурации, рассматривается расчет площади элемента разрешения угломерно-дальномерной двухпозиционной системы. При такой конфигурации, площадь будет рассчитана только в том случае, если оба луча будут направлены в область пересечения погрешностей. Увеличенная схема представлена ниже (рис. 5).

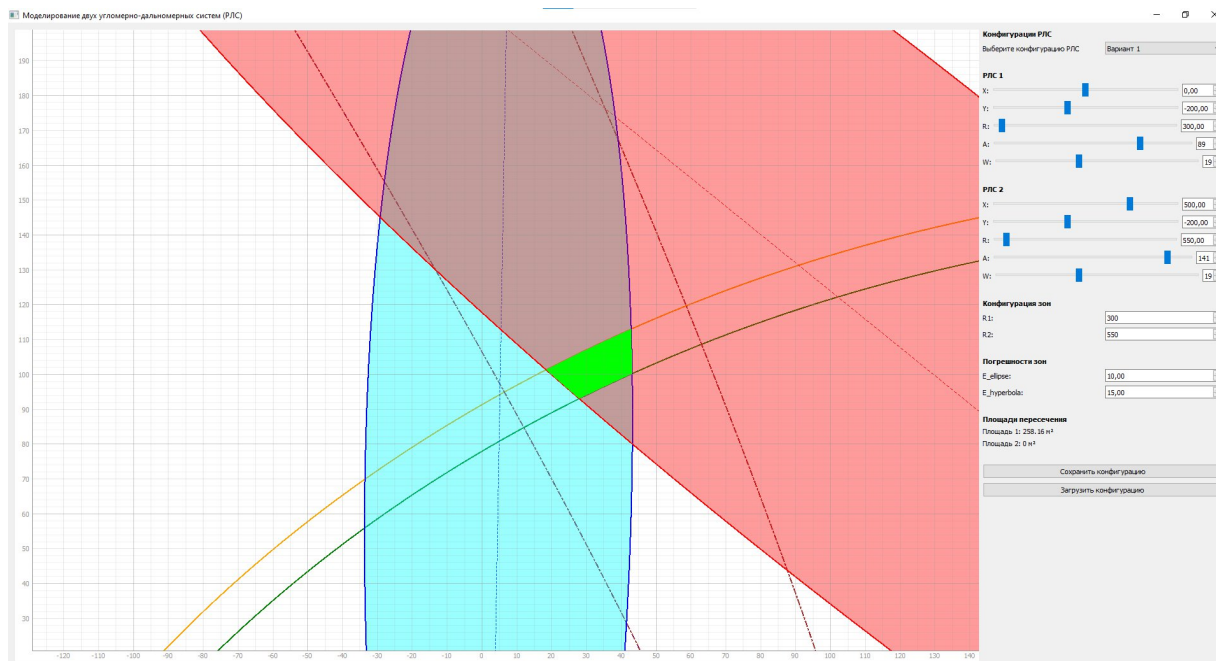


Рисунок 5 — Детальное изображение элемента разрешения

Дополнительно, каждый из параметров погрешностей можно задать при помощи следующих параметров:

- $E_{ellipse}$ – параметр, отвечающий за погрешность окружности.
- $E_{hyperbola}$ – параметр, отвечающий за погрешность гипербол.

В результате работы конфигурации искомая площадь закрашивается зеленым цветом, а ее значение выводится справа снизу в условных единицах (y.e.)

2.2.2. Конфигурация «Вариант 2».

Рассмотрим второй вариант конфигурации (рис. 6).

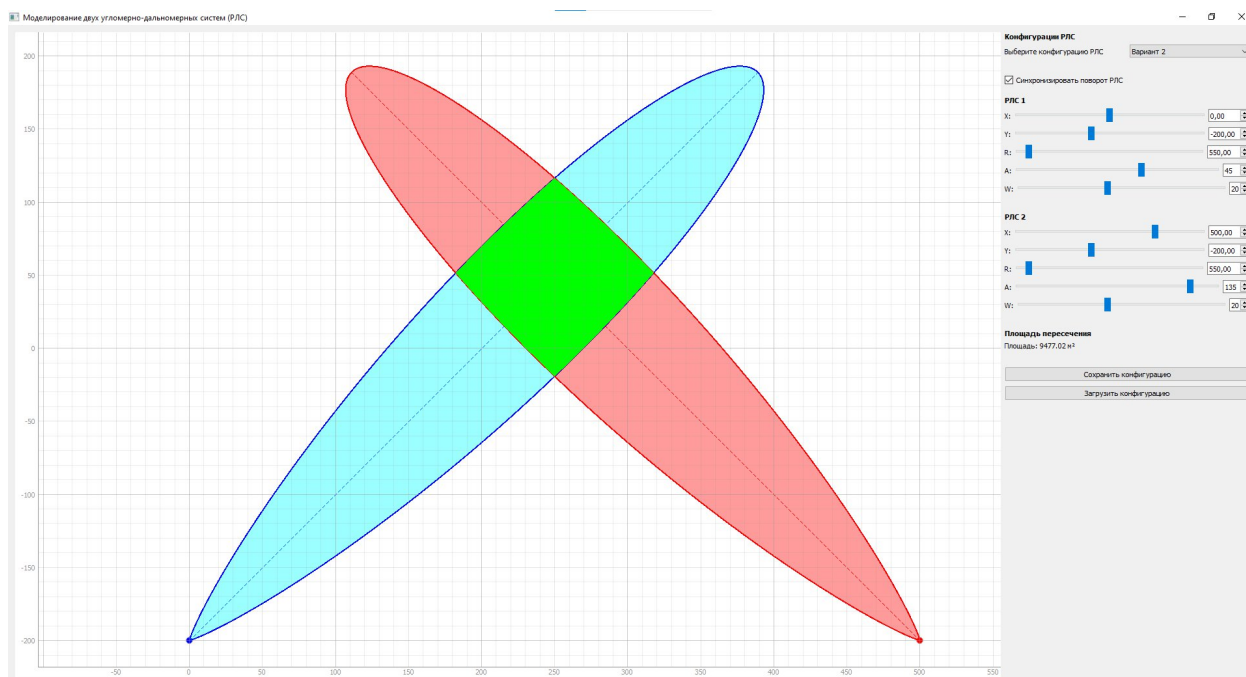


Рисунок 6 — Пример конфигурации №2

Данный вариант предназначен для исследования влияния взаимных углов направления главных лепестков на вид элемента разрешения. Для этого во вкладку параметров добавлена опция «Синхронизировать поворот РЛС», при включении которой лепестки можно поворачивать одновременно, изменяя параметр угла направления лепестка в одном из РЛС. Площадь закрашивается зеленым цветом, а ее величина указана справа снизу в у.е.

Рассмотрим еще некоторые виды конфигураций, которые приводят к различным видам элементов разрешения (рис. 7, рис. 8).



Рисунок 7 — «Вытянутая кверху» форма элемента разрешения

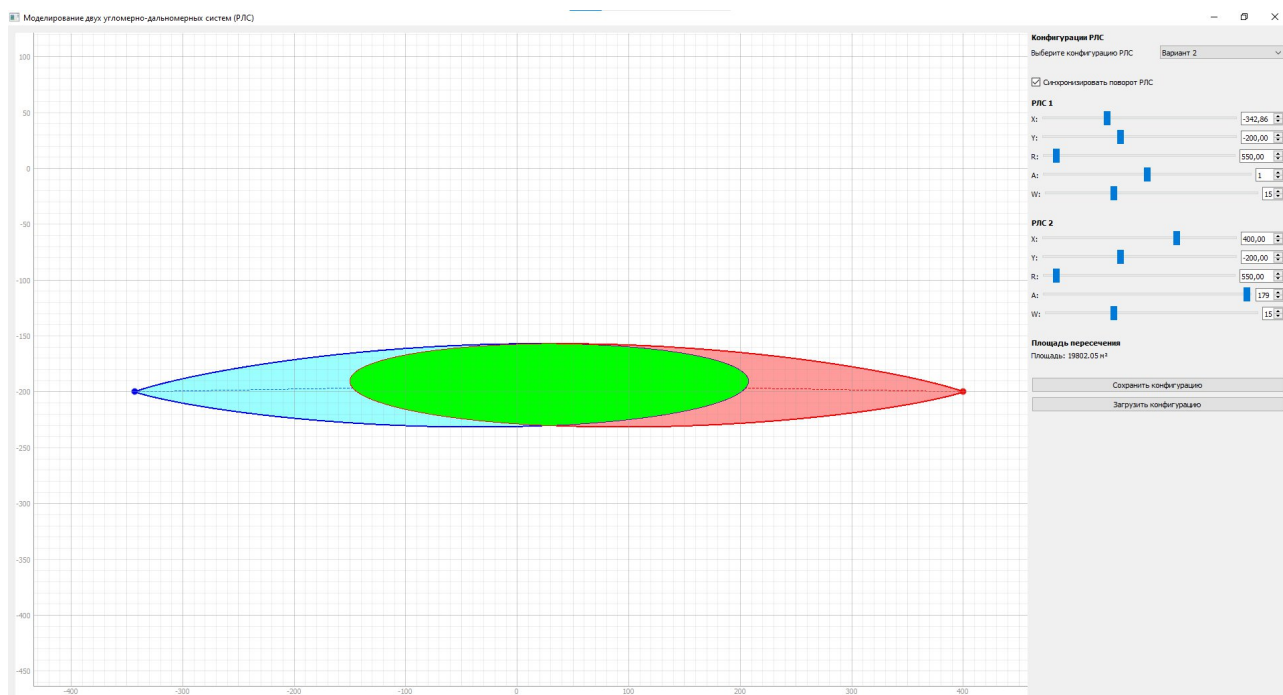


Рисунок 8 — Эллипсоидная форма элемента разрешения

2.2.3. Конфигурация «Вариант 3».

Рассмотрим третий вариант конфигурации (рис. 6).

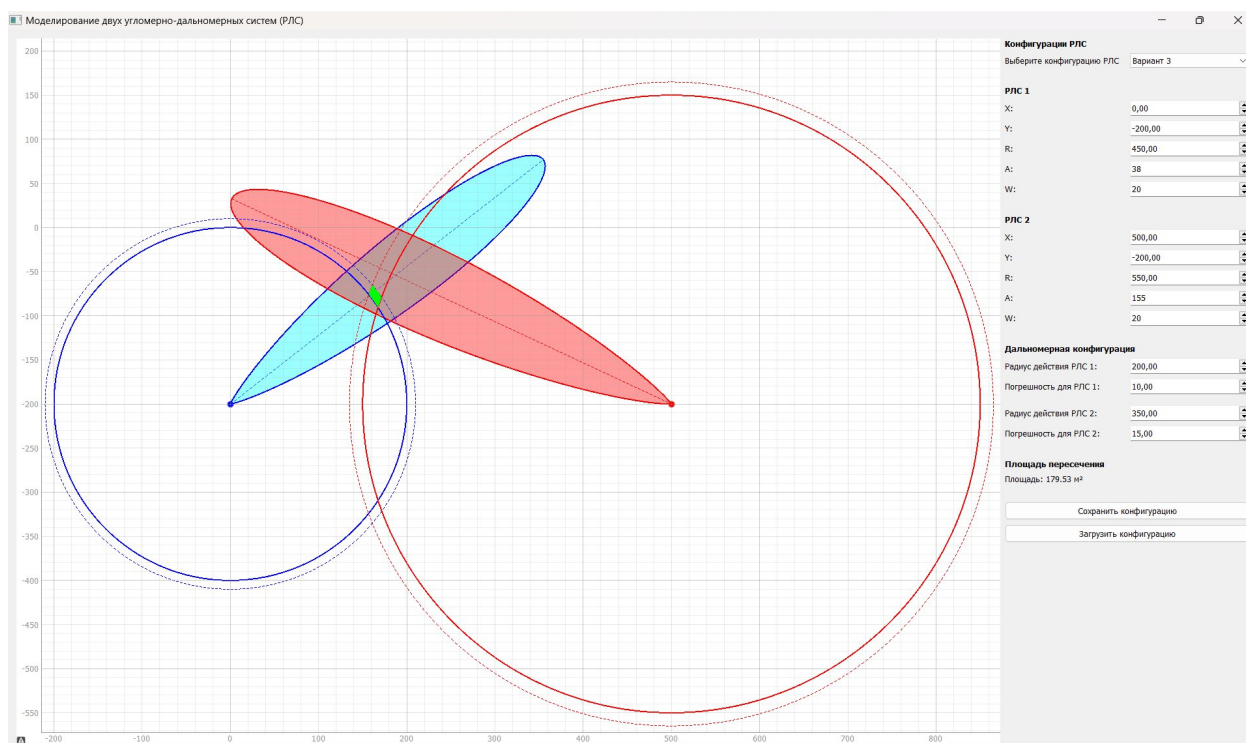


Рисунок 6 — Пример конфигурации №3

Третья конфигурация является комбинация угломерно-дальномерной двухпозиционной РЛС. Как и для ранее рассмотренных, каждая из РЛС по отдельности имеет общие параметры, которые можно редактировать в блоке управления конфигурацией РЛС. Искомая площадь элемента разрешения закрашена зеленым цветом, а ее значение представлено справа снизу в у.е. Увеличенная схема представлена ниже (рис. 10).

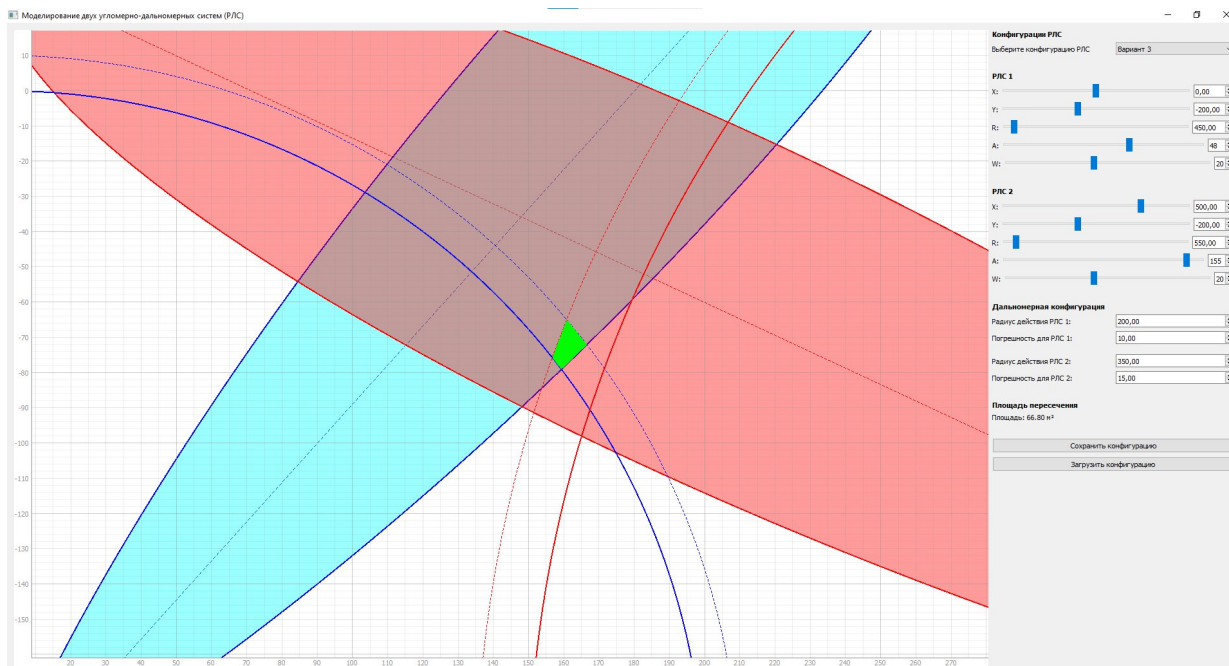


Рисунок 10 — Детальное изображение элемента разрешения

Дополнительно представлены изменяемые параметры в блоке «Дальномерная конфигурация». В ней можно задавать соответствующие параметры:

- Радиус действия РЛС 1;
- Погрешность для РЛС 1;
- Радиус действия РЛС 2;
- Погрешность для РЛС 2.

2.3. Принцип работы разработанного приложения

Рассмотрим основные положения, на которых основывается работа разработанного приложения.

Площади пересечения областей моделируются с использованием библиотек *Shapely* и *numpy*. Рассмотрим два основных типа областей:

- Главные лепестки антенн;
- Фигуры положений (эллипсы и гиперболы) с учетом погрешностей.

2.3.1. Построение диаграмм направленности антенн (ДНА)

Рассмотрим процесс построения областей ДНА.

1) Задаются параметры РЛС:

- X, Y – координаты антенны
- R – радиус действия антенны
- A – угол направления ДНА
- W – ширина диаграммы

2) Формируется диапазон углов для построения ДНА:

```
angles_deg = np.linspace(-360, 360, 3600)
theta = np.deg2rad(angles_deg)
```

3) Вычисляется затухание на границах ДНА:

Формула затухания для построения диаграммы направленности антенны представлена ниже:

$$L(\theta) = e^{-\frac{\ln(2) \cdot (\theta - A)^2}{(W/2)^2}}$$

где $\ln(2)$ – логарифмическое значение для нормализации

θ – текущий угол в радианах

A – угол направления ДНА

W – ширина диаграммы

4) Построение полигонов ДНА:

Координаты областей рассчитываются на основе углов и радиуса действия антенны:

```
x_fill = L(angles) * max_range * np.cos(theta) + position_x
y_fill = L(angles) * max_range * np.sin(theta) + position_y
```

Далее создается полигон с использованием библиотеки *Shapely*:

```
poly1 = Polygon(np.column_stack((x_fill, y_fill)))
```

2.3.2. Построение эллипсов и гипербола

Эллипсы строятся на основе суммы радиусов действия двух антенн. Формула для параметра c :

$$c = R1 + R2$$

Для построения эллипса вызывается функция *compute_ellipse*, которая возвращает координаты эллипса:

$$(x, y) = \text{compute_ellipse}(r1s1, r1s2, c)$$

В задаче используется два эллипса. Первый эллипс строится для идеального случая без учета погрешностей. Второй эллипс строится с учетом погрешностей, добавленных к радиусам $R1$ и $R2$:

$$c = (R1 + E_ellipse) + (R2 + E_ellipse)$$

Гипербола моделируется как разность радиусов действия двух антенн. Формула для параметра ch :

$$ch = R2 - R1$$

Для учета погрешностей, аналогично эллипсам, строятся две дополнительные гиперболы – первая гипербола с положительной погрешностью $(ch + 2 \cdot E_hyperbola)$ и вторая гипербола с отрицательной погрешностью $(ch - 2 \cdot E_hyperbola)$.

Координаты гиперболы вычисляются функцией *compute_hyperbola*, которая возвращает две ветви гиперболы:

$$\begin{aligned} \text{hyperbola_x1}, \quad \text{hyperbola_y1}, \quad \text{hyperbola_x2}, \quad \text{hyperbola_y2} &= \\ \text{compute_hyperbola}(r1s1, r1s2, c_h) \end{aligned}$$

Получаются две гиперболы, которые отображаются на графике с учетом погрешностей:

- Гипербола 1: $R2 - R1 = ch$
- Гипербола 2: $R1 - R2 = ch$

2.3.3. Вычисление пересечений

Рассмотрим подробно процесс вычисления пересечений.

1) Создание полигонов

Все области (главные лепестки антенн, эллипсы и гиперболы) конвертируются в объекты *Polygon* из библиотеки *Shapely*.

Визуализация главных лепестков формируются на основе вычисленных координат:

```
poly1 = Polygon(coords_1)
```

```
poly2 = Polygon(coords_2)
```

Эллипсы и гиперболы также преобразуются в полигональные области.

2) Пересечение областей

Для поиска общей области между различными зонами используется метод *.intersection*. Этот метод находит геометрическое пересечение между полигонами:

```
intersection = poly1.intersection(poly2)
```

3) Вычисление разностей для учета погрешностей

Для учета погрешностей строятся внешние и внутренние границы областей. Метод *.difference* позволяет определить разницу между внешней и внутренней границей:

```
ring = outer_polygon.difference(inner_polygon)
```

Например, для эллипса разница между большим эллипсом (с погрешностью) и идеальным эллипсом формирует "кольцо" погрешности.

Аналогично строятся зоны погрешностей для гипербол.

4) Объединение пересечений

Конечная область пересечения формируется путем объединения всех пересечений и учета погрешностей:

```
final_intersection =
```

```
poly1.intersection(ellipse_difference).intersection(hyperbola_difference)
```

В результате получается область, которая учитывает как пересечение зон покрытия, так и погрешности измерений.

5) Проверка и обработка результата

Для проверки пересечений используется метод *.is_empty*:

- Если пересечение не пустое, вычисляется площадь
- Если пересечение пустое, площадь равна нулю

6) Вывод результата

Площадь пересечения вычисляется с помощью метода *.area* и выводится на экран:

```
area = final_intersection.area
```

Результат подсчета площади отображается в панели управления в виде «Площадь: X.XX м²». Для наглядности, на графике пересекающиеся области отображаются в виде зеленых закрашенных площадных фигур.

2.3.4. Метод *.area*

Метод *.area* в библиотеке *Shapely* используется для вычисления площади геометрических объектов, таких как *Polygon* (многоугольник).

Рассмотрим принцип работы метода *.area*. Метод *.area* возвращает числовое значение, равное площади двумерного геометрического объекта в его собственной системе координат.

Площадь вычисляется на основе координат всех вершин многоугольника. Метод учитывает как основную границу многоугольника (наружный контур), так и любые внутренние отверстия (вложенные полигоны).

Площадь многоугольника, заданного последовательностью вершин $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, вычисляется по формуле:

$$\text{Area} = \frac{1}{2} \left| \sum_{i=1}^{n-1} (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i) \right|$$

где n – количество вершин многоугольника

(x_i, y_i) – координаты i -й вершины

Координаты вершин перечисляются по часовой или против часовой стрелке.

Погрешность вычисления площади заданного многоугольника исходит из представления чисел с плавающей запятой в ОЗУ компьютера, а также обеспечивается комплексностью геометрического объекта и многочисленными операциями над полигонами. Примерная погрешность составляет 0.1 у.е.^2 .

ЗАКЛЮЧЕНИЕ

В рамках данного индивидуального задания была разработана вспомогательная специальная программа-приложение, которое может быть использовано для проведения дальнейших анализов в рамках указанных ограничений.

Для написания приложения использовался широко применяющийся язык программирования Python. Методы и библиотеки, используемые в рамках данного индивидуального задания, были описаны и пояснены. Порядок установки и использования полученной программы получился понятным и простым, программа имеет небольшой вес и не является излишне требовательной.

В результате выполнения данного индивидуального задания студентами было получено визуальное представление о элементах разрешения, их размерах и формах, характере их появления. Полученные в результате выполнения данного индивидуального задания способствуют лучшему восприятию лекционного материала и помогают лучше освоить изучаемую дисциплину.

Представленный в формате архива исходный код может быть дополнительно использован для модификации программы под собственные нужды, что делает приложение отличным инструментом для дальнейшего обучения.