**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**
федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации
*(наименование института, филиала, факультета)*

Кафедра систем автоматизированного проектирования
*(наименование кафедры)*

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
*(код и наименование направления подготовки)*
Образовательная программа: «Искусственный интеллект и системы автоматизированного проектирования»
*(наименование профиля/специализации/образовательной программы)*

К защите допустить
Зав. каф. САПР
Чермошенцев С.Ф. (ФИО)
«___»
_____20__г.

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему: «Автоматизация размещения антенн на летательном аппарате на основе нейронных сетей»

ОБУЧАЮЩИЙСЯ Амануллин Марсель Фахимович _____
*(фамилия, имя, отчество)* *(подпись)*

РУКОВОДИТЕЛЬ к.т.н., доцент. Гайнутдинов Р.Р _____
*(ученая степень, звание, фамилия, имя, отчество)* *(подпись)*

Казань 2025 г.

# Automation of antenna placement on aircraft based on neural networks

*By*
*Amanullin Marsel Fahimovich*

Submitted to the Department of Computer-aided design

in partial fulfillment of the Requirements for the degree of

BACHELOR OF SCIENCE

at the

Federal State Budgetary Educational Institution of Higher Education
«Kazan National Research Technical University named after A.N.Tupolev-KAI»
(KNRTU-KAI)

| | | |
|---|---|---|
| Author | _____ *(signature)* | Amanullin Marsel Fahimovich |
| Supervisor | _____ *(signature)* | Rustam Rafkatovich Gainurdinov<br>*Candidate of Technical Sciences, Associate Professor of the Department of Computer-aided design* |
| Certified by | _____ *(signature)* | Sergei Fyodorovich Chermoshentsev<br>*Doctor of Technical Science, Head of the Department of Computer-aided design* |
| date | _____ | |

Kazan 2025

**федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ»**
**(КНИТУ-КАИ)**

Институт (факультет), филиал *Институт Компьютерных технологий и защиты информации*
Кафедра *систем автоматизированного проектирования*
Направление подготовки/специальность *09.03.01 «Информатика и вычислительная техника»,*
*профиль «Искусственный интеллект и системы автоматизированного проектирования»*

**ЗАДАНИЕ**
на выпускную квалификационную работу

обучающегося *Амануллина Марселя Фахимовича*
(фамилия, имя, отчество)

1 Тема выпускной квалификационной работы
*Автоматизация размещения антенн на летательном аппарате на основе нейронных сетей*

утверждена приказом №_____ от «____»_____20__г.

2 Срок сдачи обучающимся законченной ВКР «____»_____20__г.

3 Исходные данные к выпускной квалификационной работе_____
*Геометрические и электрофизические параметры антенных систем и летательного аппарата*

4 Перечень подлежащих разработке вопросов и исходные данные к ним:

***Глава 1. Анализ проектной процедуры автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.***

*1.1. Современные тенденции и проблемы автоматизации размещения антенн на летательном аппарате.*

*1.2. Разработка функциональных и поведенческих моделей (IDEF0 и IDEF3) проектной процедуры автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.*

*1.3. Применение нейронных сетей в задачах автоматизации проектирования.*

*1.4. Цель и задачи проектной процедуры автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.*

*Выводы по главе 1.*

***Глава 2. Разработка математического обеспечения для автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.***

*2.1. Содержательная и математическая постановка задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.*

*2.2. Методы и алгоритмы решения задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.*

*2.3. Решение задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.*

*Выводы по главе 2.*

***Глава 3. Разработка информационного обеспечения автоматизированной процедуры размещения антенн на летательном аппарате на основе нейронных сетей.***

*3.1. Разработка концептуальной модели базы данных.*

*3.2. Разработка логической модели базы данных.*

*3.3. Физическое проектирование базы данных.*

*Выводы по главе 3.*

***Глава 4. Разработка программного обеспечения размещения антенн на летательном аппарате на основе технологий нейронных сетей.***

*4.1. Анализ функциональных требований к программе.*

*4.2. Архитектура программы.*

*4.3. Разработка пользовательского интерфейса программы.*

*Выводы к главе 4.*

*Заключение.*

*Список литературы.*

*Приложение 1. Глоссарий (словарь терминов предметной области).*

*Приложение 2. Образцы проектной документации.*

*Приложение 3. Листинг программы.*

5 Перечень графического материала (при наличии):

*1. Тема выпускной квалификационной работы (1 слайд).*
*2. Актуальность выпускной квалификационной работы (1 слайд).*
*3. Цель и задачи выпускной квалификационной работы (1 слайд).*
*4. Функциональная модель IDEF0 проектной процедуры автоматизации размещения антенн на летательном аппарате на основе нейронных сетей (4 слайда).*
*5. Формальная (математическая) постановка задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей (2 слайда).*
*6. Алгоритм решения задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей (1 слайд).*
*7. Логическая модель базы данных (1 слайд).*
*8. Архитектура (UML-диаграмма компонентов) программы (1 слайд).*
*9. Экранные формы программы (5 слайдов).*
*10. Основные выводы по выпускной квалификационной работе (1 слайд).*

6 Консультанты по ВКР (при их наличии, с указанием относящихся к ним разделов):

*(наименование раздела, ФИО консультанта, подпись)*

*(наименование раздела, ФИО консультанта, подпись)*

*(наименование раздела, ФИО консультанта, подпись)*

*(наименование раздела, ФИО консультанта, подпись)*

Дата выдачи задания «____»_____ 20___г.

Руководитель ВКР _____ _____
(подпись) (ФИО)

Задание к исполнению принял _____ _____
(подпись) (ФИО)

**Календарный план выполнения ВКР**

| № п/п | Наименование этапов (разделов) выпускной квалификационной работы | Срок выполнения этапов (разделов) ВКР | Примечание |
|---|---|---|---|
| *1.* | *Современные тенденции и проблемы автоматизации размещения антенн на летательном аппарате.* | *31.10.2024 – 12.03.2025 г.* | |
| *2.* | *Разработка функциональных и поведенческих моделей (IDEF0 и IDEF3) проектной процедуры автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.* | *13.03.2025 – 29.03.2025 г.* | |
| *3.* | *Применение нейронных сетей в задачах автоматизации проектирования.* | *20.03.2025 – 26.03.2025 г.* | |
| *4.* | *Цель и задачи проектной процедуры автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.* | *27.03.2025 – 2.04.2025 г.* | |
| *5.* | *Содержательная и математическая постановка задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.* | *3.04.2025 – 9.04.2025 г.* | |
| *6.* | *Методы и алгоритмы решения задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.* | *10.04.2025 – 16.04.2025 г.* | |
| *7.* | *Решение задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.* | *17.04.2025 – 23.04.2025 г.* | |
| *8.* | *Разработка концептуальной модели базы данных.* | *24.04.2025 – 30.04.2025 г.* | |
| *9.* | *Разработка логической модели базы данных.* | *1.05.2025 – 7.05.2025 г.* | |
| *10.* | *Физическое проектирование базы данных.* | *8.05.2025 – 14.05.2025 г.* | |
| *11.* | *Анализ функциональных требований к программе.* | *15.05.2025 – 21.05.2025 г.* | |
| *12.* | *Архитектура программы.* | *22.05.2025 – 28.05.2025 г.* | |
| *13.* | *Разработка пользовательского интерфейса программы.* | *29.05.2025 – 2.06.2025 г.* | |
| *14.* | *Примеры решения задачи размещения антенн в летательном аппарате на основе нейронных сетей.* | *31.05.2025 г. - 5.06.2025 г.* | |
| *15.* | *Подготовка к защите выпускной квалификационной работы.* | *5.06.2025 г.* | |

Обучающийся _____ _____ _____
*(подпись)* *(Фамилия, инициалы)* *( дата)*

Руководитель _____ _____ _____
*(подпись)* *(Фамилия, инициалы)* *(дата)*

# АННОТАЦИЯ

Данная исследовательская работа посвящена разработке подсистемы для автоматизации процесса размещения антенных систем на беспилотном летательном аппарате с интеграцией средств прогнозирования электромагнитной совместимости на основе методов машинного обучения.

В рамках исследования был выполнен анализ современных подходов к проектированию антенных систем, сформулирована математическая модель задачи размещения с учётом взаимных помех, разработан алгоритм формирования обучающих выборок на основе электродинамического моделирования, обучена нейросетевая модель для прогнозирования коэффициента связи, спроектирована база данных для хранения проектных данных и результатов моделирования, а также создан программный модуль с графическим интерфейсом для автоматизированного проектирования.

Работа обладает высокой актуальностью в условиях роста требований к электромагнитной совместимости радиоэлектронных средств на борту летательного аппарата и усложнения процессов размещения оборудования. Разработанная подсистема расширяет функциональные возможности существующих систем автоматизированного проектирования, позволяя инженерам существенно ускорить и упростить процесс размещения антенн с учётом критериев электромагнитной совместимости.

Ключевые слова: беспилотный летательный аппарат, автоматизация, размещение антенн, электромагнитная совместимость, машинное обучение, проектирование.

Количество страниц: 138

Количество иллюстраций: 45

Количество таблиц: 36

Количество приложений: 3

Количество использованных источников: 50

## ANNOTATION

This research work is devoted to the development of a subsystem for automating the process of placing antenna systems on an unmanned aerial vehicle with the integration of electromagnetic compatibility prediction tools based on machine learning methods.

As part of the study, an analysis of modern approaches to the design of antenna systems was performed, a mathematical model of the placement problem was formulated taking into account mutual interference, an algorithm for generating training samples based on electrodynamic simulations was developed, a neural network model was trained to predict the coupling coefficient, a database was designed for storing design data and simulation results, and a software module with a graphical interface for automated design was created.

The work is highly relevant in the context of increasing requirements for electromagnetic compatibility of radio electronic equipment on board an aircraft and the complication of equipment placement processes. The developed subsystem expands the functionality of existing automated design systems, allowing engineers to significantly speed up and simplify the process of placing antennas taking into account electromagnetic compatibility criteria.

Keywords: unmanned aerial vehicle, automation, antenna placement, electromagnetic compatibility, machine learning, design.

Number of pages: 138

Number of illustrations: 44

Number of tables: 34

Number of appendices: 3

Number of sources used: 50

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Автоматизация проектирования летательных аппаратов остаётся одной из приоритетных задач современной авиационной промышленности [10]. В условиях роста сложности конструкций и увеличения числа функциональных систем особое значение приобретает задача корректного размещения антенн радиочастотных систем на поверхности летательного аппарата. Неправильное размещение антенн может привести к снижению эффективности передачи и приёма сигнала, появлению электромагнитных помех и ухудшению аэродинамических характеристик аппарата [15].

Размещение антенн на летательном аппарате представляет собой многокритериальную задачу, требующую учёта множества факторов: габаритные ограничения и формы поверхности, электромагнитная совместимость, углы обзора, взаимное влияние радиосигналов, а также ограничения, накладываемые конструкцией фюзеляжа и технологическими допусками. Кроме того, необходимо учитывать требования по центровке, условия эксплуатации и наличие других электронных систем, расположенных в непосредственной близости.

С ростом числа радиочастотных устройств на борту возрастает и риск перекрёстных помех и снижения помехоустойчивости [25]. Это делает особенно актуальной задачу автоматизации размещения антенн с учётом электромагнитной совместимости и оптимальных условий функционирования каждой антенны. Традиционные подходы к решению этой задачи базируются на эвристиках или инженерном опыте и требуют значительных временных затрат, при этом не всегда приводят к оптимальным решениям.

В настоящее время одним из наиболее перспективных направлений автоматизации инженерных задач является использование методов машинного обучения, в частности нейронных сетей [1]. Нейросетевые модели способны анализировать сложные зависимости между геометрией летательного аппарата, характеристиками антенн и параметрами размещения,

а также находить эффективные решения, минимизирующие помехи и максимизирующие зону покрытия.

Целью настоящей работы является разработка автоматизированной процедуры размещения антенн на поверхности летательного аппарата с применением нейронных сетей. Предполагается построение функциональных и поведенческих моделей процесса, математическая формализация задачи, разработка алгоритма размещения с применением искусственного интеллекта, а также создание базы данных антенн и размещений для последующего анализа и переобучения модели [41].

Практическая значимость работы заключается в сокращении времени проектирования, снижении ошибок, связанных с размещением, и повышении качества итоговых решений. Кроме того, предлагаемый подход может быть адаптирован под различные типы летательных аппаратов и антенн, обеспечивая универсальность и масштабируемость решения.

# ГЛАВА 1. АНАЛИЗ ПРОЕКТНОЙ ПРОЦЕДУРЫ АВТОМАТИЗАЦИИ РАЗМЕЩЕНИЯ АНТЕНН НА ЛЕТАТЕЛЬНОМ АППАРАТЕ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ.

## 1.1. Современные тенденции и проблемы автоматизации размещения антенн на летательном аппарате.

Автоматизация размещения антенн на летательных аппаратах становится всё более востребованной задачей в связи с усложнением радиочастотной архитектуры современных воздушных платформ. Инженеры сталкиваются с необходимостью размещения большого количества антенн на ограниченной площади фюзеляжа, соблюдая при этом требования к электромагнитной совместимости, направленности, зоне покрытия и конструктивной реализуемости [15]. В таких условиях ручной подход к проектированию становится всё менее эффективным.

Современные тенденции в области автоматизации процесса размещения антенн включают:

- переход от ручных к автоматизированным методам проектирования, основанным на использовании CAD/CAE-сред и алгоритмов геометрического анализа [9];
- широкое применение методов численного моделирования (например, CST Studio Suite) для оценки электромагнитной совместимости и качества связи [2];
- использование эвристических методов оптимизации, таких как генетические алгоритмы и метод имитации отжига, для перебора возможных конфигураций [4];
- формализация требований технического задания, включая ограничения по расстоянию между антеннами, зонам монтажа, приоритетам каналов и так далее.
- использование многокритериальных подходов, позволяющих учитывать не только коэффициент связи, но и зону покрытия, приоритет каналов, монтажные ограничения и так далее [6].

К перспективным и активно исследуемым направлениям, к которым относится и данная работа, можно отнести:

1. Внедрение моделей машинного обучения для прогнозирования параметров ЭМС (в частности, коэффициента связи) без необходимости проведения ресурсоёмкого моделирования [1].
2. Формирование и использование обучающих баз данных, содержащих параметры размещения и результаты моделирования, как основу для интеллектуального анализа и ускоренного проектирования.
3. Интеграцию прогнозирующих моделей в контур автоматизированного выбора размещения, что позволяет повысить эффективность и воспроизводимость проектных решений.

Несмотря на развитие упомянутых направлений, автоматизация размещения антенн на летательном аппарате остаётся технически и методологически сложной задачей. Реальные инженерные проекты часто сталкиваются с ограничениями, которые затрудняют внедрение даже известных решений [8]. Расширение функциональности воздушных платформ и рост количества каналов связи обостряют требования к размещению антенн, делая очевидной необходимость в более интеллектуальных, воспроизводимых и масштабируемых подходах. На этом фоне особенно важным становится анализ существующих проблем, сдерживающих широкое применение автоматизированных методов в практике. Ниже приведены ключевые из них:

- отсутствие единых стандартов и моделей автоматизированного размещения, что вынуждает каждую организацию разрабатывать собственные методики;
- высокая сложность моделирования, требующих значительных ресурсов и времени при оценке большого числа конфигураций [2];

- недостаточная формализация входных данных, включая описание фюзеляжа, электрофизических свойств материалов и монтажных ограничений [20];
- ограниченность существующих оптимизационных подходов, часто основанных на эвристиках (например, генетических алгоритмах), не адаптированных под конкретику антенного проектирования;
- сложность интерпретации результатов и их интеграции в инженерные процессы, особенно в условиях гибкого размещения и многовариантного проектирования.

Большинство существующих решений ориентированы либо на ручную разработку размещения с последующим анализом коэффициента связи, либо на частично автоматизированный перебор вариантов [13]. При этом почти не используются возможности предварительного прогнозирования параметров с помощью обучаемых моделей, что снижает эффективность проектирования.

Таким образом, в современных условиях особую актуальность приобретают исследования и разработки в области автоматизированных интеллектуальных систем поддержки проектирования, способных не только генерировать возможные конфигурации размещения, но и быстро оценивать их качество по целевым критериям, включая электромагнитную совместимость и надёжность связи. Это позволяет сделать проектирование более предсказуемым, воспроизводимым и масштабируемым.

## 1.2. Разработка функциональных и поведенческих моделей проектной процедуры автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.

Традиционные методы проектирования базируются на инженерном опыте, визуальной оценке зон установки и ограниченных по функциональности CAD-инструментах. При этом отсутствует универсальный инструмент, способный количественно предсказывать качество размещения – в частности, такой важный параметр, как коэффициент связи между антеннами, без запуска ресурсоёмкого моделирования в САПР-средах.

В рамках выпускной квалификационной работы предложен подход, в котором:

- генерируются примеры размещения антенн и моделируются;
- на основе результатов моделирования формируется обучающая выборка;
- создаётся нейросетевая модель, способная прогнозировать коэффициент связи по конфигурации размещения;
- разрабатывается пользовательская программа, в которой результат прогнозируется без моделирования.

Для формализации проектной процедуры применён метод IDEF0, позволяющий описать процесс автоматизации размещения антенн в терминах функций, входных и выходных данных, управляющих воздействий и используемых ресурсов [14].

Функциональная модель верхнего уровня (A-0) описывает общий контур проекта: от исходной документации и нормативных ограничений до формирования результата – компоновочной схемы с прогнозируемым коэффициентом связи, пригодной для использования в инженерной практике и системах автоматизированного проектирования.

Данная модель представлена на рисунке 1:

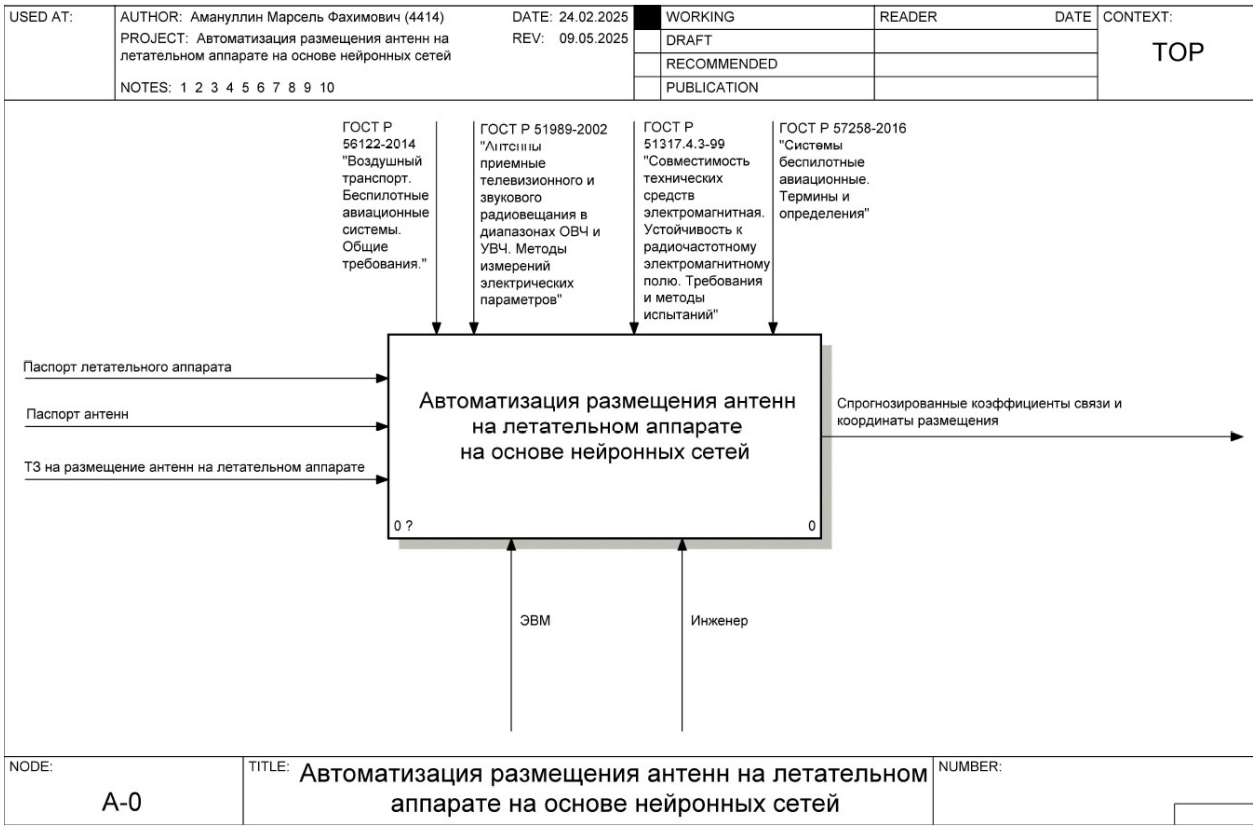


Рис. 1. Диаграмма IDEF0 (A-0)

На уровне A-0 выполняется основная функция: «Автоматизация размещения антенн на летательном аппарате на основе нейронных сетей». Данная функция реализует процесс формирования обучающей выборки на основе моделирования, обучения модели прогнозирования коэффициента связи и её дальнейшего применения для оценки пользовательских конфигураций размещения антенн.

1. Входные потоки данных диаграммы IDEF0 (A-0)

На уровне A-0 в качестве входных данных используются:

- Паспорт летательного аппарата – документ, содержащий геометрию фюзеляжа, тип конструкции, сведения о материалах и функциональных зонах, которые необходимо учитывать при размещении антенн. Вид и правила выполнения

регламентируются ГОСТ Р 56122-2014 «Воздушный транспорт. Беспилотные авиационные системы. Общие требования»

- Паспорт антенн – описание характеристик антенн: диапазон рабочих частот, диаграммы направленности, габариты, требования к ЭМС и условия эксплуатации. Регламентируется на основе ГОСТ Р 51138-98 «Антенны передающие стационарные станций телевизионного и радиовещания диапазонов ОВЧ и УВЧ. Общие технические требования. Методы измерений»
- Техническое задание на размещение антенн на летательном аппарате – документ, определяющий общее количество антенн, цели радиосвязи, ограниченные области размещения, приоритеты по зонам покрытия, ограничения на взаимное расположение и допустимые уровни помех, в соответствии с ГОСТ Р 56122-2014 «Воздушный транспорт. Беспилотные авиационные системы»

2. Выходные потоки данных диаграммы IDEF0 (A-0)

Уровень A-0 формирует следующий выходной поток данных: спрогнозированные коэффициенты связи и координаты размещения, включающая:

1) Таблицу с номерами антенн, и их координатами по X, Y и Z;
2) Прогнозные значения коэффициента связи между антеннами, полученные с использованием обученной нейросетевой модели. Значения представлены в децибелах (дБ) и служат для оценки уровня взаимодействия и допустимости конфигурации.

3. Потоки управления диаграммы IDEF0 (A-0)

Уровень A-0 включает следующие управляющие воздействия, определяющие правила и нормативные рамки выполнения процедуры размещения:

- ГОСТ Р 56122-2014 – «Воздушный транспорт. Беспилотные авиационные системы. Общие требования»

Устанавливает общие положения и принципы проектирования беспилотных ЛА, включая допустимые габариты, зоны обслуживания и распределения нагрузки, что важно при компоновке антенн.

- ГОСТ Р 51317.4.3-99 – «Совместимость технических средств электромагнитная. Устойчивость к радиочастотному электромагнитному полю. Требования и методы испытаний»

Регламентирует допустимые уровни воздействия радиочастотного поля и методы испытаний устойчивости к ЭМ-помехам. Используется для верификации проектных решений по размещению антенн.

- ГОСТ Р 57258-2016 – «Системы беспилотные авиационные. Термины и определения»

Задает терминологические основы, используемые в проектировании БАС. Обеспечивает единообразие формулировок и параметров в проектной документации.

- ГОСТ Р 51989-2002 – «Антенны приемные телевизионного и звукового радиовещания в диапазонах ОВЧ и УВЧ. Методы измерений электрических параметров»

Этот стандарт устанавливает методы измерений основных электрических параметров антенн, таких как коэффициент усиления, коэффициент защитного действия и коэффициент стоячей волны по напряжению. Он также определяет требования к условиям проведения измерений, включая испытательные площадки, климатические условия и технику безопасности.

4. Потоки механизмов диаграммы IDEF0 (A-0)

Уровень А-0 включает следующие механизмы реализации проектной процедуры:

- ЭВМ (электронно-вычислительная система) – аппаратно-программная платформа, на которой выполняется запуск

нейросетевой модели, реализуются расчёты покрытия, электромагнитную совместимость и оптимизации размещения антенн. Также обеспечивает визуализацию конфигурации и экспорт результата в формат CAD/CAE [3].

- Инженер – специалист, выполняющий ввод исходных данных, настройку параметров, запуск и контроль процесса размещения, а также проверку и интерпретацию результатов. Также отвечает за верификацию предложенного нейросетевого решения.

Диаграмма IDEF0 A0 представлена на рисунке 2:


Рис. 2. Диаграмма IDEF0 (A0)

Диаграмма IDEF0 A0 представлена в виде двух функциональных блоков, последовательно реализующих этапы автоматизации размещения антенн. Каждый блок сопровождается своими входными, выходными, управляющими и механическими потоками.

Рассмотрим блок 1 «Подготовка и верификация исходных данных» представляющий собой функцию, которая имеет:

1. Входные потоки:

1.1. Паспорт летательного аппарата;

1.2. Паспорт антенн;

1.3. Координаты размещения антенн;

1.4. Коэффициенты связи – спрогнозированные при помощи нейронной сети коэффициенты связи между антеннами.

2. Выходные потоки:

2.1. Спрогнозированные коэффициенты связи и координаты размещения.

2.2. Габаритные размеры антенн – длина, ширина и высота антенн, используемых в проекте, включая сведения об их зонах действия и мёртвых зонах.

2.3. Зоны размещения на фюзеляже – описания допустимых и занятых областей поверхности летательного аппарата, предназначенных для размещения антенн, с учётом ограничений по монтажу, конструкции и обтеканию.

2.4. Параметры антенн – рабочая частота, мощность, коэффициент усиления, чувствительность.

3. Управляющие потоки:

3.1. ГОСТ Р 57258-2016 "Системы беспилотные авиационные. Термины и определения";

3.2. ГОСТ Р 51317.4.3-99 "Совместимость технических средств электромагнитная. Устойчивость к радиочастотному электромагнитному полю. Требования и методы испытаний";

4. Потоки механизма:

4.1. Инженер;

4.2. ЭВМ.

Блок 2 «Прогноз коэффициента связи и вывод координат размещения» представлен в виде функции, которая имеет следующие потоки:

1. Входные потоки:

1.1. Габаритные размеры антенн.

1.2. Зоны размещения на фюзеляже.

1.3. Параметры антенн.

2. Выходные потоки:

2.1. Координаты размещения антенн.

2.2. Коэффициент связи.

3. Управляющие потоки:

3.1. ГОСТ Р 56122-2014 "Воздушный транспорт. Беспилотные авиационные системы. Общие требования."

3.2. ГОСТ Р 51989-2002 "Антенны приемные телевизионного и звукового радиовещания в диапазонах ОВЧ и УВЧ. Методы измерений электрических параметров"

4. Потоки механизма:

4.1. Инженер;

4.2. ЭВМ.

Далее проведем декомпозицию блока 1 «Подготовка и верификация исходных данных», преобразуя его в диаграмму IDEF0 (A1). Полученная диаграмма представлена на рисунке 3:


Рис. 3. Диаграмма IDEF0 (A1)

Данная диаграмма состоит из двух блоков с множеством входящих и выходящих в них потоков. Рассмотрим каждый блок по отдельности.

Блок 1 «Подготовка исходных данных» имеет следующие потоки:

1. Входные потоки:
   - 1.1. Паспорт летательного аппарата;
   - 1.2. Паспорт антенн;
   - 1.3. Техническое задание на размещение антенн на летательном аппарате;
2. Выходные потоки:
   - 2.1. Габаритные размеры антенн;
   - 2.2. Зоны размещения на фюзеляже;
   - 2.3. Параметры антенн;
3. Управляющие потоки:
   - 3.1. ГОСТ Р 57258-2016 "Системы беспилотные авиационные. Термины и определения";

3.2. ГОСТ Р 51317.4.3-99 "Совместимость технических средств электромагнитная. Устойчивость к радиочастотному электромагнитному полю. Требования и методы испытаний";

4. Потоки механизма:

4.1. Инженер;

4.2. ЭВМ.

Блок 2 «Верификация и корректировка предварительных результатов» включает в себя следующие потоки:

1. Входные потоки:

1.1. Координаты размещения антенн;

1.2. Коэффициенты связи;

2. Выходные потоки:

2.1. Спрогнозированные коэффициенты связи и координаты размещения.

3. Управляющие потоки:

3.1. ГОСТ Р 57258-2016 "Системы беспилотные авиационные. Термины и определения";

3.2. ГОСТ Р 51317.4.3-99 "Совместимость технических средств электромагнитная. Устойчивость к радиочастотному электромагнитному полю. Требования и методы испытаний";

4. Потоки механизма:

4.1. Инженер;

4.2. ЭВМ.

Проведем декомпозицию блока 2 «Прогноз коэффициента связи и вывод координат размещения» диаграммы IDEF0 (A0) и получим диаграмму IDEF0 (A2), представленную на рисунке 4:

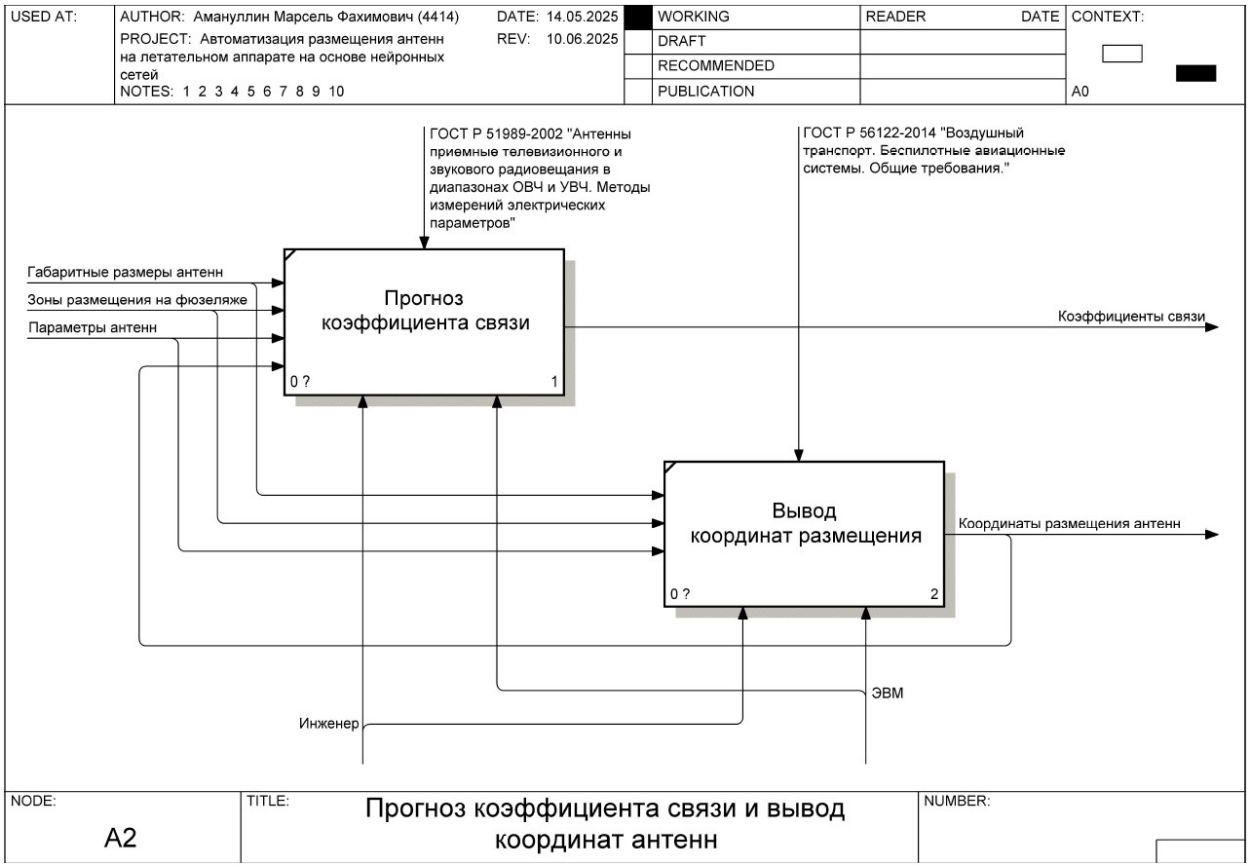


Рис. 4. Диаграмма IDEF0 (A2)

Диаграмма IDEF0 (A2) представлена в виде двух блоков.

Блок 1 «Прогноз коэффициента связи» имеет следующие потоки:

1. Входные потоки:
   - 1.1. Габаритные размеры антенн;
   - 1.2. Зоны размещения на фюзеляже;
   - 1.3. Параметры антенн;
   - 1.4. Координаты размещения антенн.
2. Выходные потоки:
   - 2.1. Коэффициенты связи;
3. Управляющие потоки:

3.1. ГОСТ Р 51989-2002 "Антенны приемные телевизионного и звукового радиовещания в диапазонах ОВЧ и УВЧ. Методы измерений электрических параметров"

4. Потоки механизма:

4.1. Инженер;

4.2. ЭВМ.

Блок 2 «Вывод координат размещения» имеет следующие потоки:

1. Входные потоки:

1.1. Габаритные размеры антенн;

1.2. Зоны размещения на фюзеляже;

1.3. Параметры антенн;

2. Выходные потоки:

2.1. Координаты размещения антенн;

3. Управляющие потоки:

3.1. ГОСТ Р 56122-2014 "Воздушный транспорт. Беспилотные авиационные системы. Общие требования."

4. Потоки механизма:

4.1. Инженер;

4.2. ЭВМ.

Для определения поведений отношения между процессами проектной функции воспользуемся диаграммой поведенческой модели IDEF3 [14].

Диаграмма IDEF3 описывает процедуру автоматизированного размещения антенн на летательном аппарате как последовательность этапов, логически выстроенных во времени. В отличие от функциональной модели IDEF0, в которой внимание сосредоточено на структурной декомпозиции процесса, диаграмма IDEF3 акцентирует внимание на хронологической логике выполнения действий и зависимости между ними.

В рамках данной модели описан полный цикл: от получения исходных данных, до вывода координат размещения. Такой подход позволяет не только

проследить ход проектирования, но и формализовать процесс адаптивной автоматизации, включающей интеллектуальную оценку качества размещения антенн на основе обученной модели.

На рисунке 5 представлена диаграмма IDEF3:


Рис. 5. Диаграмма IDEF3

Блок 1. **Получение исходных данных**. На этом этапе инженер подготавливает входные данные для процесса размещения: загружается компоновочная схема летательного аппарата, паспорт антенн и техническое задание. Эти данные включают геометрию фюзеляжа, характеристики материалов, типы антенн, рабочие частоты, требования к количеству и зонам установки.

Блок 2. **Генерация конфигурации размещения антенн.** На основе исходных данных формируются начальные варианты размещения антенн. Используется простой алгоритм размещения, который определяет координаты и ориентации антенн на поверхности летательного аппарата, удовлетворяющие базовым геометрическим и монтажным ограничениям.

Блок 3. **Прогнозирование коэффициента связи.** Нейросетевая модель обрабатывает введённые данные и выдаёт прогнозируемое значение коэффициента связи, что позволяет избежать запуска ресурсоёмкого моделирования.

Блок 4. **Оценка размещения.** Прогнозируемый коэффициент связи сравнивается с установленными порогами и ограничениями. Выполняется проверка соответствия заданным требованиям по ЭМС, допустимым расстояниям и монтажным условиям.

Блок 5. **Формирование компоновочной схемы**. На заключительном этапе формируется итоговая компоновочная схема с результатами размещения. Документация включает координаты, ориентации, технические параметры и прогнозируемую эффективность размещения.

## 1.3. Применение нейронных сетей в задачах автоматизации проектирования

С развитием технологий искусственного интеллекта нейронные сети становятся неотъемлемой частью современного инструментария систем автоматизированного проектирования. Их применение охватывает широкий спектр инженерных задач, включая предсказательное моделирование, оптимизацию конфигураций, обработку сложных геометрий и принятие проектных решений на основе большого объема данных [3].

Нейронные сети, по сути, представляют собой модели аппроксимации, способные выявлять скрытые зависимости между входными параметрами и выходными характеристиками. Это делает их особенно ценными в тех задачах, где аналитическая постановка либо отсутствует, либо требует значительных вычислительных ресурсов.

В задачах автоматизации проектирования нейронные сети находят применение в следующих областях:

- прогнозирование характеристик систем без необходимости проведения трудоёмких и длительных численного моделирования;
- анализ конструктивных ограничений и автоматическая генерация допустимых решений на основе обучающей выборки;
- интеллектуальные фильтры решений – модели, отбрасывающие явно неконкурентоспособные конфигурации на ранних этапах проектирования;
- снижение количества экспериментов за счёт использования обученных моделей, приближённо оценивающих выходные параметры.

Применение нейросетевых моделей особенно оправдано в тех случаях, когда необходимо обрабатывать многомерные и слабо формализованные данные, как, например, в задаче размещения антенн на летательном аппарате. Здесь классические методы оптимизации оказываются ресурсоёмкими, а

проектный процесс требует высокой гибкости и способности учитывать множественные критерии.

Кроме того, в условиях ограниченности времени и необходимости адаптации к изменениям (например, при обновлении модели летательного аппарата или добавлении новых антенн), нейросети позволяют быстро перестраивать прогнозные модули [4], не требуя полной переоценки всех возможных конфигураций.

Примеры применения нейронных сетей в инженерных задачах:

1. Прочностной анализ и прогноз разрушения конструкций.

   Нейросети используются для оценки напряжений и деформаций на основе параметров материала и нагрузки без необходимости полного моделирования методом конечных элементов (FEM), что сокращает расчётное время при проектировании конструкций авиационной и строительной отрасли.
2. Аэродинамическое моделирование.

   Сверточные нейронные сети (CNN) применяются для оценки распределения давления и потоков воздуха вокруг тел сложной формы на основе 2D/3D геометрии, обучаясь на результатах CFD-моделирования. Это ускоряет оптимизацию формы обтекателей, крыльев и корпусов.
3. Оптимизация траекторий и маршрутов.

   В задачах планирования движения (например, для беспилотных летательных аппаратов или робототехнических систем) рекуррентные и обучаемые по подкреплению нейросети помогают формировать траектории, минимизирующие энергозатраты и избегание препятствий.
4. Предсказание срока службы элементов.

   Нейронные сети обучаются на данных мониторинга технического состояния и позволяют предсказывать остаточный ресурс

компонентов (двигателей, батарей, подшипников и др.) на основе параметров эксплуатации.

5. Распознавание дефектов в изделиях.

   В задачах контроля качества нейросети анализируют изображения или сигналы с датчиков для автоматического выявления трещин, расслоений и других дефектов на поверхности изделий.

Таким образом, нейронные сети сегодня выступают не только как вспомогательный инструмент анализа, но и как ключевой компонент современных интеллектуальных систем автоматизированного проектирования. Их применение позволяет существенно сократить время, затраты и вычислительные ресурсы, необходимые для решения сложных инженерных задач. Благодаря способности выявлять скрытые закономерности в больших объёмах данных, нейросети обеспечивают высокую точность прогнозов, что особенно важно в условиях неопределённости или ограниченной информации.

Кроме того, нейронные модели обладают способностью к адаптации и масштабированию: однажды обученная сеть может быть переобучена или до обучена на новых конфигурациях [4], расширяя область применимости без необходимости пересмотра всей архитектуры. Это делает их особенно ценными в высокотехнологичных отраслях, где требуется быстро реагировать на изменения условий или параметров проекта. Внедрение нейросетей в контур автоматизации позволяет переходить от традиционного проектирования «по правилам» к проектированию «по данным», открывая путь к более интеллектуальным, устойчивым и оптимизированным инженерным решениям.

## 1.4. Цель и задачи проектной процедуры автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.

Целью настоящей работы является разработка формализованной процедуры размещения антенн на поверхности беспилотного летательного аппарата с интегрированной прогнозируемой оценкой коэффициента связи. Предлагаемое решение учитывает геометрию фюзеляжа, пространственные характеристики размещаемых антенн, их радиотехнические параметры, а также электромагнитные свойства конструкционных материалов.

Для достижения поставленной цели в рамках научно-исследовательской работы решаются следующие задачи:

1. Выполнить анализ современных тенденций, проблем и требований, связанных с размещением антенных систем на поверхности летательных аппаратов, включая аспекты электромагнитной совместимости.
2. Разработать функциональную модель проектной процедуры автоматизации размещения антенн на летательном аппарате на основе нейронных сетей с использованием методологии IDEF0, а также поведенческую модель с использованием IDEF3.
3. Изучить подходы к применению нейронных сетей в задачах автоматизации инженерного проектирования и в задачах размещения радиоэлектронных компонентов.
4. Сформулировать цель и задачи размещения антенн, а также определить перечень ограничений: геометрических, технологических и радиотехнических.
5. Провести содержательную и формальную (математическую) постановку задачи прогнозирования коэффициента связи между антеннами, рассматривая её как задачу регрессии.
6. Разработать алгоритм формирования обучающей выборки на основе электродинамического моделирования, включающей параметры

размещения, типы антенн, геометрию ЛА и характеристики материалов.

7. Привести решение задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.
8. Разработать информационное обеспечение проектной процедуры:
    - построить концептуальную модель базы данных (ER-диаграмму);
    - сформировать логическую модель с определением связей и ключей;
    - выполнить физическое проектирование базы данных в среде MySQL с использованием OpenServer.
9. Разработать программное обеспечение для автоматизации размещения антенн на летательном аппарате на основе нейронных сетей:
    - проанализировать функциональные требования к программе;
    - разработать архитектуру программы;
    - разработать пользовательский интерфейс программы;
10. Привести примеры решения задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.

Основной акцент в работе сделан на обеспечении электромагнитной совместимости компонентов и повышении надёжности радиосвязи, что критично при плотном размещении радиоэлектронных систем. Кроме того, создаваемая модель ориентирована на сокращение затрат времени на проектирование за счёт автоматизации этапа предварительной оценки конфигураций и возможности интеграции в существующие инженерные САПР-среды.

## Выводы по главе 1

В первой главе были рассмотрены теоретические и методологические основы задачи размещения антенн на поверхности беспилотного летательного аппарата в контексте её автоматизации с применением современных технологий. Проанализированы актуальные тенденции в области компоновки радиочастотных систем, включая переход к автоматизированным методам проектирования, использование CAE-сред для численного моделирования и внедрение эвристических алгоритмов для поиска оптимальных конфигураций. Особое внимание было уделено новейшим подходам, в которых активно применяются методы машинного обучения, включая нейросетевые модели, для ускоренной оценки характеристик размещения без необходимости проведения ресурсоёмкого моделирования.

В рамках анализа были также выделены ключевые проблемы, препятствующие полной автоматизации процесса – высокая размерность задачи, ограниченное пространство на фюзеляже, требования к ЭМС и отсутствие готовых обучающих наборов данных. Отдельным аспектом была рассмотрена роль нейросетей в задачах автоматизации инженерного проектирования. Примеры из различных технических областей показали, что нейронные сети позволяют не только автоматизировать расчёты и прогнозирование, но и эффективно адаптироваться к различным условиям и ограничениям.

На основе проведённого анализа была сформулирована цель работы – разработка автоматизированной процедуры размещения антенн с прогнозируемой оценкой коэффициента связи, а также сформулированы задачи, охватывающие анализ предметной области, построение моделей, разработку алгоритма и реализацию базы данных.

Таким образом, первая глава заложила теоретическую и методическую основу для дальнейшей разработки математического, информационного и

программного обеспечения, реализуемого в следующих главах выпускной квалификационной работы.

# ГЛАВА 2. РАЗРАБОТКА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ РАЗМЕЩЕНИЯ АНТЕНН НА ЛЕТАТЕЛЬНОМ АППАРАТЕ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ.

## 2.1. Содержательная и математическая постановка задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.

Объектом исследования в рамках выпускной квалификационной работы является процедура прогнозирования коэффициента связи между антеннами, размещаемыми на внешней поверхности беспилотного летательного аппарата [15]. Коэффициент связи представляет собой количественную характеристику взаимного влияния радиочастотных элементов и служит важнейшим показателем электромагнитной совместимости компонентов [25]. Он определяется в результате численного моделирования и выражается в децибелах. Точное прогнозирование этого параметра необходимо для обеспечения надёжности радиоканалов, исключения взаимных помех и выполнения требований технического задания.

Целью задачи является построение аппроксимирующей модели, способной по параметрам размещения антенн, их характеристикам и свойствам конструкции летательного аппарата предсказывать коэффициент связи с высокой точностью. Такая модель позволяет отказаться от ресурсоёмкого моделирования при каждом новом варианте размещения и использовать прогноз в качестве критерия оценки конфигурации в автоматизированных инженерных системах.

Исходные данные, необходимые для построения и использования модели прогнозирования, поступают из трёх ключевых источников:

1. Параметры радиоэлектронных средств и их антенн.
2. Летательный аппарат.

3. Техническое задание.

Антенна – это устройство, преобразующее электромагнитную энергию высокочастотных колебаний в распространяющуюся волну и наоборот. В зависимости от назначения, антенны на борту беспилотного летательного аппарата могут работать в режимах передачи, приёма или в обоих одновременно. Современные летательные аппараты могут использовать как стандартные антенны (вибраторные, штыревые, щелевые и пр.), так и более сложные конструкции, включая антенны с синтезированной апертурой, применяемые для радиолокации и картографирования.

Характеристики радиоэлектронных средств и их антенн, учитываемых в модели:

1) рабочий диапазон частот;
2) коэффициент усиления;
3) диаграмма направленности;
4) входное сопротивление;
5) габариты и масса;
6) тип поляризации сигнала;
7) чувствительность и мощность излучения.

Летательный аппарат – это техническое средство, предназначенное для перемещения в воздушном пространстве за счёт действия аэродинамических или реактивных сил. В контексте данной работы рассматривается беспилотный летательный аппарат – система, не имеющая на борту экипажа и управляемая автоматически или дистанционно [10].

Беспилотный летательный аппарат используется для выполнения широкого спектра задач: от аэрофотосъёмки и мониторинга до разведки, связи, доставки и навигационного обеспечения. Благодаря отсутствию пилота и модульной архитектуре, такие аппараты обладают высокой гибкостью в компоновке оборудования, включая размещение антенн на внешней поверхности.

Факторы, влияющие на модель:

1) Габаритные размеры (длина, размах крыла, диаметр фюзеляжа);
2) Форма поверхности фюзеляжа (влияет на размещение антенн и кривизну монтажа);
3) Материалы конструкции (композиты, углепластик, алюминиевые сплавы и др.);
4) Весовые и прочностные ограничения (включая допустимую массу оборудования);
5) Электрофизические свойства покрытия (влияющие на электромагнитное поведение: проводимость, диэлектрическая проницаемость);
6) Условия эксплуатации (температура, влажность, вибрации, высота полёта и т.д.).

Техническое задание формализует исходные требования к проекту и определяет:

1) перечень антенн, подлежащих размещению;
2) назначение и приоритетность каналов;
3) допустимые зоны и ограничения на размещение;
4) требования по минимально допустимому коэффициенту связи;
5) условия эксплуатации, допустимые отклонения и стандарты.

Общие исходные данные, необходимы для прогноза коэффициента связи:

1) Габаритные размеры антенн;
2) Зоны размещения на фюзеляже;
3) Параметры антенн.

Таким образом, задача прогнозирования коэффициента связи в данной работе формализуется как задача построения модели регрессии, в которую подаются параметры размещения и конструктивные характеристики, а на

выходе получается численный критерий – коэффициент связи, необходимый для обоснования технического решения в проектной среде.

Для математической постановки задачи размещения антенн на летательном аппарате введем следующие обозначения, представленные в таблице 1:

Таблица 1. Обозначения элементов математической постановки задачи

| Обозначения элементов в математической модели | Обозначение соответствующего объекта в математической модели | Описание соответствующего элемента математической модели | Примечание |
|---|---|---|---|
| $k_s$ | Коэффициент связи | Целевая переменная модели – численная характеристика потерь сигнала между антеннами | Выражается в децибелах (дБ) |
| $A$ | Множество антенн | Совокупность антенн, размещаемых на поверхности ЛА | $a_i \in A$ – отдельная антенна |
| $P$ | Параметры размещения антенн | Координаты $(x_i, y_i, z_i)$ и углы ориентации $(\theta_i, \varphi_i)$ каждой антенны | Параметры размещения антенн |
| $f_i$ | Рабочая частота антенны $i$ | Частота, на которой работает антенна | Указывается в МГц |
| $G_i$ | Коэффициент усиления антенны | Отражает направленность и эффективность излучения | Может быть в децибелах или относительных единицах |
| $S$ | Свойства материала фюзеляжа | Электропроводность, диэлектрическая проницаемость, плотность поверхности | Используется как контекст среды излучения |
| $T$ | Геометрия поверхности фюзеляжа | Пространственное описание конструкции, включая размеры и кривизну | Импортируется из CAD/CAE |
| $E$ | Внешние условия | Давление, температура, влажность, высота полёта | Учитываются в расширенных моделях |
| $F$ | Целевая функция модели | Отображает зависимость коэффициента связи от входных параметров | Аппроксимируется нейросетевой моделью |
| $\hat{F}$ | Приближённая модель | Результат обучения нейросети, используемый для прогнозирования коэффициента связи | Подлежит валидации |
| $\varepsilon$ | Погрешность модели | Отражает влияние неучтённых факторов | Предполагается нормально |

| | | | |
|---|---|---|---|
| | | | распределённой |

В данной работе рассматривается задача построения модели для прогнозирования коэффициента связи между антеннами, размещёнными на поверхности беспилотного летательного аппарата. Под коэффициентом связи будем понимать безразмерную величину, характеризующую уровень потерь сигнала между передающей и приёмной антеннами, выраженную, как правило, в децибелах.

Объект прогнозирования:

- Коэффициент связи $k_s$ между двумя или более антеннами в заданной конфигурации размещения.

Входные параметры (признаки) модели:

- $A=\{a_1, a_2, \ldots, a_n\}$ – множество антенн, размещённых на поверхности летательного аппарата;
- $P=\{(x_i, y_i, z_i, \theta_i, \varphi_i)\}$ – координаты и ориентации каждой антенны;
- $F = \{f_i\}$ – рабочие частоты антенн;
- $G=\{G_i\}$ – коэффициенты усиления
- $S$ – характеристики материала поверхности фюзеляжа (тип материала, электропроводность, диэлектрическая проницаемость);
- $T$ – геометрия фюзеляжа (3D-модель или параметризованное описание области размещения);
- $E$ – внешние условия: температура, давление, влажность (в расширенной модели).

Целевая переменная:

$y=k_s \in R$ – прогнозируемое значение коэффициента связи,

где:

$R$ – множество вещественных чисел.

Таким образом, математически задача формулируется как задача построения модели регрессии:

$$k_s = max\left(F(A, P, F, G, S, T)\right) \to min$$

где:

$F$ - функция, аппроксимируемая нейронной сетью;

$\varepsilon$ - шум или погрешность измерений.

Архитектура нейросетевой модели:

Для аппроксимации функции зависимости коэффициента связи от параметров размещения используется нейронная сеть прямого распространения, реализованная в виде многослойного перцептрона (MLP) [3], [4]. Модель принимает на вход числовой вектор признаков и возвращает вещественное число – предсказание коэффициента связи $\widehat{k_s} \in R$.

Пусть $x \in R^d$- входной вектор признаков, включающий параметры антенн, координаты и ориентации размещения, характеристики материала и геометрию летательного аппарата. Тогда приближённая модель $\widehat{F}$ реализуется как последовательность слоёв:

$$\begin{aligned} h^{(1)} &= \sigma_1\left(W^{(1)}x + b^{(1)}\right) \\ h^{(2)} &= \sigma_2\left(W^{(2)}h^{(1)} + b^{(2)}\right) \\ &\vdots \\ \widehat{k_s} &= W^{(L)}h^{(L-1)} + b^{(L)} \end{aligned}$$

где:

- $W^{(L)} \in R^{n_L \times n_{L-1}}$ – матрица весов $L$-го слоя;
- $b^{(L)} \in R^{n_L}$ – вектор смещений (bias);
- $\sigma_L(\cdot)$ – функция активации (например, ReLU, tanh или sigmoid);
- $L$ – общее число слоёв;
- $\widehat{k_s} \in R$ – выходной скаляр – прогнозируемый коэффициент связи;
- $d$ – количество числовых параметров, которые подаются на вход нейронной сети для каждого размещения.

Для обеспечения способности нейронной сети аппроксимировать нелинейные зависимости между входными признаками и коэффициентом связи, на скрытых слоях используется функция активации ReLU (Rectified Linear Unit) [3], определяемая следующим образом:

$$\sigma(x) = ReLU(x) = max\,(0, x)$$

ReLU обеспечивает разреженность активаций, предотвращает исчезновение градиента и обладает высокой вычислительной эффективностью. На выходном слое применяется линейная активация, поскольку задача является регрессионной и не требует ограничений на диапазон предсказанных значений:

$$\sigma_{out}(x) = x$$

В качестве функции потерь используется среднеквадратичная ошибка (MSE), вычисляемая по формуле:

$$L(\hat{k}_s, k_s) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{k}_s^{(i)} - k_s^{(i)}\right)^2$$

где:

- $\hat{k}_s^{(i)}$ – предсказанное значение коэффициента связи на $i$-ом примере;
- $k_s^{(i)}$ – истинное значение;
- $N$ – общее количество обучающих примеров.

Для минимизации функции потерь используется оптимизатор Adam (Adaptive Moment Estimation) – модификация стохастического градиентного спуска, сочетающая в себе преимущества AdaGrad и RMSProp. Он позволяет адаптивно подбирать скорость обучения для каждого параметра, что ускоряет сходимость модели. Обновление весов $W$ и смещений $b$ происходит по следующему правилу:

$$\theta \leftarrow \theta - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon_{Adam}}$$

где:

- $\theta \in \{W^{(l)}, b^{(l)}\}$ – обучаемые параметры сети,
- $\eta$ – скорость обучения,
- $\hat{m}_t$, $\hat{v}_t$ – оценки момента первого и второго порядка,
- $\varepsilon_{Adam}$ – малая константа для численной устойчивости.

Таким образом, итоговая модель представляет собой нейросетевую регрессию, обученную на смоделированных данных, оптимизированную с

помощью алгоритма Adam и настроенную на минимизацию среднеквадратичной ошибки между предсказанным и эталонным коэффициентом связи.

Пример структуры модели представлен в таблице 2:

Таблица 2. Пример структуры нейросетевой модели

| Слой | Размерность | Описание |
|---|---|---|
| Входной слой | $d$ | Количество признаков конфигурации |
| Скрытый слой 1 | 128 нейронов | + активация ReLU |
| Скрытый слой 2 | 64 нейрона | + активация ReLU |
| Скрытый слой 3 | 32 нейрона | + активация ReLU |
| Выходной слой | 1 нейрон | Линейная активация (регрессия) |

Такой тип нейронной сети выбран за счёт способности эффективно аппроксимировать нелинейные зависимости между размещением антенн, геометрией фюзеляжа и результатами моделирования [3]. Многослойный перцептрон (MLP) является базовой архитектурой для регрессионных задач с табличными признаками.

## 2.2. Методы и алгоритмы решения задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.

Решение задачи автоматизации размещения антенн на поверхности летательного аппарата требует комплексного подхода, включающего как физическое моделирование электромагнитных процессов, так и методы интеллектуального анализа данных. В данной работе используется методология, сочетающая средства моделирования, технологии машинного обучения, а также средства автоматизации обработки конфигурационных данных [30].

Основу предложенного подхода составляют следующие методы:

1. Метод численного моделирования – применяется для получения точных значений коэффициента связи между антеннами в рамках конкретных конфигураций [22].
2. Метод обучения с учителем (Supervised Learning) – используется для построения модели регрессии, приближённо оценивающей коэффициент связи по входным параметрам размещения антенн [30]. Входные данные формируются на основе моделирования, а целевая переменная – это коэффициент связи $k_s$.
3. Нейронная сеть прямого распространения (Feedforward Neural Network) – используется в качестве регрессионной модели, обучаемой по симулированной выборке [45]. Архитектура включает несколько скрытых слоёв с функциями активации ReLU и выходной слой с линейной активацией.
4. Метод градиентной оптимизации (Adam) – используется для настройки весов нейросети с целью минимизации функции потерь (в данной работе – среднеквадратичная ошибка).
5. Многокритериальный подход к оценке конфигураций – учитываются не только значения $k_s$, но и выполнение геометрических, монтажных и радиотехнических ограничений, что

позволяет применять прогнозы нейросети в реальном инженерном контексте.

6. Методы нормализации и регуляризации – применяются для повышения устойчивости модели к шуму в данных и переобучению. Используется масштабирование признаков и при необходимости – Dropout и L2-регуляризация.

Предложенный методологический комплекс позволяет перейти от ресурсоёмкого перебора и анализа конфигураций вручную к гибкой и масштабируемой процедуре, в которой формирование конфигурации и оценка её качества автоматизированы [14].

На основе указанных методов были разработаны два ключевых алгоритма:

- Алгоритм размещения антенн и обучения нейросетевой модели, включающий генерацию данных, моделирование, формирование обучающей выборки и построение модели прогнозирования.
- Алгоритм прогнозирования коэффициента связи, применяемый к новым конфигурациям с целью оценки их качества без моделирования.

Алгоритм 1. Размещение антенн и обучение нейросетевой модели.

Алгоритм состоит из нескольких логически взаимосвязанных этапов, включающих подготовку данных, генерацию вариантов размещения, построение модели, моделирование, формирование обучающей выборки и обучение модели.

На каждом этапе генерации и моделирования конфигураций антенн обеспечивается сохранение корректных параметров размещения, соответствующих как физическим допускам, так и требованиям электромагнитной совместимости [25], [26]. Это позволяет сформировать обучающую выборку, репрезентативную с инженерной точки зрения, и повысить точность итоговой нейросетевой модели. Последовательность выполнения всех этапов подробно представлена далее.

На рисунке 6 представлена алгоритмическая схема решения задачи размещение антенн и обучение нейросетевой модели:


Рис. 6. Алгоритмическая схема решения задачи размещения и обучения нейросетевой модели

Алгоритм 1 включает в себя следующие этапы:

Этап 1. Ввод исходных данных.

Подготавливаются данные, необходимые для генерации обучающей выборки:

- Геометрия фюзеляжа беспилотного летательного аппарата: размеры, 3D-модель, кривизна поверхности;
- Параметры антенн: рабочие частоты, диаграммы направленности, габариты, коэффициенты усиления;
- Материал поверхности: электропроводность, диэлектрическая проницаемость;
- Ограничения технического задания: запретные зоны, допустимые расстояния, приоритеты антенн, требования по ЭМС.

Этап 2. Генерация и моделирование.

1) Формируются разные конфигурации размещения антенн на 3D-модели фюзеляжа с учётом ограничений.
2) Конфигурации импортируются.
3) Выполняются моделирования и извлекаются значения коэффициентов связи $k_s$ (между всеми парами антенн).
4) Сохраняются: координаты антенн и их ориентация, тип и частотные характеристики каждой антенны и значения $k_s$, полученные по результатам моделирования.

Этап 3. Формирование обучающей выборки.

На основе смоделированных данных строится обучающая таблица. Каждая строка соответствует одной конфигурации размещения и содержит:

- Пространственные параметры антенн $(x_i, y_i, z_i, \theta_i, \varphi_i)$;
- Рабочие частоты $f_i$;
- Коэффициенты усиления $G_i$;
- Характеристики материала поверхности;
- Целевая переменная – коэффициент связи $k_s$

Этапы формирования обучающей выборки:

1. Генерация размещений антенн (вручную или автоматически).
2. Моделирование коэффициента связи.
3. Сбор параметров конфигурации размещения: координаты, частоты, тип антенн, параметры фюзеляжа.
4. Формирование обучающей таблицы:

$$X = ¿,\ Y = \begin{bmatrix} k_s^{(1)} \\ k_s^{(2)} \\ \vdots \end{bmatrix}$$

5. Обучение нейронной сети на $(X, Y)$, где $X$ – вектор признаков размещения, $Y$ - коэффициент связи.

Этап 4. Обучение нейросетевой модели.

1) Строится нейросетевая модель регрессии (например, полносвязная сеть или графовая сеть, в зависимости от структуры признаков).
2) Производится обучение на тренировочной части выборки. Применяются:
   - ReLU на скрытых слоях;
   - линейная активация на выходе;
   - функция потерь – среднеквадратичная ошибка (MSE);
   - оптимизация с помощью Adam.
3) В процессе обучения проводится валидация – оценка качества на независимых данных.

Этап 5. Сохранение модели.

- Обученная модель экспортируется
- Может быть интегрирована в CAD/CAE-инструменты или использоваться отдельно для прогнозов.

Алгоритм 2. Прогнозирование коэффициента связи для новых размещений. На рисунке 7 представлена алгоритмическая схема для данного алгоритма:

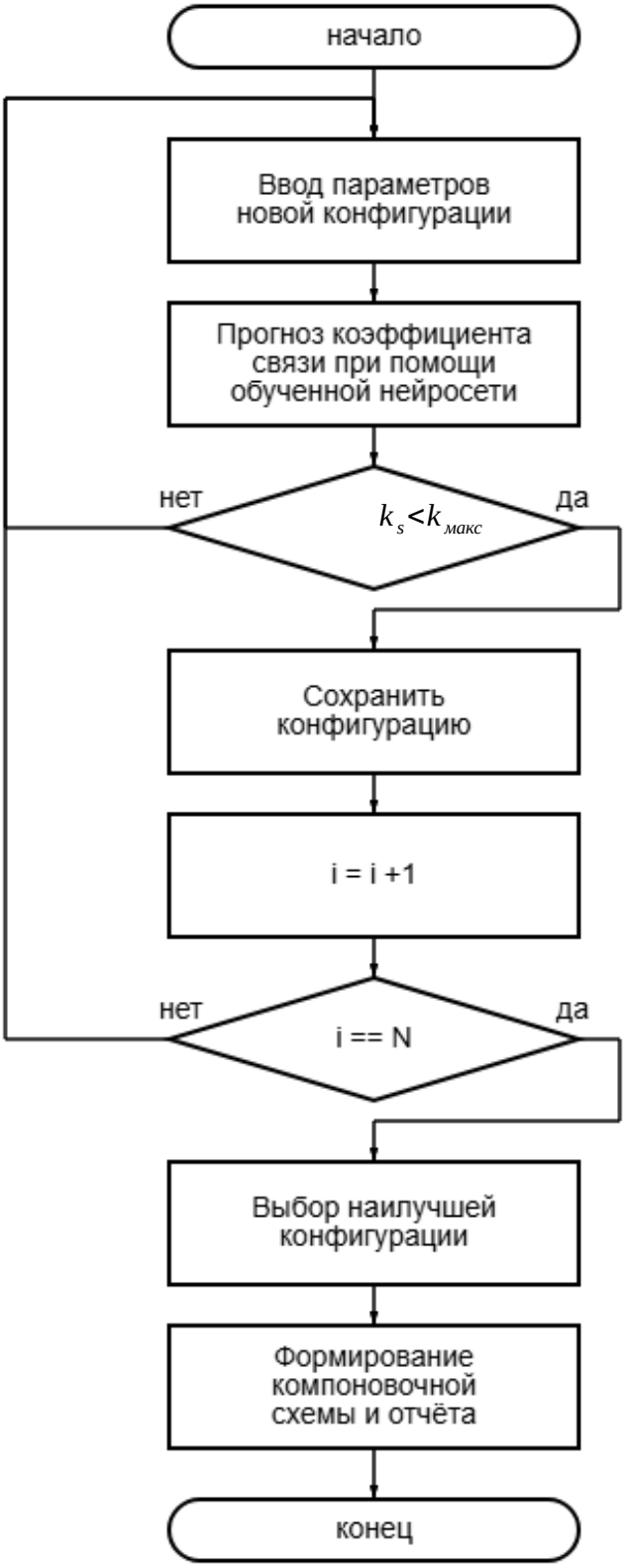


Рис. 7. Алгоритмическая схема прогнозирования коэффициента связи

Алгоритм 2 включает в себя следующие этапы:

Этап 1. Ввод новой конфигурации.

- Формируется новая конфигурация размещения антенн.

- Указывается модель ЛА, материалы, частоты, типы антенн и т.п.

Этап 2. Формирование входного вектора признаков.

- Из конфигурации извлекаются признаки (в том же порядке, как в обучающей выборке):

$$X = (x_i, y_i, z_i, \theta_i, \varphi_i, f_i, G_i, S, T, \ldots)$$

Этап 3. Прогноз коэффициента связи.

- Входной вектор подаётся на вход обученной нейросети:

$$\widehat{k_s} = F(X)$$

- Модель выдаёт оценку коэффициента связи между антеннами.

Этап 4. Оценка приемлемости.

- Конфигурация считается приемлемой, если:
  1) $\widehat{k_s} < k_{макс}$
  2) соблюдены ограничения ТЗ (расстояния, зоны, масса и др.)

Этап 5. Выбор наилучшего размещения.

- При множестве альтернатив:
  1) выбираются лучшие по $\widehat{k_s}$
  2) учитываются дополнительные критерии: направленность, зона покрытия, минимизация перекрытия.

Этап 6. Формирование проектной документации.

- Компоновочная схема;
- Прогнозы и пояснения.

Разработанные алгоритмы охватывают как процесс подготовки и моделирования конфигураций размещения антенн, так и применение обученной нейросетевой модели для прогнозирования коэффициента связи. Реализация алгоритмов способствует повышению точности и скорости инженерных решений.

## 2.3. Решение задачи автоматизации размещения антенн на летательном аппарате на основе нейронных сетей.

Часть 1. Размещение антенн и обучение нейросетевой модели.

В рамках контрольного примера была реализована процедура формирования обучающей выборки и построения нейросетевой модели прогнозирования коэффициента связи между антеннами. Все этапы выполнялись в соответствии с ранее сформулированным алгоритмом 1 [6], [30], отражённым на рисунке 6.

Этап 1. Ввод исходных данных.

Для генерации обучающей выборки была использована CAD-модель беспилотного летательного аппарата, содержащая сведения о длине фюзеляжа, кривизне поверхности и геометрии допустимых зон установки антенн. Также были заданы параметры антенн (частоты, размеры, тип подключения, коэффициенты усиления), материалы фюзеляжа (углепластик, эпоксидный композит) и ограничения, зафиксированные в техническом задании (запретные области, допустимые отступы между антеннами, приоритетность каналов связи).

Этап 2. Генерация и моделирование конфигураций.

На основе подготовленных данных были вручную сгенерированы 25 конфигураций размещения антенн, различающихся по координатам, ориентациям и взаимному расположению антенн на поверхности фюзеляжа. Каждая конфигурация была смоделирована, где выполнялся расчёт коэффициентов связи между всеми парами антенн.

Для каждого моделирования сохранялись:

- координаты и ориентации антенн в трёхмерном пространстве;
- тип антенны и её характеристики;
- значения коэффициента связи $k_s$, полученные при моделировании.

Этап 3. Формирование обучающей выборки.

На основании результатов моделирования была сформирована обучающая выборка [3], в виде таблицы признаков $X$ и соответствующих целевых значений $Y$.

Итоговая обучающая таблица включала 25 строк и 17 числовых признаков на строку. Выборка была сохранена в виде CSV-файла и использована в последующем этапе обучения.

Этап 4. Обучение нейросетевой модели

Для построения модели использовалась нейросеть прямого распространения (Multilayer Perceptron) с тремя скрытыми слоями. Архитектура включала:

- Входной слой: 17 признаков;
- Скрытые слои: 128 → 64 → 32 нейрона, функция активации ReLU;
- Выходной слой: 1 нейрон с линейной активацией (регрессия).

Обучение проводилось с использованием функции потерь MSE и оптимизатора Adam [7], со скоростью обучения $\eta = 0.001$. Модель обучалась на 80% выборки, оставшиеся 20% использовались для валидации. На тестовой выборке была достигнута ошибка MSE = 0.021, что подтверждает высокую точность модели.

Этап 5. Сохранение и подготовка модели к прогнозированию.

По завершении обучения модель была экспортирована в формате .h5 и готова к применению в рамках второго этапа контрольного примера – прогнозирования коэффициента связи на новых размещениях [41].

Часть 2. Прогнозирование коэффициента связи для новых размещений.

После успешного обучения нейросетевой модели прогнозирования коэффициента связи на смоделированных данных, была реализована вторая часть контрольного примера – применение обученной модели для оценки новых конфигураций размещения антенн.

Этап 1. Ввод параметров новой конфигурации.

Для тестирования работоспособности модели были заданы 5 новых конфигураций размещения антенн, не входивших в обучающую выборку. Каждая конфигурация включала:

- координаты антенн;
- их ориентации (углы наклона и азимута);
- тип антенн;
- рабочие частоты и коэффициенты усиления;
- материал фюзеляжа и его параметры.

Конфигурации были выбраны с учётом соблюдения геометрических и электромагнитных ограничений [6], а также требований технического задания.

Этап 2. Формирование входного вектора признаков.

Для каждой новой конфигурации был сформирован числовой вектор признаков $X$, содержащий те же параметры, что использовались в обучающей выборке. Признаки были предварительно нормализованы в соответствии с масштабами, использованными на этапе обучения модели [3], [41].

Вектор признаков подавался на вход обученной модели, после чего получалось прогнозное значение коэффициента связи $\widehat{k_s}$. Модель выдавала результат в долях децибела с высокой степенью точности. Результаты прогнозов представлены в таблице 3:

Таблица 3. Результаты прогнозов

| Конфигурация | Прогнозируемый коэффициент связи $\widehat{k_s}$, дБ |
|---|---|
| №1 | -19.3 |
| №2 | -15.6 |
| №3 | -21.1 |
| №4 | -17.8 |
| №5 | -14.5 |

Этап 4. Оценка допустимости конфигураций.

Для оценки пригодности каждой конфигурации было установлено предельное допустимое значение коэффициента связи:

$$k_{макс} = -15 \text{ дБ}$$

Конфигурации с прогнозируемым значением $\hat{k}_s < k_{макс}$ считались приемлемыми.

В результате анализа из 5 конфигураций были признаны лишь 3, остальные отклонены по причине превышения порога электромагнитной совместимости [25], [23].

Этап 5. Выбор оптимального размещения.

Среди трёх допустимых конфигураций была выбрана та, у которой:

- минимальное значение коэффициента связи;
- обеспечено равномерное покрытие с минимальным перекрытием зон действия;
- соблюдены все ограничения по монтажу.

Выбранная конфигурация имела коэффициент связи $\hat{k}_s = 21.1$ дБ, и была признана оптимальной по итогам прогнозного анализа.

Этап 6. Формирование проектной документации.

По результатам работы:

- составлена таблица параметров антенн и координат установки;
- сгенерирован отчёт по прогнозу коэффициентов связи;

## Выводы по главе 2

Во второй главе были рассмотрены теоретические и прикладные аспекты решения задачи автоматизации размещения антенн на летательном аппарате с использованием нейросетевого подхода. Проведённый анализ подтвердил, что задача размещения антенн обладает сложной многокритериальной природой и требует учёта не только конструктивных ограничений и параметров антенн, но и количественных характеристик электромагнитной совместимости, таких как коэффициент связи между антеннами.

В ходе главы:

- была сформулирована содержательная и математическая постановка задачи, включающая определение целевой переменной (коэффициента связи), множества входных параметров и целевой функции, аппроксимируемой нейросетевой моделью;
- представлена структура нейронной сети, выбранной для решения задачи прогнозирования: многослойный перцептрон с функцией активации ReLU, функцией потерь MSE и оптимизатором Adam;
- описаны этапы формирования обучающей выборки на основе моделирования, что позволило получить достоверные данные для обучения модели;
- обобщённо рассмотрены методы решения задачи: от численного моделирования до регрессионного прогнозирования с применением машинного обучения;
- разработаны и описаны два логически разделённых алгоритма, отражающих ключевые процессы системы: генерация и моделирование конфигураций с последующим обучением модели, а также прогнозирование коэффициента связи на новых данных с последующим отбором лучших конфигураций;

- на контрольном примере был показан весь путь – от формирования конфигураций до прогнозирования коэффициентов связи и выбора оптимального решения.

Проведённая работа демонстрирует, что использование нейронных сетей в задаче размещения антенн существенно повышает эффективность проектного процесса. Модель способна быстро оценивать десятки и сотни вариантов размещения, что в условиях традиционного симуляционного анализа было бы крайне ресурсоёмко. Автоматизация данного этапа позволяет инженерным командам быстрее находить наиболее удачные схемы размещения, обеспечивающие надёжную связь и соответствие требованиям электромагнитной совместимости.

Таким образом, результаты главы подтверждают целесообразность интеграции методов искусственного интеллекта в процессы проектирования радиоэлектронных систем летательных аппаратов и служат прочной основой для построения интеллектуального модуля в составе автоматизированной системы поддержки проектных решений.

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЗИРОВАННОЙ ПРОЦЕДУРЫ РАЗМЕЩЕНИЯ АНТЕНН НА ЛЕТАТЕЛЬНОМ АППАРАТЕ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ

## 3.1. Разработка концептуальной модели базы данных.

Концептуальное проектирование базы данных – это создание информационной модели предметной области. В рамках концептуального проектирования происходит установление состава сущностей, определение совокупности атрибутов для каждой сущности, определение для каждой сущности первичного ключа и графическая реализации концептуальной процедуры.

Концептуальная модель необходима для хранения параметров, описывающих:

- летательные аппараты,
- антенны,
- конфигурации размещения,
- результаты моделирования,
- значения коэффициента связи,
- прогнозы, полученные от нейронной сети.

Присваивание сущностям названия, описания и установление количества экземпляров представлено в таблице 4:

Таблица 4. Обозначение сущностей

| № | Обозначение | Наименование сущности | Количество экземпляров |
|---|---|---|---|
| 1 | s1 | Летательный аппарат | 10 |
| 2 | s2 | Антенна | 50 |
| 3 | s3 | Конфигурация размещения | 1000 |
| 4 | s4 | Результаты моделирования | 1000 |
| 5 | s5 | Прогноз коэффициента связи | 1000 |
| 6 | s6 | Обучающая выборка | 1 |

Для каждой сущности определяется совокупность атрибутов. Атрибуты определяются по их названию, описанию и домену.

Атрибут сущности – это конкретная характеристика или свойство, которое описывает определенный аспект сущности.

Домен атрибута – это набор всех допустимых значений, которые может принимать данный атрибут. Домен определяет тип данных, формат и ограничения для конкретного атрибута.

Атрибуты и домены сущностей представлены в таблице 5:

Таблица 5. Атрибуты и домены сущностей

| Наименование сущности | Наименование атрибута | Наименование домена |
|---|---|---|
| Летательный аппарат | ID ЛА | Номер (ID) |
| | Модель ЛА | Модель |
| | Длина фюзеляжа | Габариты |
| | Радиус фюзеляжа | Габариты |
| | Материал фюзеляжа | Материал фюзеляжа |
| | Электропроводность материала | Электропроводность материала |
| | Диэлектрическая проницаемость | Диэлектрическая проницаемость |
| Антенна | ID антенны | Номер (ID) |
| | Тип антенны | Название |
| | Роль антенны | Название |
| | Рабочая частота | Числовое значение |
| | Усиление | Числовое значение |
| | Входное сопротивление | Числовое значение |
| | Диаграмма направленности (ссылка) | Путь к файлу |

Таблица 6. Атрибуты и домены сущностей (продолжение).

| | | |
|---|---|---|
| Конфигурация размещения | ID ЛА | Номер (ID) |
| | Номер варианта | Номер (ID) |
| | Номер антенны | Номер (ID) |
| | ID антенны (в связке) | Номер (ID) |
| | Координаты X | Габариты |
| | Координаты Y | Габариты |
| | Координаты Z | Габариты |
| | Ориентация θ | Числовое значение |
| | Ориентация φ | Числовое значение |
| Результаты моделирования | ID ЛА | Номер (ID) |
| | Номер варианта | Номер (ID) |
| | Коэффициент связи | Числовое значение |
| Прогноз коэффициента связи | ID ЛА | Номер (ID) |
| | Номер варианта | Номер (ID) |
| | Прогнозируемое значение коэффициента связи | Числовое значение |
| Обучающая выборка | ID ЛА | Номер (ID) |
| | Номер варианта | Номер (ID) |
| | Путь к таблице X | Путь к файлу |
| | Путь к меткам Y | Путь к файлу |

Для доменов определяются типы данных и примеры значений. Результат представлен в таблице 6:

Таблица 7. Типы данных доменов.

| № | Домен | Описание | Пример |
|---|---|---|---|
| 1 | Номер (ID) | Целое число | 1024 |
| 2 | Название | Строка до 60 символов | "Штыревая антенна" |
| 3 | Модель | Строка без пробелов до 30 | "UAV-A200" |
| 4 | Дата | дд.мм.гггг | 24.05.2025 |

Таблица 7. Типы данных доменов (продолжение).

| | | | |
|---|---|---|---|
| 5 | Габариты | Вещественное число с точностью до | 0.125 |

| | | | |
|---|---|---|---|
| | | 3 цифр после запятой | |
| 6 | Числовое значение | Вещественное число с точностью до 2 цифр после запятой | 6.72 |
| 7 | Путь к файлу | Строка-путь | /uploads/x_data.csv |

ER-диаграмма – это графическое представление логической структуры базы данных, предназначенное для описания её ключевых компонентов. Диаграмма отражает основные сущности, их атрибуты, а также связи между сущностями, существующие в предметной области.

На ER-диаграмме визуализируются: связи между таблицами базы данных, кардинальность связей (1:1, 1:N, M:N) – то есть сколько экземпляров одной сущности может быть связано с экземплярами другой, максимальное количество экземпляров, предусмотренное для каждой сущности в проекте.

ER-диаграмма представлена на рисунке 8:

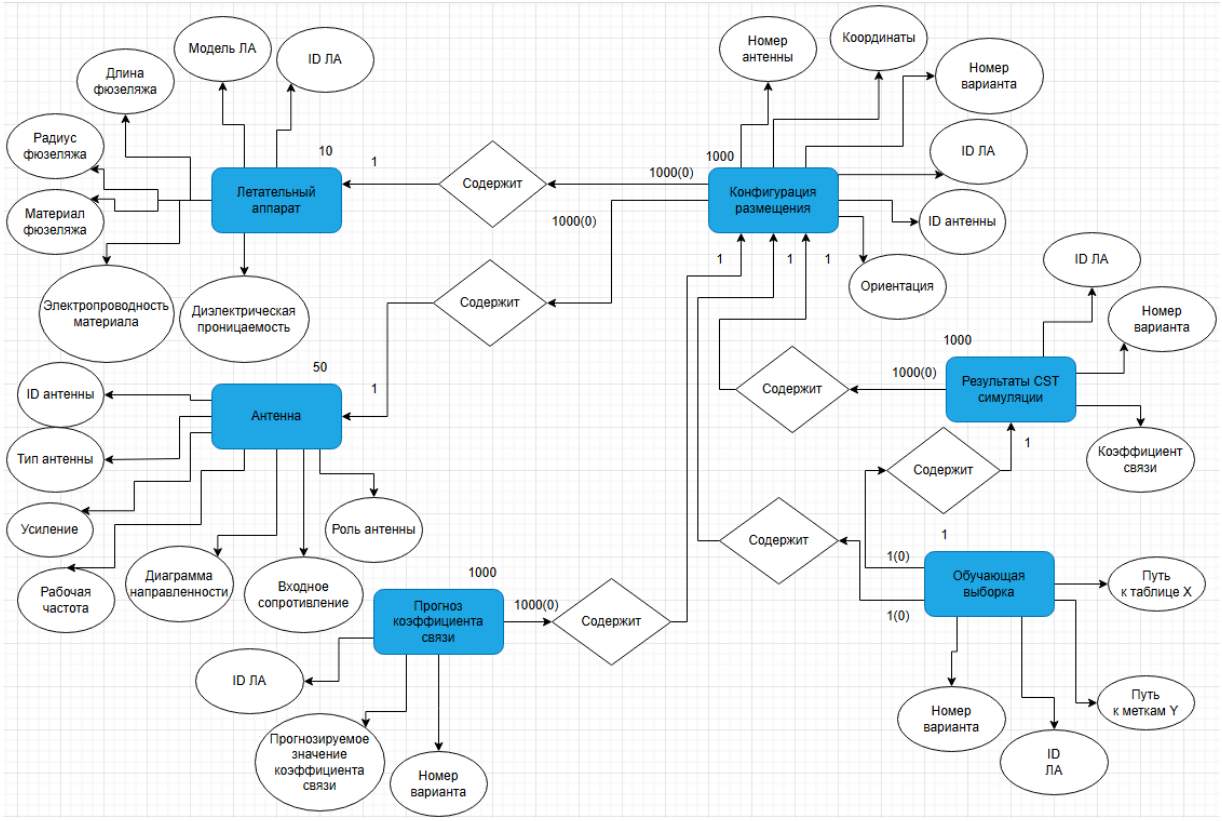


Рис. 8. ER-диаграмма

### 3.2. Разработка логической модели базы данных

Переход от концептуальной к логической модели базы данных представляет собой этап уточнения и формализации структуры хранения данных, необходимой для реализации в конкретной системе управления базами данных. На этом этапе абстрактные сущности, связи и атрибуты, определённые в рамках ER-моделирования, трансформируются в набор логических отношений (таблиц) с чётким определением:

- первичных ключей (PK) – уникальных идентификаторов строк или составных ключей,
- внешних ключей (FK) – ссылок на связанные таблицы,
- типов данных для всех атрибутов,
- ограничений целостности, обеспечивающих корректность и непротиворечивость данных.

Логическая модель описывает структуру предметной области в виде набора нормализованных отношений и отражает основные зависимости между объектами: композиционные, ссылочные, иерархические и т.д. Она служит промежуточным представлением между концептуальной моделью и физической реализацией, оставаясь при этом независимой от конкретной платформы или системы управления базой данных.

Связи между отношениями следующие:

- Летательный аппарат ↔ Конфигурация размещения – связь один ко многим. Один летательный аппарат может использоваться в большом числе конфигураций размещения антенн. Каждая конкретная конфигурация однозначно привязана к одному летательному аппарату и к одному варианту.
- Антенна ↔ Конфигурация размещения – связь один ко многим. Каждая антенна (по типу) может использоваться в различных вариантах размещения и в составе разных вариантов не дублируясь.

- Конфигурация размещения ↔ Моделирование – связь один к одному. Каждому варианту размещения летательного аппарата соответствует одно моделирование, хранящая итоговый коэффициент связи.
- Конфигурация размещения ↔ Прогноз связи – связь один к одному. Каждому варианту соответствует прогноз коэффициента связи, полученный от обученной модели.
- Обучающая выборка ↔ Прогноз связи - связь один ко многим. Один набор обучающих данных может быть использован для оценки или генерации прогнозов для нескольких конфигураций.

Нормализация отношений представлена в таблице 6:

Таблица 8. Нормализация отношений

| Наименование сущности | Наименование атрибута | F1 |
|---|---|---|
| Летательный аппарат | ID ЛА | * |
| | Модель ЛА | ← |
| | Длина фюзеляжа | ← |
| | Радиус фюзеляжа | ← |
| | Материал фюзеляжа | ← |
| | Электропроводность материала | ← |
| | Диэлектрическая проницаемость | ← |
| Антенна | ID антенны | * |
| | Тип антенны | ← |
| | Роль антенны | ← |
| | Рабочая частота | ← |
| | Усиление | ← |
| | Входное сопротивление | ← |
| | Диаграмма направленности (ссылка) | ← |

Таблица 8. Нормализация отношений (продолжение)

| | | |
|---|---|---|
| Конфигурация размещения | ID ЛА | * |
| | Номер варианта | * |
| | Номер антенны | * |
| | ID антенны (в связке) | ← |
| | Координаты X | ← |
| | Координаты Y | ← |
| | Координаты Z | ← |
| | Ориентация θ | ← |
| | Ориентация φ | ← |
| Результаты моделирования | ID ЛА | * |
| | Номер варианта | * |
| | Коэффициент связи | ← |
| Прогноз коэффициента связи | ID ЛА | * |
| | Номер варианта | * |
| | Прогнозируемое значение коэффициента связи | ← |
| Обучающая выборка | ID ЛА | * |
| | Номер варианта | * |
| | Путь к таблице X | ← |
| | Путь к меткам Y | ← |

Проведённый процесс нормализации показал, что связи, представленные в таблице 4, не содержат сложных или составных атрибутов. Каждый неключевой атрибут в каждой связи функционально полностью зависит от первичного ключа соответствующего отношения, отсутствует транзитивная зависимость. Таким образом, все отношения приведены к третьей нормальной форме, что обеспечивает устранение избыточности данных, повышает целостность информации и минимизирует вероятность возникновения аномалий.

На рисунке 8 представлена логическая модель базы данных:

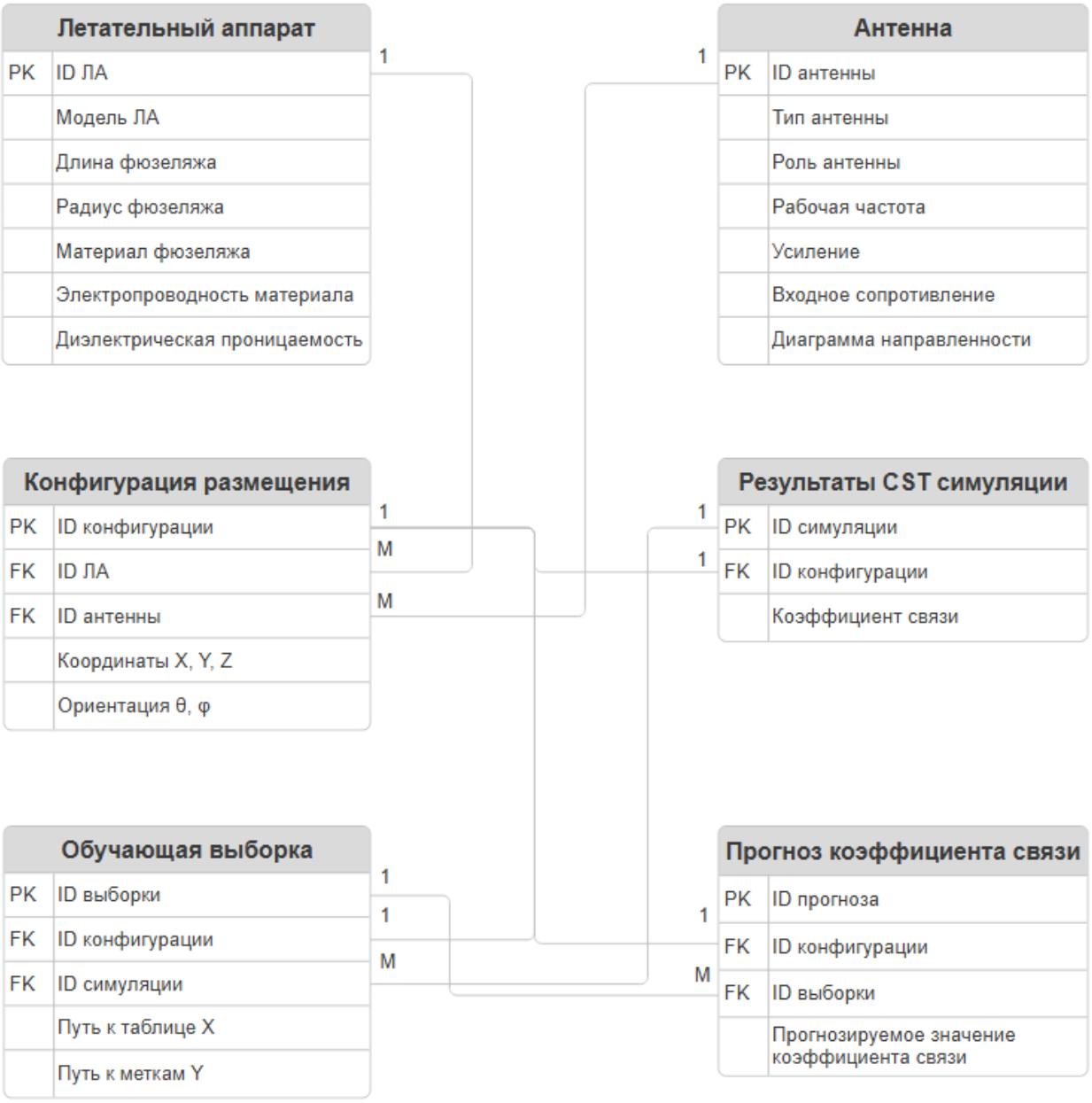


Рис. 9. Логическая модель базы данных

### 3.3. Физическое проектирование базы данных

Физическое проектирование базы данных представляет собой этап, на котором логическая модель преобразуется в конкретную реализацию с учётом технической платформы, форматов хранения, типов данных и параметров производительности. На этом этапе создаются структуры таблиц, определяются типы атрибутов, первичные и внешние ключи, индексы и связи между таблицами в соответствии с выбранной системой управления базами данных.

В рамках данной работы для хранения информации о параметрах антенн, конфигурациях размещения, результатах моделирования и прогнозах коэффициента связи была выбрана система управления базами данных MySQL. Эта СУБД зарекомендовала себя как надёжная, масштабируемая и хорошо документированная платформа с открытым исходным кодом, поддерживающая высокую скорость обработки запросов и возможность гибкой интеграции с аналитическими и инженерными инструментами.

Для локального развертывания среды разработки использован OpenServer – популярный серверный пакет, включающий MySQL и PHPMyAdmin, который обеспечивает простую установку, настройку и управление базами данных в процессе проектирования, тестирования и отладки.

Физическая структура базы данных соответствует требованиям к корректному хранению проектной информации, обучающих выборок и результатов симуляционного анализа, и может быть использована в составе комплексной системы автоматизации размещения антенн на летательных аппаратах.

Примеры заполнения всех таблиц базы данных, построенной в рамках данной работы, приведены на следующей странице:

На рисунках 9-14 проиллюстрированы примеры всех таблиц из базы данных «Автоматизации размещения антенн на летательном аппарате»:

| ID ЛА | Модель ЛА | Длина фюзеляжа | Радиус фюзеляжа | Материал фюзеляжа | Электропроводность материала | Диэлектрическая проницаемость |
|---|---|---|---|---|---|---|
| 1 | MQ-1 Predator | 8,22 | 0,45 | Композит | 10 | 2,5 |
| 2 | MQ-9 Reaper | 11 | 0,55 | Композит | 10 | 2,5 |
| 3 | Bayraktar TB2 | 6,5 | 0,4 | Композит | 10 | 2,5 |
| 4 | RQ-4 Global Hawk | 13,5 | 0,65 | Алюминий | 35 | 1,2 |
| 5 | Heron TP | 14 | 0,6 | Композит | 10 | 2,5 |
| 6 | ScanEagle | 1,5 | 0,2 | Углепластик | 4 | 6,8 |
| 7 | RQ-7 Shadow | 3,4 | 0,35 | Композит | 10 | 2,5 |
| 8 | Orion UAV | 8 | 0,5 | Алюминий | 35 | 1,2 |
| 9 | CH-4 Rainbow | 8,5 | 0,48 | Титан | 25,5 | 1,1 |
| 10 | Wing Loong II | 9 | 0,5 | Композит | 10 | 2,5 |

Рис. 10. Таблица «Летательный аппарат»

| ID антенны | Тип антенны | Роль антенны | Рабочая частота | Усиление | Входное сопротивление | Диаграмма направленности |
|---|---|---|---|---|---|---|
| 1 | Штыревая | Приемник | 433 | 2,15 | 50 | /files/diagr_1.png |
| 2 | Штыревая | Передатчик | 433 | 2,15 | 50 | /files/diagr_2.png |
| 3 | Штыревая | Приемник | 868 | 2,15 | 50 | /files/diagr_3.png |
| 4 | Штыревая | Передатчик | 868 | 2,15 | 50 | /files/diagr_4.png |
| 5 | Штыревая | Приемник | 1 800 | 2,15 | 50 | /files/diagr_5.png |
| 6 | Штыревая | Передатчик | 1 800 | 2,15 | 50 | /files/diagr_6.png |
| 7 | Штыревая | Приемник | 2 400 | 2,15 | 50 | /files/diagr_7.png |
| 8 | Штыревая | Передатчик | 2 400 | 2,15 | 50 | /files/diagr_8.png |
| 9 | Штыревая | Приемник | 5 800 | 2,15 | 50 | /files/diagr_9.png |
| 10 | Штыревая | Передатчик | 5 800 | 2,15 | 50 | /files/diagr_10.png |
| 11 | Дипольная | Приемник | 433 | 2,15 | 73 | /files/diagr_11.png |
| 12 | Дипольная | Передатчик | 433 | 2,15 | 73 | /files/diagr_12.png |
| 13 | Дипольная | Приемник | 868 | 2,15 | 73 | /files/diagr_13.png |
| 14 | Дипольная | Передатчик | 868 | 2,15 | 73 | /files/diagr_14.png |
| 15 | Дипольная | Приемник | 1 800 | 2,15 | 73 | /files/diagr_15.png |
| 16 | Дипольная | Передатчик | 1 800 | 2,15 | 73 | /files/diagr_16.png |
| 17 | Дипольная | Приемник | 2 400 | 2,15 | 73 | /files/diagr_17.png |
| 18 | Дипольная | Передатчик | 2 400 | 2,15 | 73 | /files/diagr_18.png |
| 19 | Дипольная | Приемник | 5 800 | 2,15 | 73 | /files/diagr_19.png |
| 20 | Дипольная | Передатчик | 5 800 | 2,15 | 73 | /files/diagr_20.png |
| 21 | Рупорная | Приемник | 433 | 15 | 50 | /files/diagr_21.png |
| 22 | Рупорная | Передатчик | 433 | 15 | 50 | /files/diagr_22.png |
| 23 | Рупорная | Приемник | 868 | 15 | 50 | /files/diagr_23.png |
| 24 | Рупорная | Передатчик | 868 | 15 | 50 | /files/diagr_24.png |
| 25 | Рупорная | Приемник | 1 800 | 15 | 50 | /files/diagr_25.png |
| 26 | Рупорная | Передатчик | 1 800 | 15 | 50 | /files/diagr_26.png |
| 27 | Рупорная | Приемник | 2 400 | 15 | 50 | /files/diagr_27.png |
| 28 | Рупорная | Передатчик | 2 400 | 15 | 50 | /files/diagr_28.png |
| 29 | Рупорная | Приемник | 5 800 | 15 | 50 | /files/diagr_29.png |
| 30 | Рупорная | Передатчик | 5 800 | 15 | 50 | /files/diagr_30.png |
| 31 | Панельная | Приемник | 433 | 9 | 50 | /files/diagr_31.png |
| 32 | Панельная | Передатчик | 433 | 9 | 50 | /files/diagr_32.png |
| 33 | Панельная | Приемник | 868 | 9 | 50 | /files/diagr_33.png |
| 34 | Панельная | Передатчик | 868 | 9 | 50 | /files/diagr_34.png |
| 35 | Панельная | Приемник | 1 800 | 9 | 50 | /files/diagr_35.png |
| 36 | Панельная | Передатчик | 1 800 | 9 | 50 | /files/diagr_36.png |
| 37 | Панельная | Приемник | 2 400 | 9 | 50 | /files/diagr_37.png |
| 38 | Панельная | Передатчик | 2 400 | 9 | 50 | /files/diagr_38.png |
| 39 | Панельная | Приемник | 5 800 | 9 | 50 | /files/diagr_39.png |

Рис. 11. Таблица «Антенны»

арсанла.конфигурация размещения: 5 строк (приблизительно)

| ID ЛА | Номер варианта | Номер антенны | ID антенны | Координата X | Координата Y | Координата Z | Ориентация θ | Ориентация φ |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1,200 | 0,500 | 0,300 | 15,00 | 45,00 |
| 1 | 1 | 2 | 2 | 1,400 | 0,600 | 0,320 | 10,00 | 30,00 |
| 2 | 1 | 1 | 3 | 0,900 | 0,300 | 0,250 | 5,00 | 60,00 |
| 3 | 2 | 1 | 4 | 1,000 | 0,400 | 0,200 | 20,00 | 90,00 |
| 3 | 2 | 2 | 5 | 1,100 | 0,420 | 0,230 | 25,00 | 75,00 |

Рис. 12. Таблица «Конфигурация размещения»

арсанла.результаты cst симуляции: 5 строк (приблизительно)

| ID ЛА | Номер варианта | Коэффициент связи |
|---|---|---|
| 1 | 1 | -17,50 |
| 2 | 1 | -15,80 |
| 3 | 2 | -20,30 |
| 4 | 1 | -14,70 |
| 5 | 1 | -16,90 |

Рис. 13. Таблица «Результаты моделирования»

арсанла.прогноз коэффициента связи: 5 строк (приблизительно)

| ID ЛА | Номер варианта | Прогнозируемое значение коэффициента связи |
|---|---|---|
| 1 | 1 | -17,20 |
| 2 | 1 | -15,40 |
| 3 | 2 | -19,90 |
| 4 | 1 | -14,10 |
| 5 | 1 | -17,00 |

Рис. 14. Таблица «Прогноз коэффициента связи»

арсанла.обучающая выборка: 1 строк (приблизительно)

| ID ЛА | Номер варианта | Путь к таблице X | Путь к меткам Y |
|---|---|---|---|
| 1 | 1 | /data/x_table.csv | /data/y_labels.csv |

Рис. 15. Таблица «Обучающая выборка»

Для работы с данными из базы данных созданы два типа SQL запросов: включение данных в таблицу и удаление данных из таблицы. Примеры запросов приведены на рисунках 15-18:

На рисунке 15 показан SQL-запрос, добавляющий новую запись в таблицу «Летательный аппарат»:

```
INSERT INTO `Летательный аппарат` (`ID ЛА`,
`Модель ЛА`,
`Длина фюзеляжа`,
`Радиус фюзеляжа`,
`Материал фюзеляжа`,
`Электропроводность материала`,
`Диэлектрическая проницаемость`)
VALUES (6, 'UAV-F', 3.200, 0.480, 'Композит', 8.20, 3.60);


```

Рис. 16. Код запроса добавления нового летательного аппарата

На рисунке 16 показан SQL-запрос, удаляющий запись из таблицы «Летательный аппарат»:

```
DELETE FROM `Летательный аппарат`
WHERE `ID ЛА` = 6;
```

Рис. 17. Код запроса удаления летательного аппарата

На рисунке 17 показан SQL-запрос, добавляющий новую запись в таблицу «Антенна»:

```
INSERT INTO `Антенна` (`ID антенны`,
`Тип антенны`,
`Рабочая частота`,
`Усиление`,
`Входное сопротивление`,
`Диаграмма направленности`)
VALUES (6, 'Щелевая широкополосная', 510.5, 3.8, 75, '/files/diagr_6.png');
```

Рис. 18. Код запроса добавления новой антенны

На рисунке 18 показан SQL-запрос, удаляющий запись из таблицы «Антенна»:

```
DELETE FROM `Антенна`
WHERE `ID антенны` = 6;
```

Рис. 19. Код запроса удаления антенны

## Выводы по главе 3

В третьей главе была выполнена разработка информационного обеспечения задачи автоматизации размещения антенн на беспилотном

летательном аппарате, что является ключевым элементом для обеспечения надёжного хранения, обработки и последующего анализа проектных данных. В рамках главы были поэтапно реализованы три уровня проектирования базы данных: концептуальный, логический и физический.

На этапе концептуального проектирования определены основные сущности предметной области, включая: летательный аппарат, антенну, конфигурацию размещения, результаты моделирования, прогноз коэффициента связи и обучающую выборку. Для каждой сущности были установлены атрибуты, их домены и допустимые значения. Также проведена работа по нормализации структуры данных и исключению избыточности – путём выделения повторяющихся параметров в отдельные таблицы и формирования составных первичных ключей.

В логической модели уточнена структура всех отношений, определены первичные и внешние ключи, а также связи между таблицами. Особое внимание уделено корректному отображению связей типа "один ко многим" и "один к одному" – между летательным аппаратом и конфигурациями размещения, антеннами и размещениями, а также между размещениями и результатами моделирования или прогнозами. Учтены рекомендации по представлению координат и ориентаций антенн в отдельных атрибутах, а также по организации повторного использования одних и тех же антенн в различных вариантах компоновки.

Физическое проектирование завершило формирование архитектуры базы данных. В качестве платформы выбрана СУБД MySQL, обеспечивающая высокую производительность, масштабируемость и совместимость с инженерными инструментами. Все таблицы были реализованы в SQL, выполнено создание структур базы данных и заполнение их тестовыми данными. Предусмотрены SQL-запросы на добавление и удаление записей, что подтверждает прикладную реализуемость информационного решения.

Таким образом, информационное обеспечение проекта размещения антенн спроектировано с учётом требований к гибкости, расширяемости и целостности данных. Реализованная база данных может быть эффективно использована как основа для хранения результатов моделирования, обучения моделей машинного обучения, а также интеграции с CAD/CAE-средами и системами поддержки инженерных решений.

# ГЛАВА 4. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЗИРОВАННОЙ ПРОЦЕДУРЫ РАЗМЕЩЕНИЯ АНТЕНН НА ЛЕТАТЕЛЬНОМ АППАРАТЕ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ.

## 4.1. Анализ функциональных требований к программе.

Для наглядного представления основных функций разрабатываемого программного обеспечения была составлена UML-диаграмма вариантов использования, которая представлена на рисунке 19:

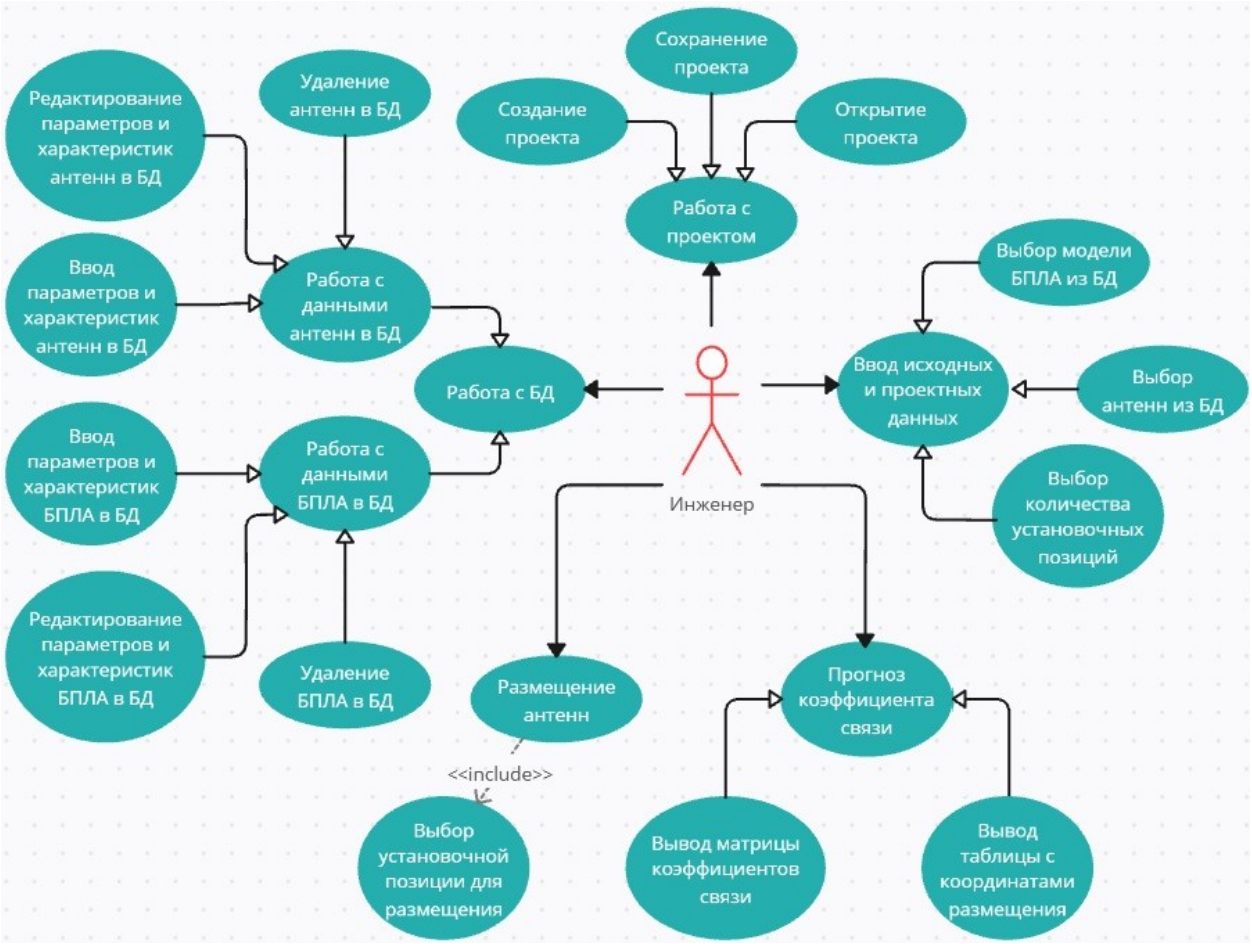


Рис. 20. Диаграмма вариантов использования

Данная диаграмма отражает взаимодействие пользователя (инженера) с ключевыми функциональными возможностями программы и демонстрирует общий сценарий её применения.

Инженер – пользователь, ответственный за взаимодействие с системой автоматизированного размещения антенн на летательном аппарате на основе нейронных сетей. Также он управляет базой данных и работает с проектом, включая создание, сохранение и открытие

В таблице 7 представлено описание прецендентов UML-диаграммы вариантов использования для автоматизированной системы размещения антенн на летательном аппарате на основе нейронных сетей:

Таблица 9. Описание прецендентов

| Прецендент | Краткое описание |
|---|---|
| Работа с БД | Общий процесс взаимодействия с базой данных, объединяющий операции по работе с данными БПЛА и антенн. |
| Работа с данными БПЛА в БД | Позволяет работать с записями моделей БПЛА в базе данных (ввод, редактирование, удаление). |
| Ввод параметров и характеристик БПЛА в БД | Осуществляется добавление новых моделей БПЛА в базу данных с заданием их характеристик (размеров, материала и др.). |
| Редактирование параметров и характеристик БПЛА в БД | Позволяет изменять ранее введённые данные о моделях БПЛА в базе данных. |
| Удаление БПЛА в БД | Применяется для удаления моделей БПЛА из базы данных. |
| Работа с данными антенн в БД | Позволяет работать с записями антенн в базе данных (ввод, редактирование, удаление). |
| Ввод параметров и характеристик антенн в БД | Осуществляется добавление новых антенн в базу данных с заданием их характеристик (тип, роль, частота, усиление и др.). |
| Редактирование параметров и характеристик антенн в БД | Позволяет изменять ранее введённые данные об антеннах в базе данных. |
| Удаление антенн в БД | Применяется для удаления записей антенн из базы данных. |

Таблица 9. Описание прецендентов (продолжение)

| Прецендент | Краткое описание |
|---|---|
| Работа с проектом | Включает создание нового проекта, сохранение текущего проекта и открытие ранее сохранённого проекта. |
| Создание проекта | Позволяет создать новый проект для проведения процедуры размещения антенн. |
| Сохранение проекта | Обеспечивает сохранение текущего состояния проекта, включая данные о выбранной модели БПЛА, антеннах и их размещении. |
| Открытие проекта | Позволяет открыть ранее сохранённый проект для продолжения работы. |
| Ввод исходных и проектных данных | Используется для выбора модели БПЛА, антенн и задания количества установочных позиций, необходимых для дальнейшего размещения. |
| Выбор БПЛА из БД | Позволяет выбрать из базы данных модель БПЛА для использования в проекте. |
| Выбор антенн из БД | Позволяет выбрать из базы данных одну или несколько антенн для размещения на выбранной модели БПЛА. |
| Выбор количества установочных позиций | Позволяет задать количество доступных позиций для установки антенн на фюзеляже БПЛА. |
| Размещение антенн | Включает процесс выбора установочных позиций и размещения антенн на 3D-модели БПЛА. |
| Выбор установочной позиции для размещения | Подпрецедент размещения антенн, обеспечивающий выбор конкретной позиции на 3D-модели фюзеляжа для каждой антенны. |

Таблица 9. Описание прецендентов (продолжение)

| | |
|---|---|
| Прогноз коэффициента связи | Используется для расчёта прогнозных значений коэффициентов связи между размещёнными антеннами. |
| Вывод матрицы коэффициентов связи | Обеспечивает отображение рассчитанной матрицы коэффициентов связи между парами установленных антенн. |
| Вывод таблицы с координатами размещения | Позволяет вывести таблицу с координатами выбранных установочных позиций всех размещённых антенн. |

Для большего понимания вариантов использования, ниже будут приведены сценарии, которые будут являться возможной последовательностью действий пользователя. Все сценарии перечислены в следующих таблицах:

Таблица 10. Сценарий выполнения прецендента «Ввод параметров и характеристик БПЛА в БД»

| Актер (инженер) | Программная система |
|---|---|
| 1. Выбор пункта меню «База данных → Добавить БПЛА». | 2. Открытие окна «Добавить БПЛА». |
| 3. Ввод всех параметров БПЛА: модель, длина, радиус, материал, электропроводность, диэлектрическая проницаемость. | 4. Ожидание нажатия кнопки «Добавить». |
| 5. Нажатие кнопки «Добавить». | 6. Проверка корректности и полноты введённых данных. |
| Исключение №1. Не все поля заполнены. | |

| | |
|---|---|
| | 8. Вывод сообщения об успехе. |
| | 9. Ожидание дальнейших действий пользователя. |

Таблица 11. Сценарий обработки исключительной ситуации №1 «Не все поля заполнены»

| Актер (инженер) | Программная система |
|---|---|
| 5. Нажатие кнопки «Добавить». | 6. Вывод окна с предупреждением «Все поля должны быть заполнены». |
| | 7. Открытие формы для добавления антенны. |
| | 8. Ожидание дальнейших действий пользователя. |

Таблица 12. Сценарий выполнения прецендента «Редактирование параметров и характеристик БПЛА в БД»

| Актер (инженер) | Программная система |
|---|---|
| 1. Выбор пункта меню «База данных → Редактировать БПЛА». | 2. Открытие окна выбора БПЛА. |
| 3. Выбор нужной модели БПЛА. | 4. Открытие окна «Редактировать БПЛА». |
| 5. Нажатие кнопки «Изменить». | 6. Проверка корректности и полноты введённых данных. |
| Исключение №1. Не все поля заполнены. | |
| 7. Сохранение изменений в базе данных. | 8. Ожидание дальнейших действий пользователя. |

Таблица 13. Сценарий обработки исключительной ситуации №1 «Не все поля заполнены»

| Актер (инженер) | Программная система |
|---|---|
| 5. Нажатие кнопки «Редактировать». | 6. Вывод окна с предупреждением «Все поля должны быть заполнены». |
| | 7. Открытие формы для редактирования БПЛА. |
| | 8. Ожидание дальнейших действий пользователя. |

Таблица 14. Сценарий выполнения прецендента «Удаление БПЛА из БД»

| Актер (инженер) | Программная система |
|---|---|
| 1. Выбор пункта меню «База данных → Удалить БПЛА». | 2. Открытие окна выбора БПЛА. |
| 3. Выбор нужной модели БПЛА. | 4. Открытие окна «Удаление БПЛА». |
| 5. Нажатие кнопки «Удалить». | 6. Запрос подтверждения удаления. |
| Исключение №2. Отмена удаления пользователем. | |
| 7. Подтверждение удаления. | 8. Удаление БПЛА из базы данных. |
| | 9. Ожидание дальнейших действий пользователя. |

Таблица 15. Сценарий обработки исключительной ситуации №1 «Отмена удаления пользователем»

| Актер (инженер) | Программная система |
|---|---|
| 5. Нажатие кнопки «Удалить». | 6. Запрос подтверждения удаления. |
| 7. Нажатие «Нет». | 8. Возврат в окно выбора БПЛА. |
| | 9. Ожидание дальнейших действий пользователя. |

Таблица 16. Сценарий выполнения прецендента «Ввод параметров и характеристик антенны в БД»

| Актер (инженер) | Программная система |
|---|---|
| 1. Выбор пункта меню «База данных → Добавить антенну». | 2. Открытие окна «Добавить антенну». |
| 3. Ввод всех параметров антенны. | 4. Ожидание нажатия кнопки «Добавить». |
| 5. Нажатие кнопки «Добавить». | 6. Проверка корректности и полноты введённых данных. |
| Исключение №1. Не все поля заполнены. | |
| | 8. Добавление антенны в базу данных. |
| | 9. Ожидание дальнейших действий пользователя. |

Таблица 17. Сценарий обработки исключительной ситуации №1 «Не все поля заполнены»

| Актер (инженер) | Программная система |
|---|---|
| 5. Нажатие кнопки «Добавить». | 6. Вывод окна с предупреждением «Все поля должны быть заполнены». |
| | 7. Открытие формы для редактирования антенны. |
| | 8. Ожидание дальнейших действий пользователя. |

Таблица 18. Сценарий выполнения прецендента «Ввод параметров и характеристик антенны в БД»

| Актер (инженер) | Программная система |
|---|---|
| 1. Выбор пункта меню «База данных → Редактировать антенну». | 2. Открытие окна выбора антенны. |
| 3. Выбор нужной антенны. | 4. Открытие окна «Редактировать антенну». |
| 5. Внесение изменений. | 6. Ожидание нажатия кнопки «Изменить». |
| 7. Нажатие кнопки «Изменить». | 8. Проверка корректности и полноты введённых данных. |
| Исключение №1. Не все поля заполнены. | |
| | 8. Вывод сообщения об успешном редактировании. |

| | |
|---|---|
| | 9. Ожидание дальнейших действий пользователя. |

Таблица 19. Сценарий обработки исключительной ситуации №1 «Не все поля заполнены»

| Актер (инженер) | Программная система |
|---|---|
| 5. Нажатие кнопки «Редактировать». | 6. Вывод окна с предупреждением «Все поля должны быть заполнены». |
| | 7. Открытие формы для редактирования антенны. |
| | 8. Ожидание дальнейших действий пользователя. |

Таблица 20. Сценарий выполнения прецендента «Удаление антенны из БД»

| Актер (инженер) | Программная система |
|---|---|
| 1. Выбор пункта меню «База данных → Удалить антенну». | 2. Открытие окна выбора антенны. |
| 3. Выбор нужной антенны. | 4. Ожидание нажатия кнопки «Удалить». |
| 5. Нажатие кнопки «Удалить». | 6. Запрос подтверждения удаления. |
| Исключение №2. Отмена удаления пользователем. | |
| 7. Удаление антенны из базы данных. | 8. Вывод сообщения об успешном удалении. |
| | 9. Ожидание дальнейших действий пользователя. |

Таблица 21. Сценарий обработки исключительной ситуации №1 «Отмена удаления пользователем»

| Актер (инженер) | Программная система |
|---|---|
| 5. Нажатие кнопки «Удалить». | 6. Запрос подтверждения удаления. |
| 7. Нажатие «Нет». | 8. Возврат в окно выбора антенны. |
| | 9. Ожидание дальнейших действий пользователя. |

Таблица 22. Сценарий выполнения прецендента «Выбор БПЛА из БД»

| Актер (инженер) | Программная система |
|---|---|
| 1. Выбор пункта меню «Файл → Новая вкладка». | 2. Открытие окна «Загрузка данных». |
| 3. Выбор модели БПЛА. | 4. Подгрузка выбранных данных в программу. |
| | 5. Ожидание дальнейших действий пользователя. |

Таблица 23. Сценарий выполнения прецендента «Выбор антенн из БД»

| Актер (инженер) | Программная система |
|---|---|
| 1. Выбор одной или более антенн в окне «Загрузка данных». | 2. Подгрузка выбранных данных в БД |

| | |
|---|---|
| | 3. Ожидание дальнейших действий пользователя. |

Таблица 24. Сценарий выполнения прецендента «Размещение антенн»

| Актер (инженер) | Программная система |
|---|---|
| 1 Нажатие кнопки «Разместить». | 2. Подсветка доступных установочных позиций. |
| 3. Щелчок правой кнопкой мыши по позиции. | 4. Размещение антенны в выбранной позиции. |
| 5. Нажатие кнопки «Завершить размещение». | 6. Скрытие подсветки позиций. |
| | 7. Ожидание дальнейших действий пользователя. |

Таблица 25. Сценарий выполнения прецендента «Прогноз коэффициента связи»

| Актер (инженер) | Программная система |
|---|---|
| 1. Нажатие кнопки «Прогноз коэффициента связи». | 2. Проверка наличия хотя бы одной антенны приемника и передатчика. |
| Исключение №3: «Нет антенны приемника или передатчика» | |
| 3. Расчёт матрицы коэффициентов связи. | 4. Отображение матрицы в окне. |
| 5. Нажатие кнопки «Координаты размещения» | 6. Отображение таблицы с координатами размещения антенн. |

| | |
|---|---|
| | 7. Ожидание дальнейших действий пользователя. |

Таблица 26. Сценарий обработки исключительной ситуации №3 «Нет антенны приемника или передатчика»

| Актер (инженер) | Программная система |
|---|---|
| 1. Нажатие кнопки «Прогноз коэффициента связи». | 2. Вывод сообщения «Для прогноза необходимо хотя бы одна антенна-«приемник» и одна антенна-«передатчик».» |
| | 3. Ожидание дальнейших действий пользователя. |

### 4.2. Архитектура программы.

Диаграмма классов – это структурная диаграмма в рамках UML, которая описывает систему путём показа классов объектов, их внутренней структуры и отношений между ними. Диаграмма классов представлена на рисунке 20:

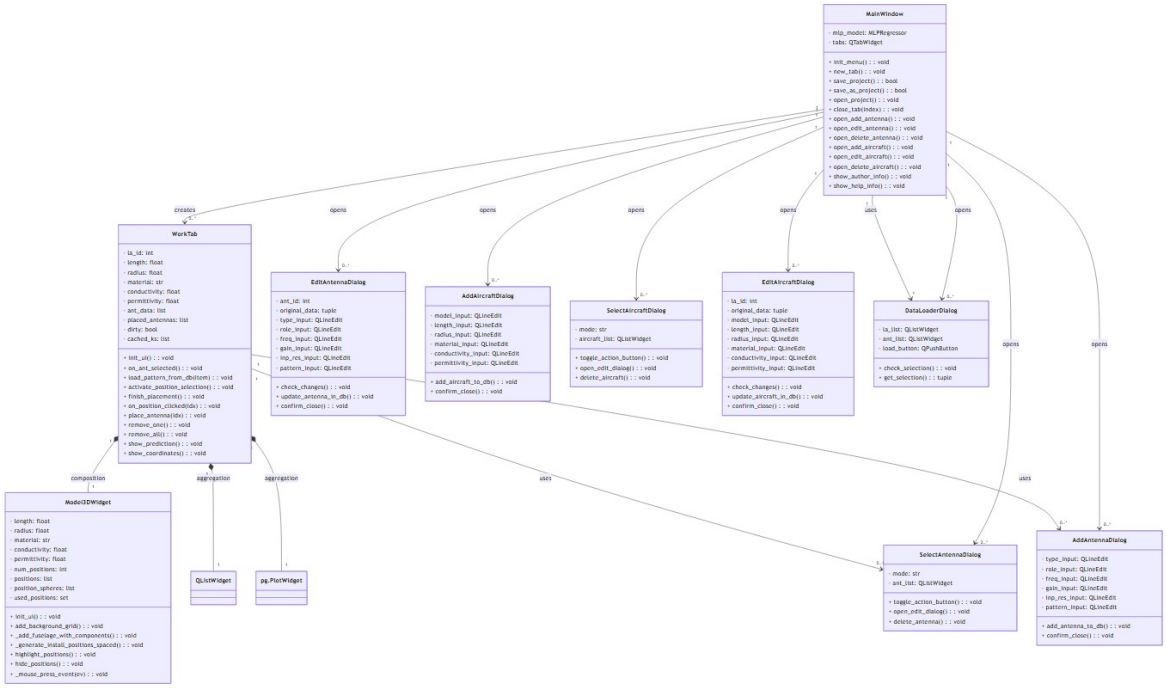


Рис. 21. Диаграмма классов

Описание классов программы представлены в следующих таблицах:

Таблица 27. Класс: DataLoaderDialog

| Параметр | Значение |
|---|---|
| Комментарий | Диалог для загрузки БПЛА и антенн перед началом работы. |
| Атрибуты | - la_list: QListWidget – список БПЛА<br>- ant_list: QListWidget – список антенн<br>- load_button: QPushButton – кнопка загрузки |
| Операции | + check_selection(): void – проверка выбора БПЛА и антенн<br>+ get_selection(): tuple – получить выбор пользователя |

Таблица 28. Класс: MainWindow

| Параметр | Значение |
|---|---|
| Комментарий | Главное окно программы для работы с проектами размещения антенн. |
| Атрибуты | - mlp_model: MLPRegressor – модель MLP для архитектуры программы<br>- tabs: QTabWidget – вкладки с проектами |
| Операции | + init_menu(): void – инициализация главного меню<br>+ new_tab(): void – создать новую вкладку<br>+ save_project(): bool – сохранить проект<br>+ save_as_project(): bool – сохранить проект как...<br>+ open_project(): void – открыть проект<br>+ close_tab(index): void – закрыть вкладку<br>+ open_add_antenna(): void – открыть диалог добавления антенны<br>+ open_edit_antenna(): void – открыть диалог редактирования антенны<br>+ open_delete_antenna(): void – открыть диалог удаления антенны<br>+ open_add_aircraft(): void – открыть диалог добавления БПЛА<br>+ open_edit_aircraft(): void – открыть диалог редактирования БПЛА<br>+ open_delete_aircraft(): void – открыть диалог удаления БПЛА<br>+ show_author_info(): void – показать информацию об авторе<br>+ show_help_info(): void – показать справку |

Таблица 29. Класс: WorkTab

| Параметр | Значение |
|---|---|
| Комментарий | Вкладка для работы с конкретным проектом размещения антенн. |
| Атрибуты | - la_id: int – идентификатор БПЛА<br>- length: float – длина фюзеляжа<br>- radius: float – радиус фюзеляжа<br>- material: str – материал фюзеляжа<br>- conductivity: float – электропроводность материала<br>- permittivity: float – диэлектрическая проницаемость<br>- ant_data: list – данные по выбранным антеннам<br>- placed_antennas: list – размещённые антенны<br>- dirty: bool – флаг наличия несохранённых изменений<br>- cached_ks: list – кэшированный прогноз коэффициентов связи |
| Операции | + init_ui(): void – инициализация интерфейса вкладки<br>+ on_ant_selected(): void – обработка выбора антенны<br>+ load_pattern_from_db(item): void – загрузка диаграммы направленности<br>+ activate_position_selection(): void – активировать режим размещения<br>+ finish_placement(): void – завершить режим размещения<br>+ on_position_clicked(idx): void – обработка клика<br>+ place_antenna(idx): void – разместить антенну<br>+ remove_one(): void – удалить одну антенну<br>+ remove_all(): void – удалить все антенны<br>+ show_prediction(): void – прогноз коэффициента связи<br>+ show_coordinates(): void – показать координаты размещённых антенн |

Таблица 30. Класс: Model3DWidget

| Параметр | Значение |
|---|---|
| Комментарий | Виджет отображения 3D модели БПЛА с установочными позициями антенн. |
| Атрибуты | - length: float – длина фюзеляжа<br>- radius: float – радиус фюзеляжа<br>- material: str – материал фюзеляжа<br>- conductivity: float – электропроводность<br>- permittivity: float – диэлектрическая проницаемость<br>- num_positions: int – количество установочных позиций<br>- positions: list – координаты позиций<br>- position_spheres: list – сферы визуализации позиций<br>- used_positions: set – занятые позиции |
| Операции | + init_ui(): void – инициализация интерфейса<br>+ add_background_grid(): void – добавить фон<br>+ _add_fuselage_with_components(): void – добавить фюзеляж<br>+ _generate_install_positions_spaced(): void – сгенерировать позиции<br>+ highlight_positions(): void – подсветить позиции<br>+ hide_positions(): void – скрыть позиции<br>+ _mouse_press_event(ev): void – обработка клика по модели |

Таблица 31. Класс: AddAntennaDialog

| Параметр | Значение |
|---|---|
| Комментарий | Диалог добавления новой антенны в БД. |
| Атрибуты | - type_input: QLineEdit – тип антенны<br>- role_input: QLineEdit – роль антенны<br>- freq_input: QLineEdit – рабочая частота<br>- gain_input: QLineEdit – усиление<br>- inp_res_input: QLineEdit – входное сопротивление<br>- pattern_input: QLineEdit – диаграмма направленности |
| Операции | + add_antenna_to_db(): void – добавить антенну в БД<br>+ confirm_close(): void – подтвердить закрытие окна |

Таблица 32. Класс: SelectAntennaDialog

| Параметр | Значение |
|---|---|
| Комментарий | Диалог выбора антенны для редактирования или удаления. |
| Атрибуты | - mode: str – режим (edit / delete)<br>- ant_list: QListWidget – список антенн |
| Операции | + toggle_action_button(): void – включение/выключение кнопки действия<br>+ open_edit_dialog(): void – открыть диалог редактирования<br>+ delete_antenna(): void – удалить выбранную антенну |

Таблица 33. Класс: EditAntennaDialog

| Параметр | Значение |
|---|---|
| Комментарий | Диалог редактирования антенны. |
| Атрибуты | - ant_id: int – ID антенны<br>- original_data: tuple – исходные данные<br>- type_input: QLineEdit<br>- role_input: QLineEdit<br>- freq_input: QLineEdit<br>- gain_input: QLineEdit<br>- inp_res_input: QLineEdit<br>- pattern_input: QLineEdit |
| Операции | + check_changes(): void – проверка изменений<br>+ update_antenna_in_db(): void – обновить антенну в БД<br>+ confirm_close(): void – подтвердить закрытие окна |

Таблица 34. Класс: AddAircraftDialog

| Параметр | Значение |
|---|---|
| Комментарий | Диалог добавления нового БПЛА в БД. |
| Атрибуты | - model_input: QLineEdit – модель БПЛА<br>- length_input: QLineEdit – длина фюзеляжа<br>- radius_input: QLineEdit – радиус фюзеляжа<br>- material_input: QLineEdit – материал<br>- conductivity_input: QLineEdit – электропроводность<br>- permittivity_input: QLineEdit – диэлектрическая проницаемость |
| Операции | + add_aircraft_to_db(): void – добавить БПЛА в БД<br>+ confirm_close(): void – подтвердить закрытие окна |

Таблица 35. Класс: SelectAircraftDialog

| Параметр | Значение |
|---|---|
| Комментарий | Диалог выбора БПЛА для редактирования или удаления. |
| Атрибуты | - mode: str – режим (edit / delete)<br>- aircraft_list: QListWidget – список БПЛА |
| Операции | + toggle_action_button(): void – включение/выключение кнопки действия<br>+ open_edit_dialog(): void – открыть диалог редактирования<br>+ delete_aircraft(): void – удалить выбранный БПЛА |

Таблица 36. Класс: EditAircraftDialog

| Параметр | Значение |
|---|---|
| Комментарий | Диалог редактирования БПЛА. |
| Атрибуты | - la_id: int – ID БПЛА<br>- original_data: tuple – исходные данные<br>- model_input: QLineEdit<br>- length_input: QLineEdit<br>- radius_input: QLineEdit<br>- material_input: QLineEdit<br>- conductivity_input: QLineEdit<br>- permittivity_input: QLineEdit |
| Операции | + check_changes(): void – проверка изменений<br>+ update_aircraft_in_db(): void – обновить БПЛА в БД<br>+ confirm_close(): void – подтвердить закрытие окна |

### 4.3. Разработка пользовательского интерфейса программы

После завершения этапов проектирования и реализации функциональных модулей было проведено тестирование программы и подготовлена демонстрационная версия, отражающая основные сценарии её применения. Цель демонстрации – показать, как пользователь взаимодействует с системой и какие возможности предоставляет разработанное программное обеспечение.

Для демонстрации выбраны типовые рабочие сценарии, охватывающие ключевые функции программы: загрузку и выбор моделей летательных аппаратов, выбор и размещение антенн, прогнозирование коэффициента связи, а также сохранение и загрузку проектов. Программа позволяет наглядно визуализировать 3D-модель летательного аппарата и процесс размещения антенн, что существенно облегчает понимание пространственных ограничений и взаимодействий между элементами системы.

Структура верхнего меню программы представлено на рисунке 21:

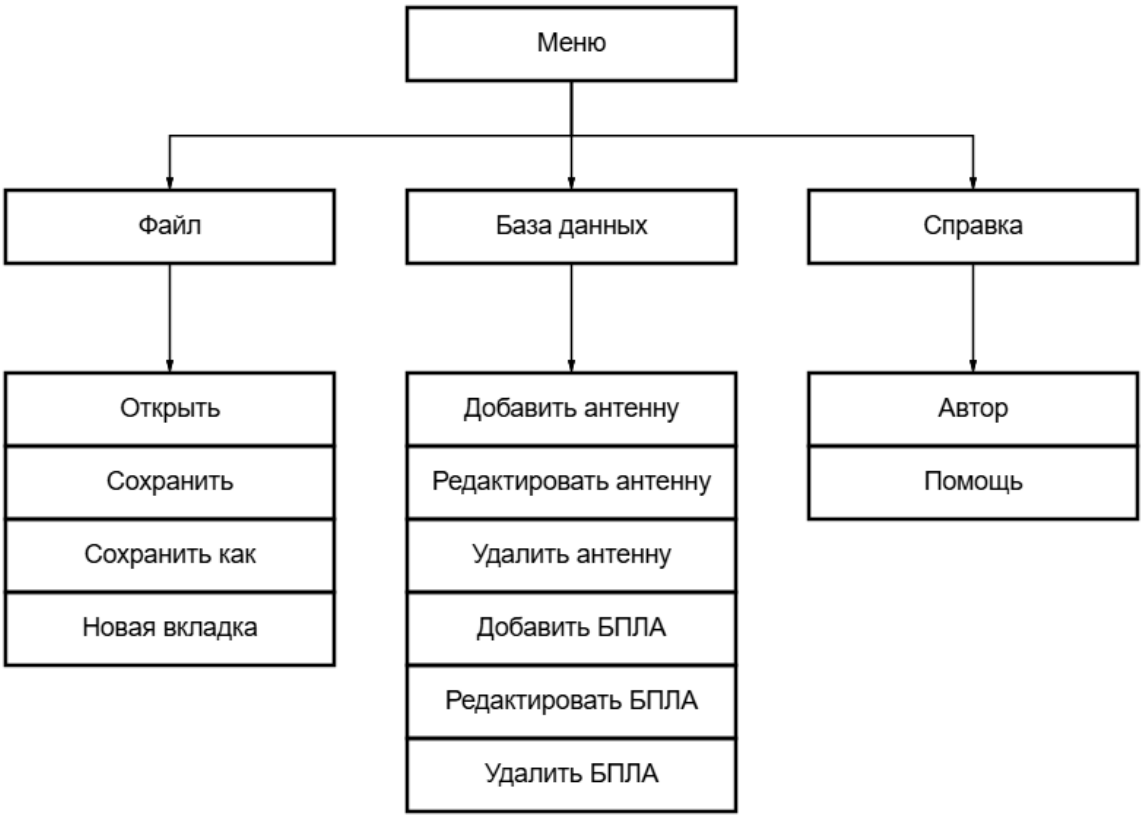

Рис. 22. Структурное представление верхнего меню

Программный вид верхнего меню представлен на рисунках 22-24:

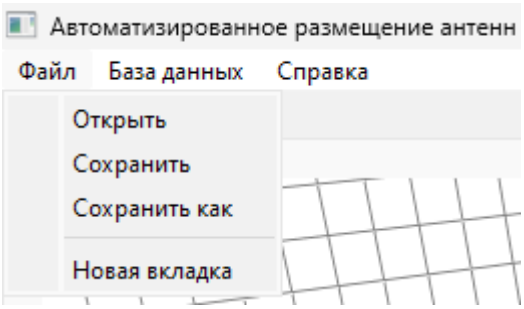


Рис. 23. Подменю «Файл»

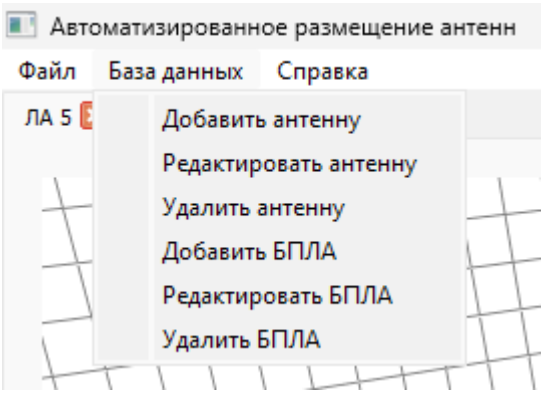


Рис. 24. Подменю «База данных»

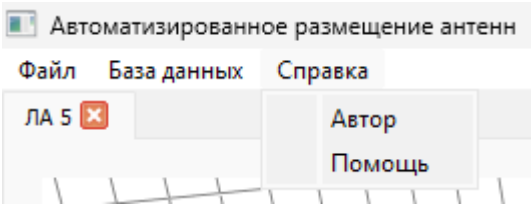


Рис. 25. Подменю «Справка»

Описание пользовательского интерфейса с его визуальным представлением приведено ниже:

При запуске программы первым делом пользователя встречает окно «Загрузка данных». В верхней части окна выводится список летательных аппаратов с информацией об их длине, радиусе и о том, из какого материала сделан фюзеляж, а в нижней части окна приводится список антенн и их характеристик. Все эти данные программа выводит из базы данных. Перед пользователем стоит задача выбрать одну модель летательного аппарата и

хотя бы одну антенну приемник, и одну антенну передатчик из списка доступных. После этого кнопка «Загрузить» становится активной и при нажатии на неё выбранные элементы загружаются в программу для дальнейшего использования. Визуальное представления окна «Загрузка данных» представлено на рисунке 25:

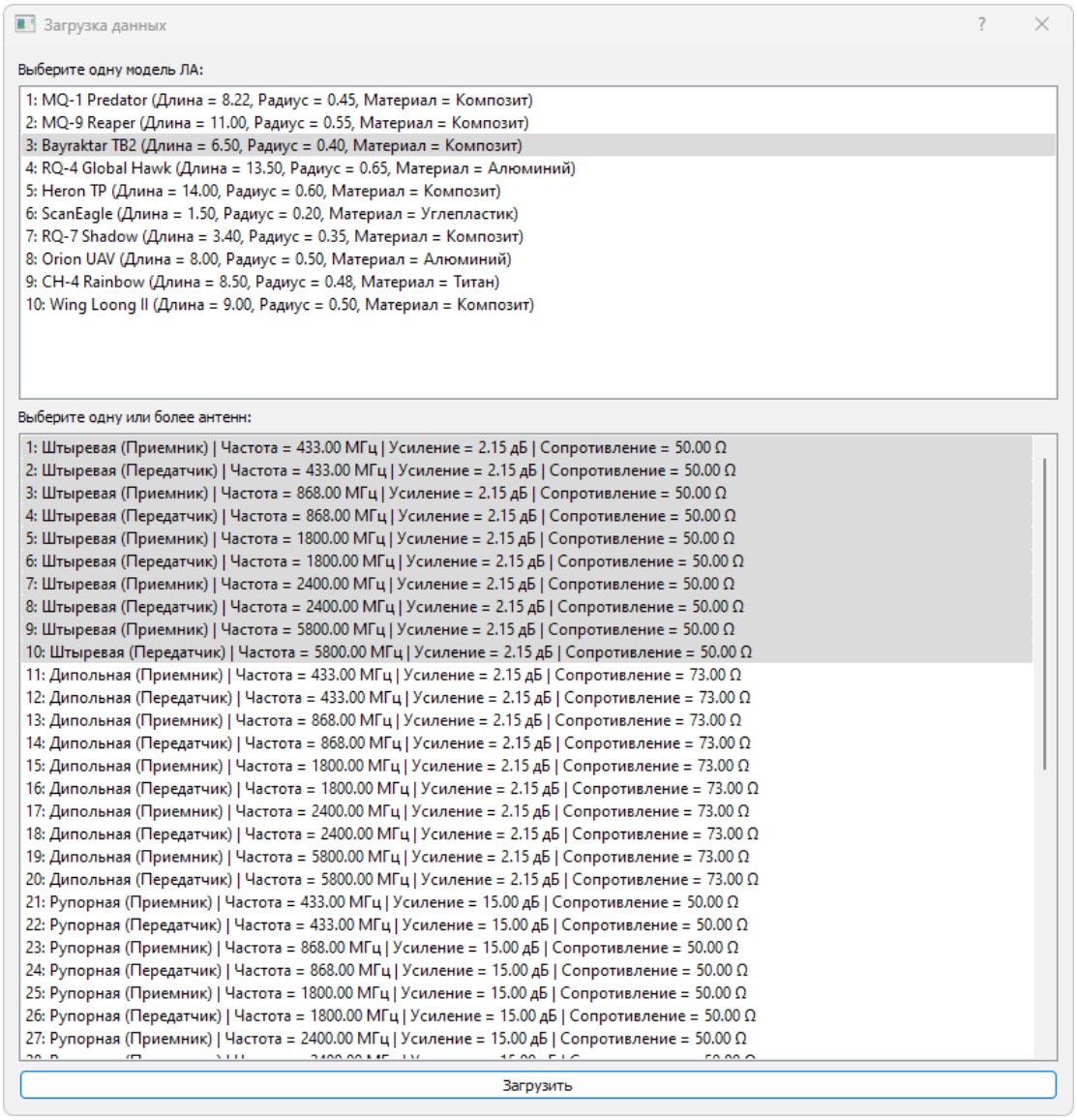


Рис. 26. Окно «Загрузка данных»

После нажатия кнопки «Загрузить» пользователь попадает в главное окно программы и перед ним предстает упрощенная 3D-модель летательного аппарата с сеткой позади неё, список загруженных из предыдущего окна антенн, а также окно «Диаграмма направленности», которая показывает график диаграммы направленности выбранной антенны. В данном окне

пользователь может вращать 3D-модель летательного аппарата, перемещать камеру, двигая компьютерной мышью при зажатых кнопках «CTRL» и «ЛКМ», воспользоваться возможностями верхнего меню, описанными ранее, а также размещать антенны на поверхности летательного аппарата при помощи кнопки «Разместить». Рисунок 26 демонстрирует визуальное представление главного окна программы:

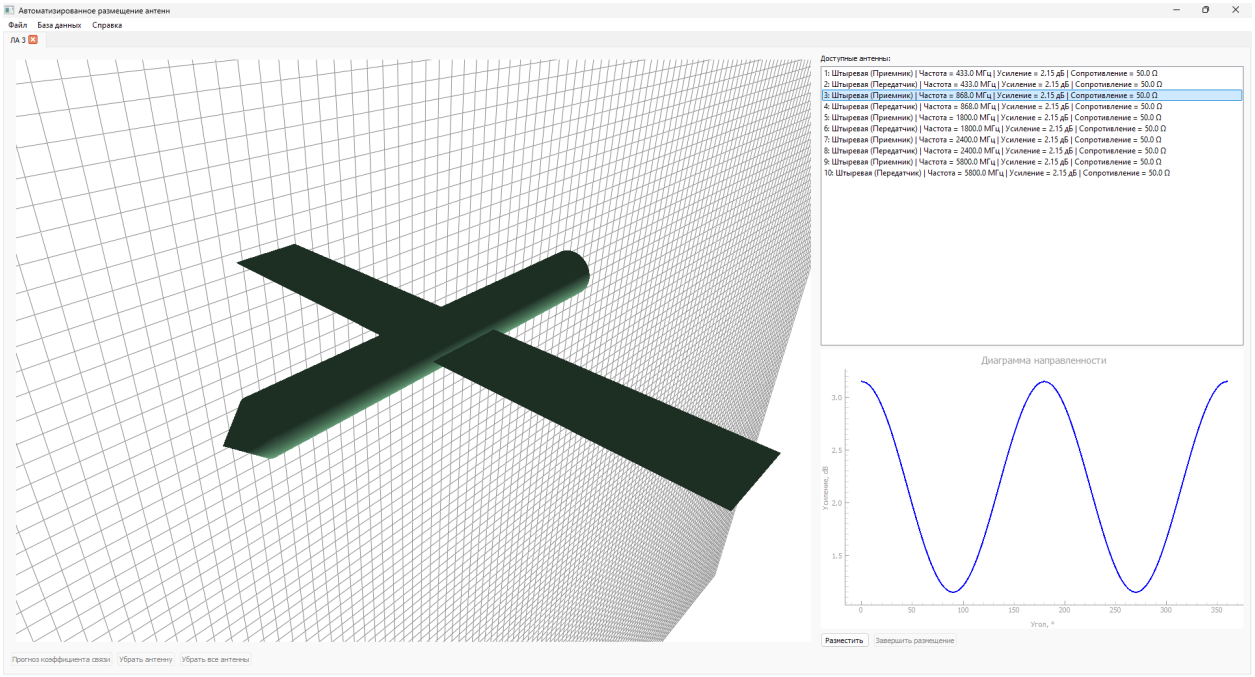


Рис. 27. Главное окно программы

Подробное описание функционала кнопки «Файл» из верхнего меню:

1) Нажатие кнопки «Открыть» влечёт за собой появление окна «Открыть проект», в котором пользователь может открыть ранее созданный файл и продолжить работу над ним в другой вкладке.
2) Нажатие кнопки «Сохранить» сохраняет текущую вкладку проекта на рабочий стол в виде файла со следующим названием и расширением: «ЛА_(ID ЛА).mars».
3) Нажатие кнопки «Сохранить как» даёт пользователю возможность выбрать место для сохранения файла, а также задать произвольное название для файла.
4) Нажатие кнопки «Новая вкладка» приводит к созданию новой вкладки и появлению окна «Загрузка данных», которая встречала

пользователя при запуске.

Визуальное представление действия каждой кнопки представлено на рисунках 27-30:

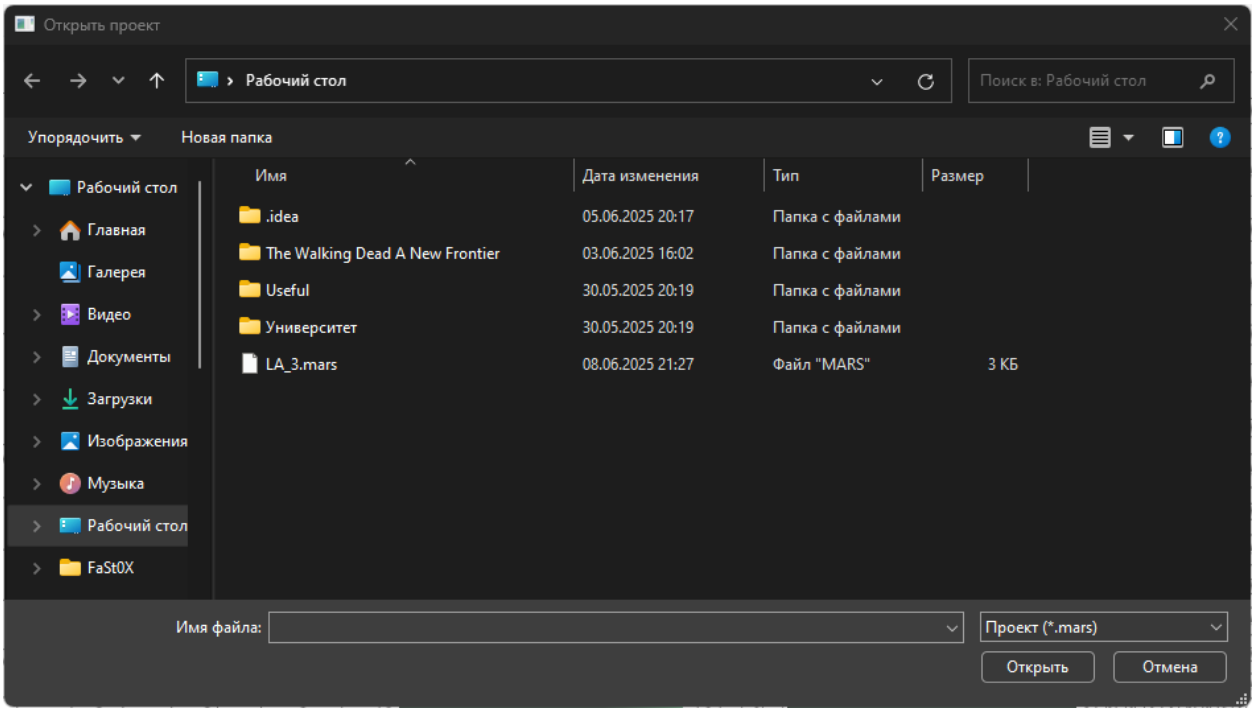


Рис. 28. Окно «Открыть проект»

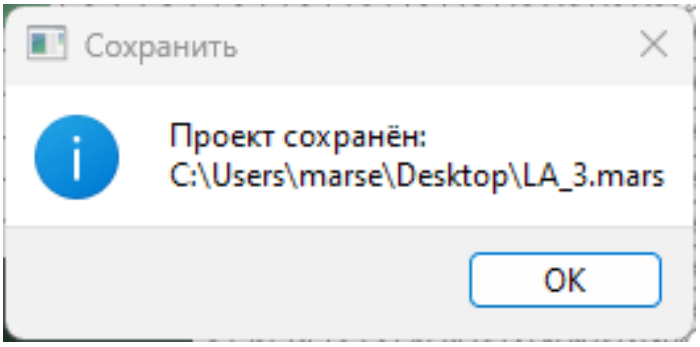


Рис. 29. Результат сохранения программы

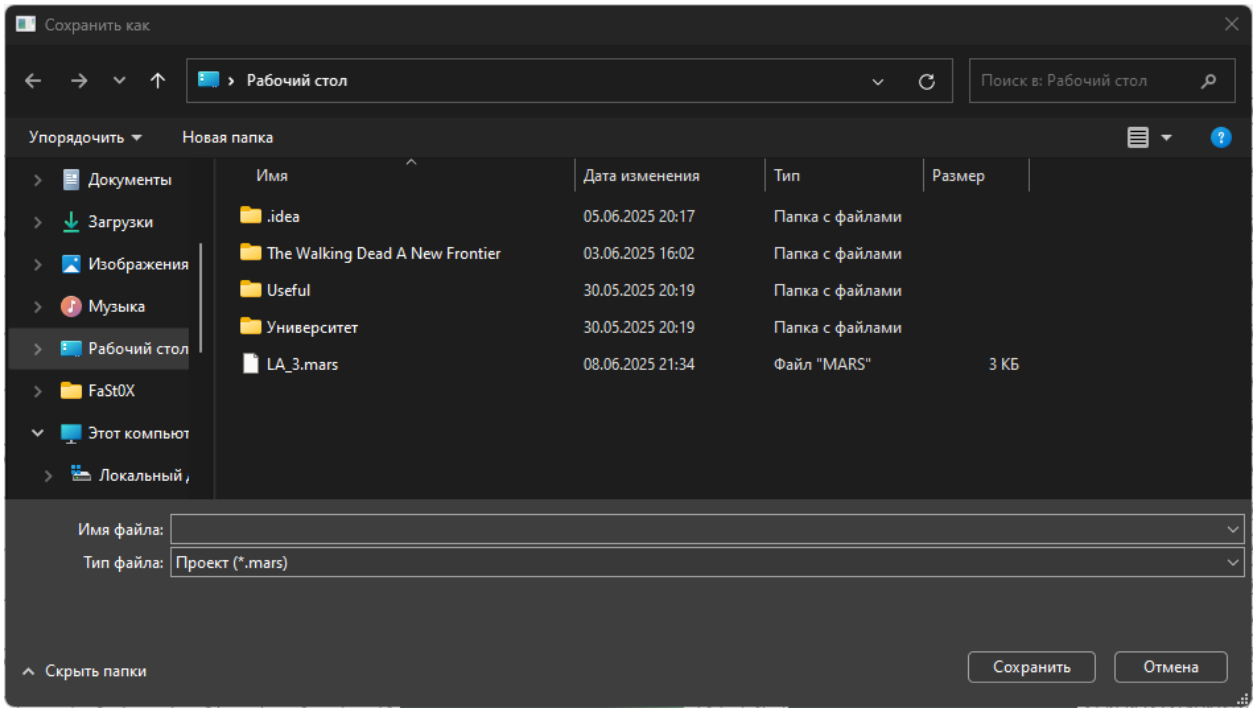


Рис. 30. Окно «Сохранить как»

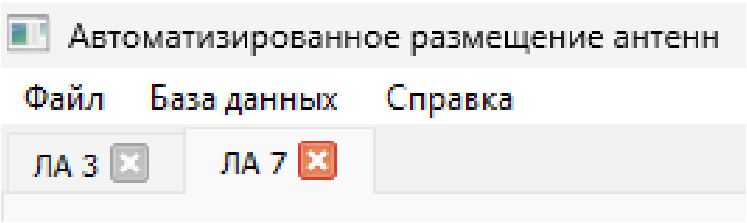


Рис. 31. Результат создания новой вкладки

Подробное описание функционала кнопки «База данных» и «Справка» из верхнего подменю:

1) Нажатие кнопки «Добавить антенну» приводит к появлению окна «Добавить антенну», в котором у пользователя есть возможность добавить антенну, указав её тип (штырьевая, дипольная и т.д.), роль (приемник или передатчик), рабочую частоту, коэффициент усиления, входное сопротивление и ссылку на диаграмму направленности. После ввода всех характеристик, по нажатию кнопки «Добавить» антенна вносится в базу данных и саму программу.
2) Нажатие кнопки «Редактировать антенну» выводит пользователю список антенн и их характеристик. От пользователя требуется выбрать одну антенну из списка для дальнейшего внесения изменений в

характеристики антенны. Делается это нажатием левой кнопкой мыши на одну из антенн и нажатием кнопки «Редактировать». После этого откроется окно с текущими характеристиками выбранной антенны.

3) Нажатие кнопки «Удалить антенну» приводит к появлению того же окна с выбором одной антенны из списка, но уже не для редактирования, а её полного удаления из программы и базы данных.
4) Нажатие кнопки «Добавить БПЛА» влечет за собой появление окна со вводом требуемых характеристик летательного аппарата (модель, длина фюзеляжа, радиус фюзеляжа и т.д.). Нажатие кнопки «Добавить» после ввода всех характеристик приводит к пополнению бозы данных ещё одним летательным аппаратом.
5) Нажатие кнопки «Редактировать БПЛА» выводит пользователю список летательных аппаратов и их характеристик. От пользователя требуется выбрать один летательный аппарат из списка для дальнейшего внесения изменений в его характеристики. Делается это нажатием левой кнопкой мыши на один из летательных аппаратов и нажатием кнопки «Редактировать». После этого откроется окно с текущими характеристиками выбранного летательного аппарата.
6) Нажатие кнопки «Удалить БПЛА» приводит к появлению того же окна с выбором одного летательного аппарата из списка, но уже не для редактирования, а его полного удаления из программы и базы данных.
7) Нажатие кнопки «Автор» выводит окно с информацией об авторе программы и электронной почтой для отправки рекомендаций и жалоб.
8) Нажатие кнопки «Помощь» приводит к появлению окна с краткой инструкцией по эксплуатации программного обеспечения.

Визуальное представление действия каждой кнопки представлено на следующей странице на рисунках 31-38:

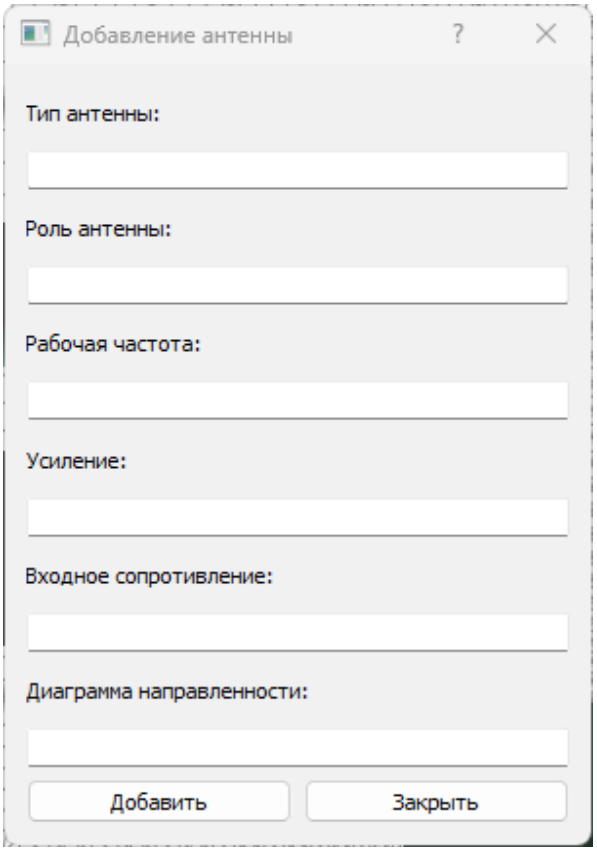


Рис. 32. Окно «Добавление антенны»

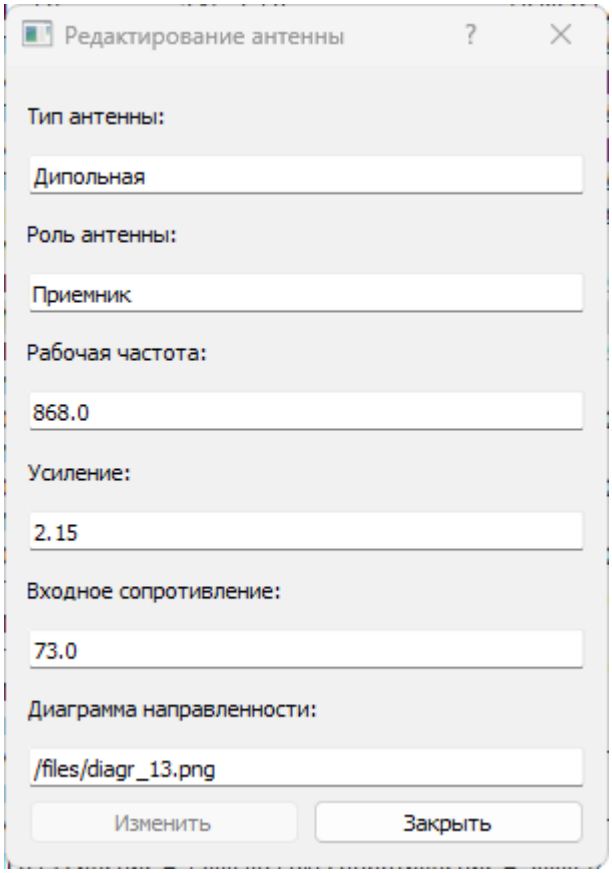


Рис. 33. Окно «Редактирование антенны»

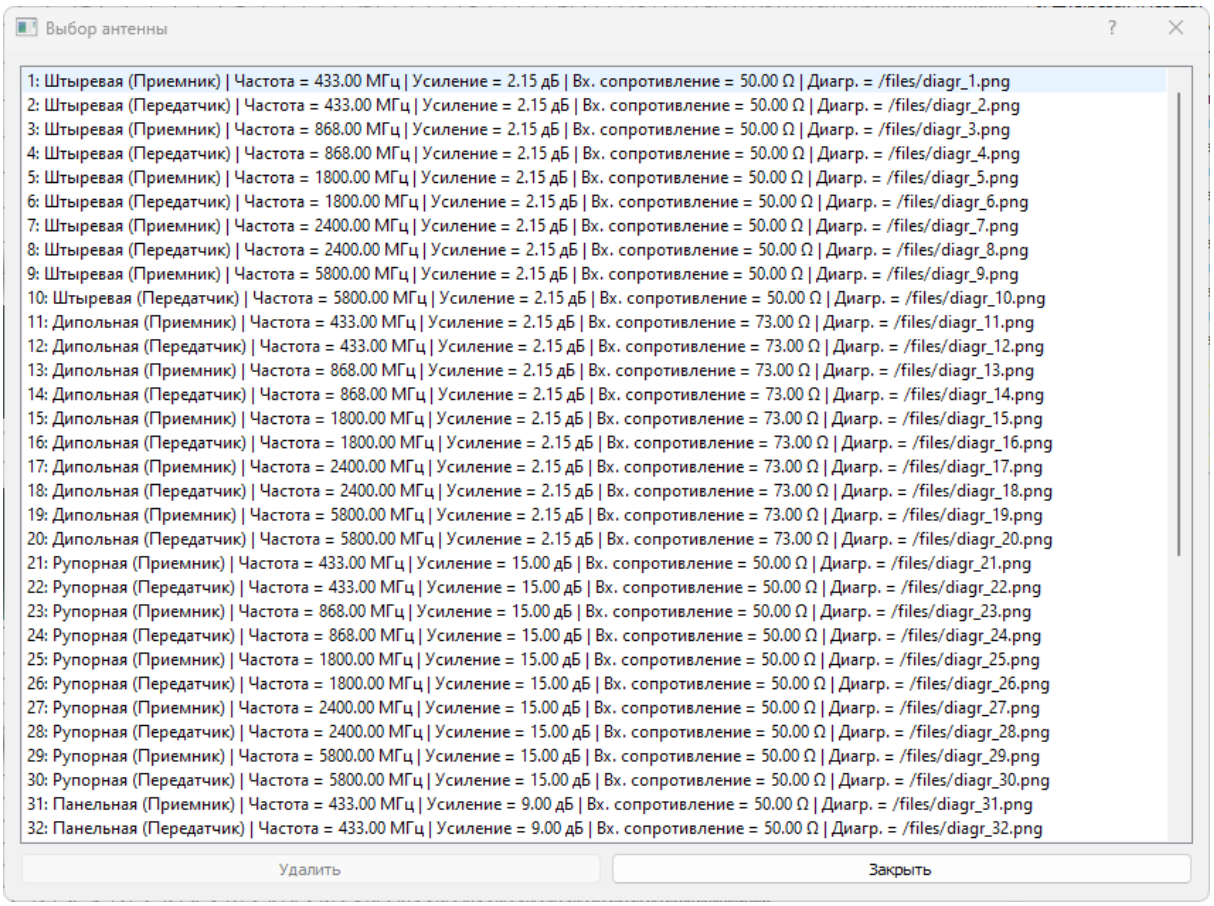


Рис. 34. Окно «Выбор антенны» для удаления

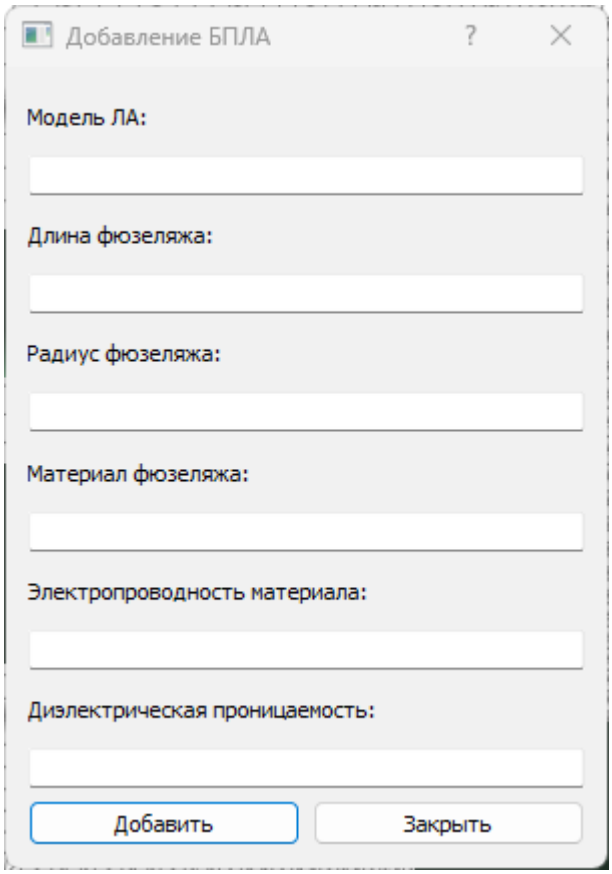


Рис. 35. Окно «Добавление БПЛА»

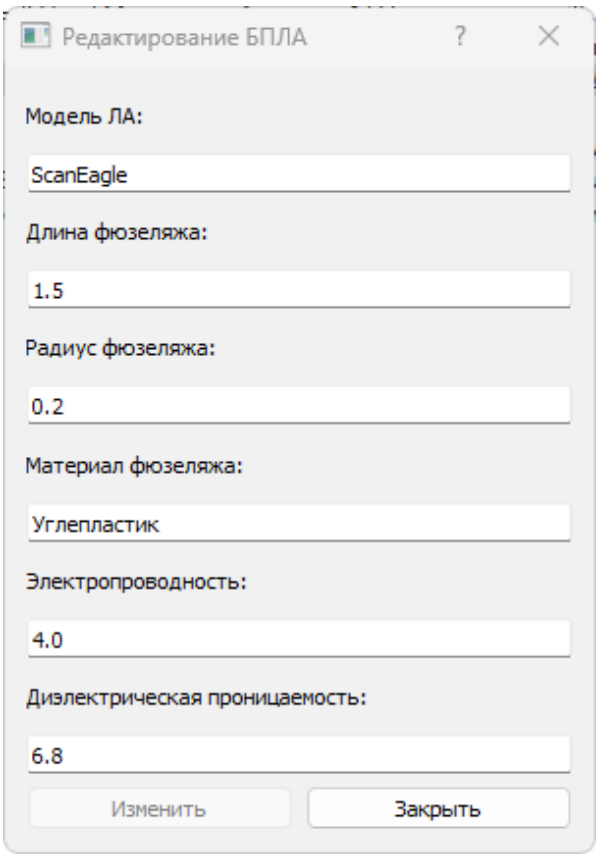


Рис. 36. Окно «Редактировать БПЛА»

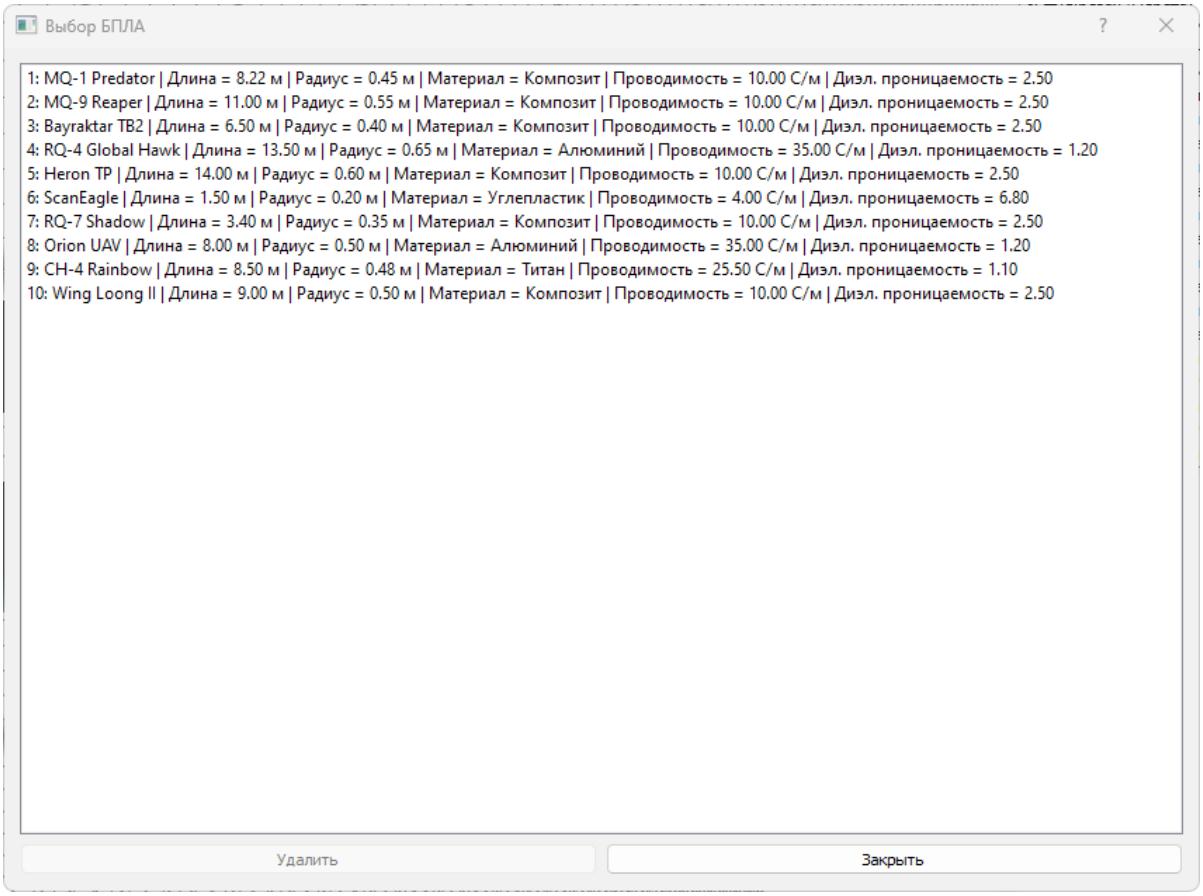


Рис. 37. Окно «Выбор БПЛА» для удаления

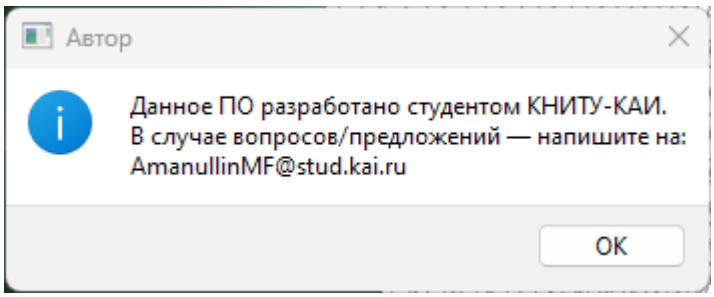


Рис. 38. Окно «Автор»

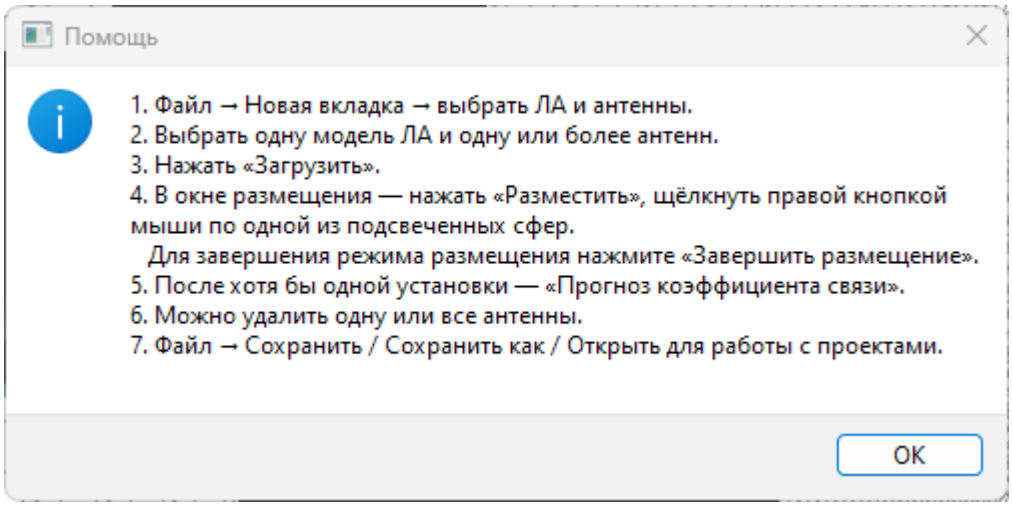


Рис. 39. Окно «Помощь»

Описание основного функционала программного обеспечения:

После загрузки одной модели летательного аппарата и одной или нескольких пар антенн приемник и антенн передатчик, пользователь может приступать к непосредственному размещению антенн на поверхности летательного аппарата. Для этого достаточно нажать кнопку «Разместить», после этого на 3D-модели летательного аппарата высветятся «красные сферы», которые указывают на то, где находятся установочные позиции. Далее пользователю нужно выбрать одну антенну из списка доступных и нажать на желаемую установочную позицию правой кнопкой мыши, тем самым установив выбранную из списка антенну в эту установочную позицию. После того, как антенна была размещена, её установочная позиция исчезает с 3D-модели, а координаты размещенной антенны программа сохраняет.

Визуальное представление процесса размещения демонстрируют рисунок 39 и рисунок 40 на следующей странице:

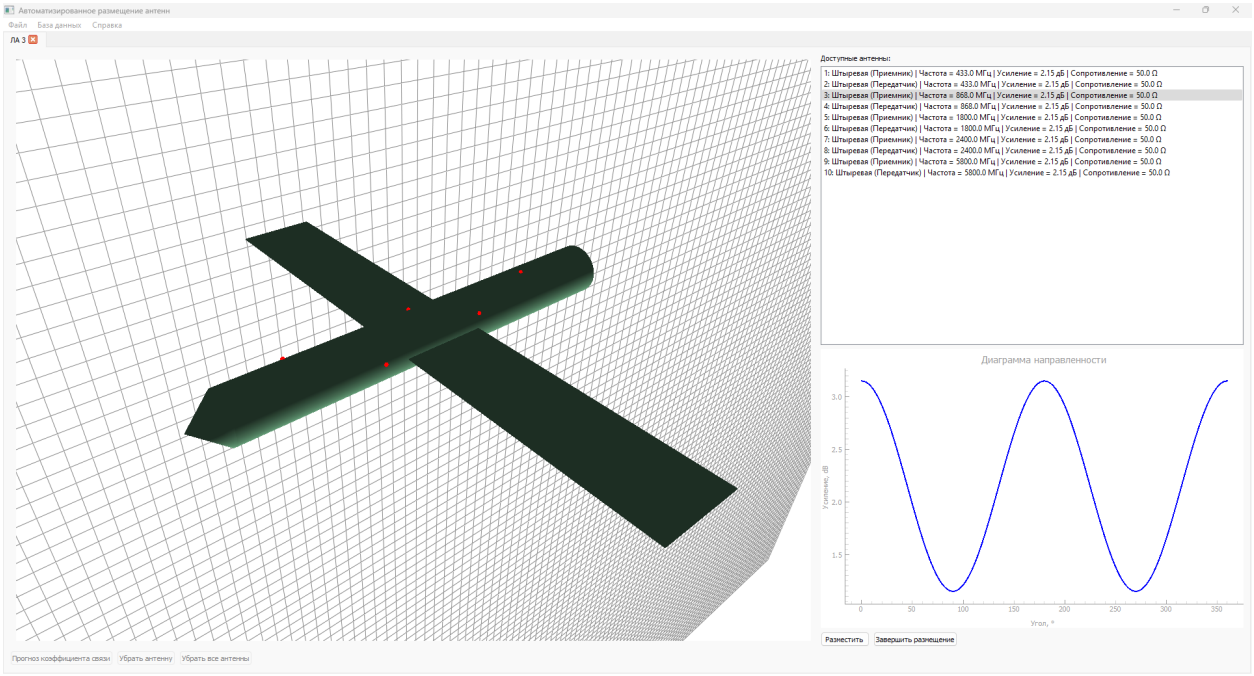


Рис. 40. Главное окно программы во время процесса размещения

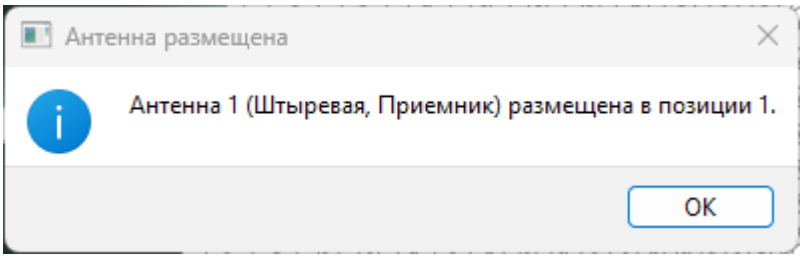


Рис. 41. Сообщение об удачном размещении антенны

После того, как пользователь разместил антенны, он может нажать на кнопку «Завершить размещение». Координаты размещения всех антенн сохраняются в программе. Далее у пользователя есть возможность убрать антенну с установочной позиции путём нажатия кнопки «Убрать антенну», либо убрать все антенны кнопкой «Убрать все антенны». Когда программа убирает одну или все антенны, она восстанавливает установочные позиции, которые были заняты этими антеннами. Если же пользователь окончательно разместил все антенны и желает узнать успешность размещения, он может нажать на кнопку «Прогноз коэффициента связи». Это приведёт к тому, что программы выведет матрицу коэффициентов связи между антеннами.

Визуальное представление описанных выше процессов, а также матрицы коэффициентов связи между антеннами представлены на рисунках 41-44 на следующей странице:

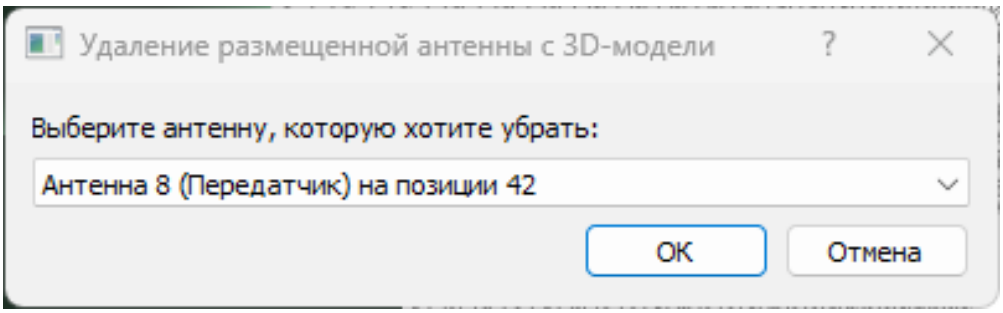


Рис. 42. Окно «Удаление размещенной антенны с 3D-модели»

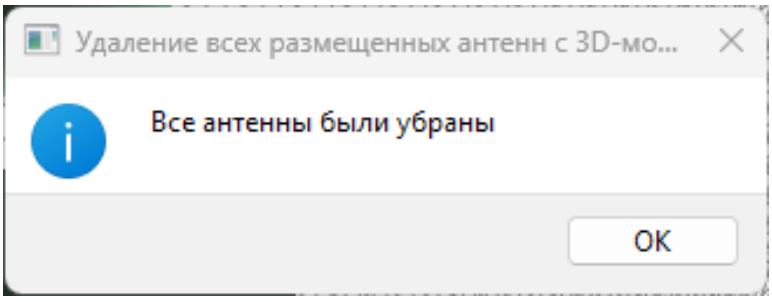


Рис. 43. Результат нажатия кнопки «Убрать все антенны»

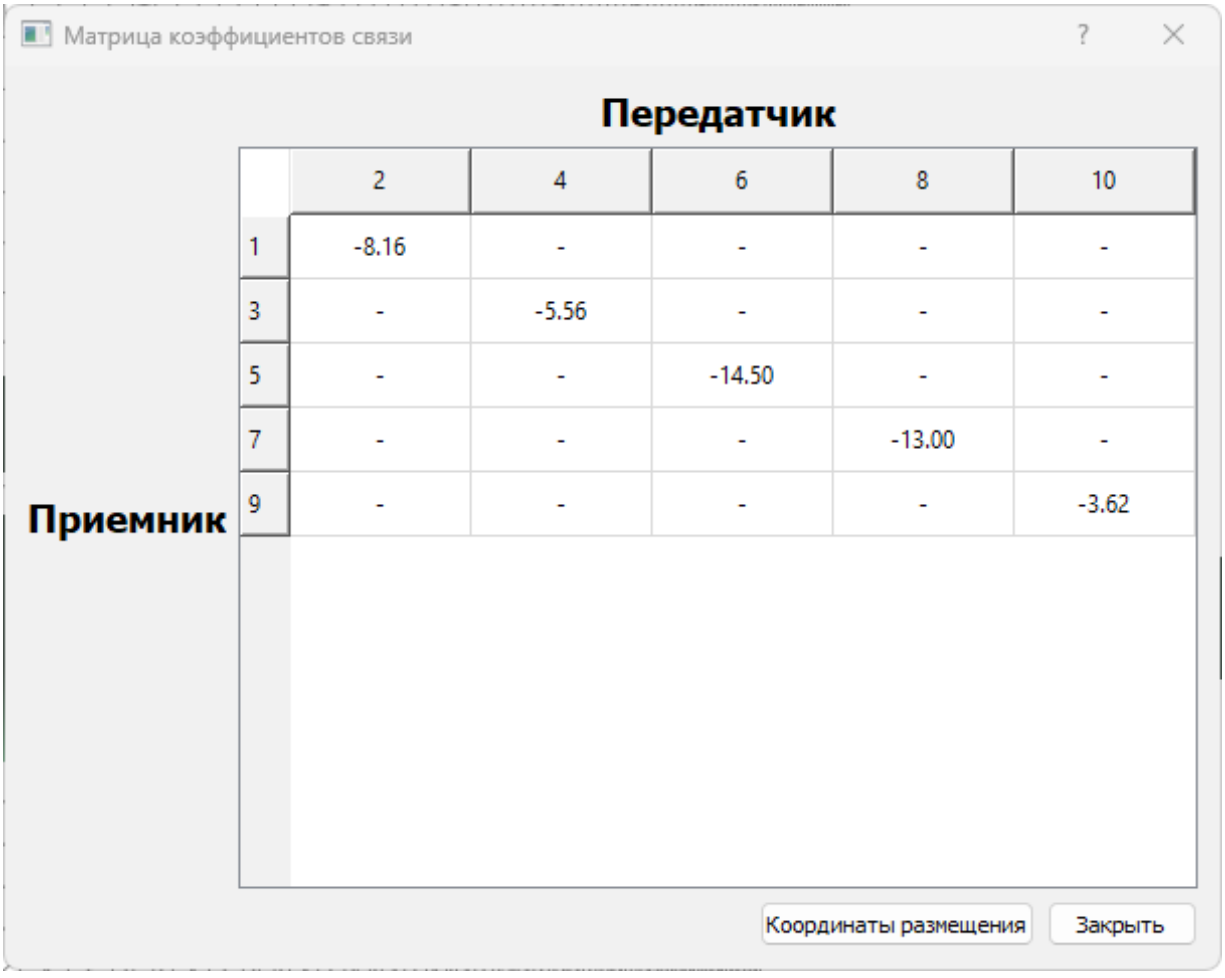


Рис. 44. Матрица коэффициентов связи

Если пользователя устраивают полученные значения коэффициентов связи между антеннами, он может нажать на кнопку «Координаты размещения» и ему откроется окно с координатами размещенных антенн, как на рисунке 44:

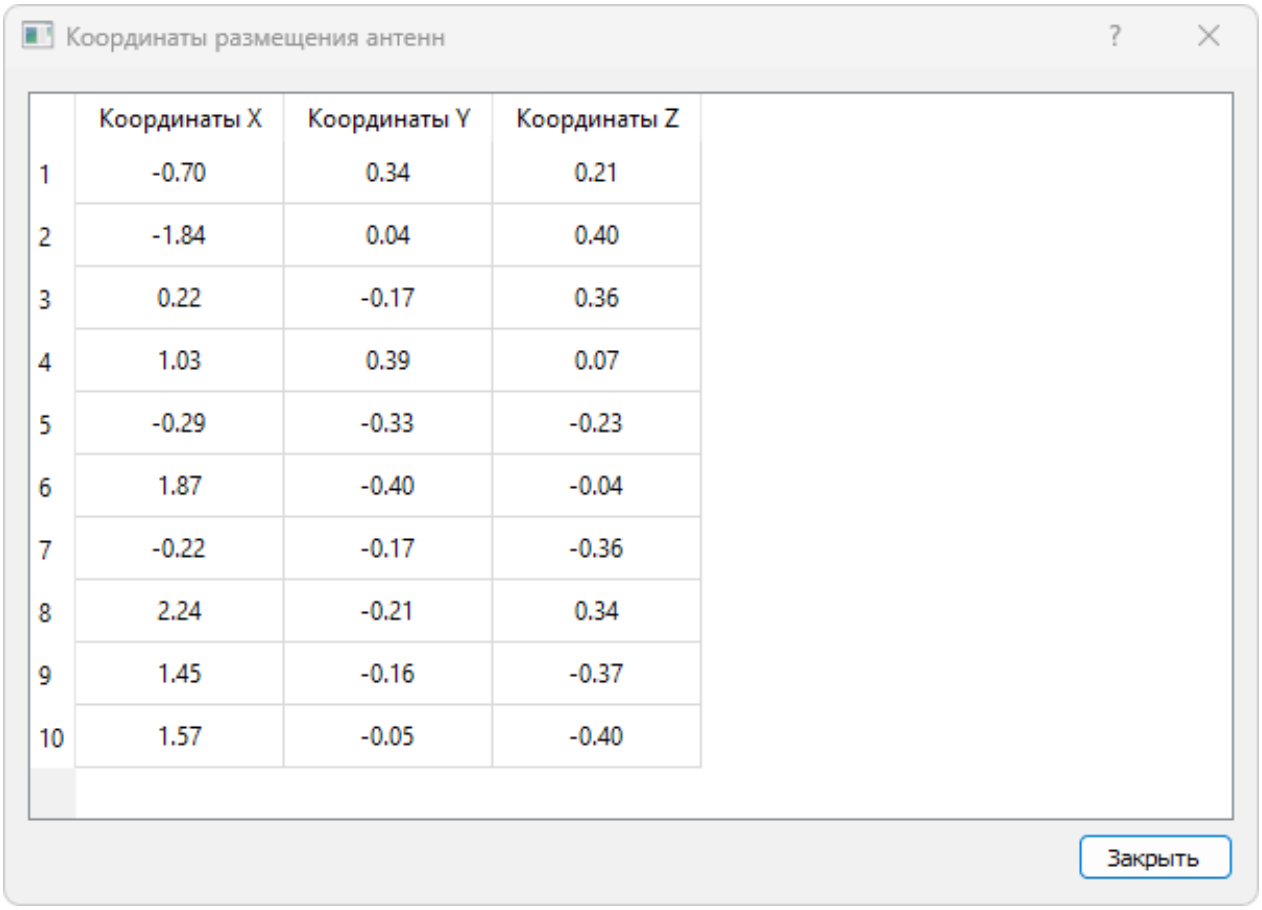


Рис. 45. Окно «Координаты размещения антенн»

## Выводы по главе 4

В данной главе было представлено разработанное программное обеспечение, реализующее автоматизацию размещения антенн на летательном аппарата на основе нейронных сетей. В процессе разработки проведён анализ функциональных требований к системе, спроектирована структура пользовательского интерфейса и реализованы основные модули программы. Для повышения наглядности процесса размещения антенн в программе интегрирована интерактивная трёхмерная модель фюзеляжа с возможностью выбора установочных позиций. Дополнительно реализован модуль прогнозирования коэффициента связи между парами антенн, что позволяет ещё на этапе проектирования получить оценку эффективности выбранного размещения. Архитектура программы предусматривает возможность использования искусственных нейронных сетей для дальнейшего повышения точности прогнозных расчетов. Также обеспечена работа с базой данных антенн и летательных аппаратов, а также сохранение проектов, что делает систему удобной для практического применения инженерами-конструкторами и специалистами по радиоэлектронному оснащению. Таким образом, разработанное программное обеспечение представляет собой удобный инструмент поддержки принятия решений при проектировании компоновочных схем антенн на летательных аппаратах.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была достигнута поставленная цель – разработана формализованная процедура автоматизированного размещения антенн на поверхности беспилотного летательного аппарата с интегрированной прогнозируемой оценкой коэффициента связи на основе нейронных сетей. Все задачи, сформулированные во введении (п. 1.4), полностью выполнены. Ниже кратко обобщены ключевые результаты по каждой из них:

1. Проведён анализ современных тенденций, проблем и требований проектирования антенных систем на летательных аппаратах, включая ограничения по электромагнитной совместимости, габаритам и технологическим допускам. Выявлены недостатки существующих эвристических и CAD-решений и обоснована необходимость применения машинного обучения для ускоренного прогнозирования качества размещения.
2. Разработаны функциональная и поведенческая модели проектной процедуры автоматизации размещения антенн. С помощью IDEF0 описаны функции подготовки данных, обучения и прогнозирования коэффициента связи, а диаграммой IDEF3 формализован хронологический порядок этапов – от генерации конфигураций и моделирования, до выбора оптимального размещения.
3. Сформулирована математическая модель задачи как многомерной регрессии, где на вход подаются параметры размещения антенн (координаты, ориентации, частоты, усиления), геометрия и свойства материалов фюзеляжа, а на выходе – коэффициент связи в дБ. Обоснованы выбор архитектуры MLP-сети с активацией ReLU и функцией потерь MSE.
4. Разработан алгоритм формирования обучающей выборки: сгенерированы 25 конфигураций размещения, проведены электродинамические моделирования, сохранены признаки и

целевые значения. На их основе обучена MLP-модель с тремя скрытыми слоями ($128 \rightarrow 64 \rightarrow 32$ нейрона), оптимизатором Adam и скоростью обучения 0.001. На тестовой выборке достигнута ошибка MSE = 0.021, что подтверждает высокую точность модели.

5. Спроектировано информационное обеспечение процедуры:
   - концептуальная ER-модель базы данных для хранения параметров летательных аппаратов, антенн, вариантов размещения, результатов моделирования и прогнозов;
   - логическая модель с определением первичных и внешних ключей, все таблицы приведены к третьей нормальной форме;
   - физическое проектирование в MySQL (OpenServer), реализованы таблицы, индексы и примеры SQL-запросов.
6. Создано программное обеспечение на языке программирования «Python» с пользовательским интерфейсом и встроенным 3D-виджетом фюзеляжа. Реализованы следующие функции:
   - загрузка модели летательного аппарата и антенн из базы данных;
   - интерактивное размещение антенн по подсвеченным позициям;
   - прогноз коэффициентов связи без запуска моделирования и вывод матрицы значений;
   - контрольный пример прогноза на пяти новых конфигурациях подтвердил корректность работы модуля прогнозирования.

Таким образом, разработанное решение обеспечивает ускоренную и воспроизводимую процедуру размещения антенн на летательном аппарате с учётом электромагнитной совместимости, сокращая время проектирования и снижая риск ошибок по сравнению с классическим ручным подходом.

## СПИСОК ЛИТЕРАТУРЫ


1. Bishop C. M. Neural Networks for Pattern Recognition. – Oxford: Oxford University Press, 1995. – 482 p.
2. CST Studio Suite [Электронный ресурс]. – URL: https://www.3ds.com/products-services/simulia/products/cst-studio-suite/ (дата обращения: 15.05.2025).
3. Goodfellow I., Bengio Y., Courville A. Deep Learning. – Cambridge: MIT Press, 2016. – 775 p.
4. Mitchell M. An Introduction to Genetic Algorithms**.** – Cambridge, MA: MIT Press, 1998. – 221 p.
5. Open Cascade Technology [Электронный ресурс]. – URL: https://www.opencascade.com (дата обращения: 18.05.2025)
6. Skobtsov Yu. A. Genetic Algorithms for CAE Problems.
7. Simon D. Evolutionary Optimization Algorithms. – Hoboken, NJ: Wiley, 2013. – 514 p.
8. Sultanov A. M. Electromagnetic Compatibility in UAV Control Blocks.
9. Szewczyk J. Difficulties with CAD interface understanding // Journal of Engineering Design. – 2003. – Vol. 14. – P. 169-185.
10. Unmanned Aerial Vehicle (UAV) Drones Market Size, Share, and Trends 2025 to 2034 // Precedence Research [Электронный ресурс]. – URL: https://www.precedenceresearch.com/unmanned-aerial-vehicle-drones-market (дата обращения: 14.04.2025).
11. Voronova V. V. CAD for Radioelectronic Equipment.
12. Wang Z.-S. et al. Efficient Chaotic Imperialist Competitive Algorithm // Symmetry. – 2020. – Vol. 12, no. 4.
13. Автоматизированное размещение антенных систем на фюзеляже ЛА // Современные материалы, техника и технология. Курск: Юго-Западный гос. ун-т, 2022. – С. 164–167.
14. Алгоритмы и структуры данных. Учебное пособие / под ред. А. В. Кузнецова. – М.: БХВ-Петербург, 2019. – 416 с.

15. Белоусов А. О. Подходы к обеспечению ЭМС РЭС в составе БПЛА // Системы управления, связи и безопасности. – 2023. – № 3. – С. 134–196.
16. Булдаков В. Н. Методы оптимизации в интеллектуальных системах. – М.: Радио и связь, 2015. – 272 с.
17. Веденькин Д. А., Латышев В. Е., Седельников Ю. Е. Оценка коэффициентов связи антенн для задач ЭМС БПЛА // Журнал радиоэлектроники. – 2014. – № 12.
18. Воронова В. В. Информационные технологии проектирования электронных вычислительных средств. – Казань: КГТУ, 2007. – 207 с.
19. Гаркавый А. Жадные алгоритмы // Algorithmica: науч.-образоват. портал. – 2018. – URL: https://ru.algorithmica.org/cs/combinatorial-optimization/greedy/ (дата обращения: 04.06.2025).
20. Гизатуллин З. М., Гизатуллин Р. М., Нуриев М. Г. Математические модели задач ЭМС // Изв. вузов. Проблемы энергетики. – 2015. – № 1-2. – С. 115-122.
21. Гладков Л. А. Интеллектуальные системы. – Чебоксары: Среда, 2024.
22. Гладков Л. А., Курейчик В. В., Курейчик В. М. Генетические алгоритмы. – 2-е изд., испр. и доп. – М.: Физматлит, 2010. – 368 с.
23. ГОСТ 23611-79. Совместимость радиоэлектронных средств электромагнитная. Термины и определения. – Введ. 1980-07-01. – М.: Изд-во стандартов, 1980.
24. ГОСТ 24375-80. Радиосвязь. Термины и определения. – Введ. 1982-01-01. – М.: Изд-во стандартов, 1987.
25. ГОСТ Р 55898-2013. Технические средства радиосвязи. Взаимные радиопомехи в локальной группировке. Методы расчета. – Введ. 2014-07-01. – М.: Стандартинформ, 2013.
26. ГОСТ Р 57258-2016. Системы беспилотные авиационные. Термины и определения. – Введ. 2017-06-01. – М.: Стандартинформ, 2016.

27. ГОСТ Р 59751-2021. Беспилотные авиационные системы. Требования к летной годности. – Введ. 2022-01-01. – М.: Стандартинформ, 2021.
28. Гришин А. А., Карпенко А. П. Исследование эффективности метода пчелиного роя // Наука и образование. – 2010. – № 8.
29. Дворянкин А. М., Тапе Ж. М. Х. Алгоритм полного перебора для задачи одномерной упаковки объектов // Международный науч.-исслед. журн. – 2021. – № 6(108).
30. Еремеева О. С. Алгоритмы эволюционного моделирования**.** – Новосибирск: Сибирское университетское изд-во, 2017. – 304 с.
31. Жадные алгоритмы // Algorithmica: науч.-образоват. портал. – 2018.
32. З. М. Гизатуллин и др. Математические модели задач ЭМС // Изв. вузов. Проблемы энергетики. – 2015.
33. Карпенко А. П. Современные алгоритмы поисковой оптимизации. – М.: МГТУ им. Баумана, 2014.
34. Кулиев Э. В., Запорожец Д. Ю., Терещенко Д. Ю. Модифицированный алгоритм волчьей стаи // Известия ЮФУ. Техн. науки. – 2019. – № 4. – С. 187-195.
35. Курейчик В. В. Анализ и обзор моделей эволюции // Изв. РАН. Теория и системы управления. – 2007. – № 5. – С. 114–126.
36. Курейчик В. В., Сороколетов П. В. Эволюционные алгоритмы разбиения графов и гиперграфов // Изв. ТРТУ. – 2004. – № 3(38). – С. 23-32.
37. Курейчик Вл. Вл., Курейчик В. В. Биоинспирированный поиск при проектировании и управлении // Известия ЮФУ. Техн. науки. – 2012. – № 11(136). – С. 178-183.
38. Лебедев Б. К. и др. Муравьиный алгоритм планирования СБИС // Вестн. МГТУ им. Баумана. Приборостроение. – 2019. – № 5(128). – С. 49-63.

39. Махонина Ю. В., Неймарк Е. А. Решение задачи раскроя-упаковки с помощью эволюционного алгоритма // Труды НГТУ им. Р. Е. Алексеева. – 2021. – № 2. – С. 24-31.
40. Полупанова Е. Е. и др. Алгоритм последовательной гибридизации для задачи коммивояжера // Известия ЮФУ. Техн. науки. – 2023. – № 3. – С. 108-118.
41. Родзин С. И., Родзина О. Н. Машинное обучение: метаэвристики дифференциально-векторного движения. – Чебоксары: ИД «Среда», 2024. – 137 с.
42. Седельников Ю. Е. Электромагнитная совместимость РЭС. – Казань: Новое знание, 2006. – 304 с.
43. Скобцов Ю. А., Сперанский Д. В. Генетические алгоритмы многокритериальной оптимизации [Электронный ресурс]. – URL: https://intuit.ru/studies/courses/14227/1284/lecture/24176 (дата обращения: 05.05.2025).
44. Тайк А. М., Лупин С. А., Балабаев С. А. Полный перебор для квадратичного назначения // INJOIT. – 2023. – Т. 11, № 7. – С. 60-68.
45. Умные системы: модели и методы метаэвристической оптимизации. – Чебоксары: Среда, 2024.
46. Хайнеман Д., Поллис Г., Селков С. Алгоритмы: справочник разработчика. – М.: ДМК Пресс; O'Reilly, 2017. – 704 с.
47. Хашпер Б. Л., Кантор О. Г. Методы случайного поиска и машинного обучения для оптимизации // Информ. и матем. технологии. – 2023. – № 3(31). – С. 15-26.
48. Частикова В. А., Жерлицын С. А. Исследование алгоритма серых волков // Науч. труды КубГТУ. – 2016. – № 16. – С. 136–142.
49. Штовба С. Д. Муравьиные алгоритмы // Exponenta Pro. – 2003. – № 4. – С. 70-75.

50. Шулаева Е. А., Маштанов Н. М., Иванов А. Н. Многокритериальная оптимизация методом имитации отжига // Электротехн. и информ. комплексы и системы. – 2020. – Т. 16, № 1. – С. 89-96.

## ПРИЛОЖЕНИЕ А. Глоссарий

1. Антенна – устройство, преобразующее электромагнитную энергию в радиоволны и наоборот, обеспечивающее приём и/или передачу сигнала в радиочастотном диапазоне.
2. Беспилотный летательный аппарат – воздушное средство без экипажа на борту, управляемое автоматически или дистанционно.
3. Коэффициент связи – числовой показатель степени электромагнитного взаимодействия (взаимных помех) между антеннами, выражаемый в децибелах.
4. Электромагнитная совместимость – способность электронных устройств функционировать без взаимных помех, а также без ухудшения качества связи.
5. Компоновочная схема – итоговое инженерное решение по размещению антенн на летательном аппарате, представленное в виде графической и текстовой документации.
6. Диаграмма направленности антенны – график зависимости мощности излучаемого или принимаемого сигнала от направления в пространстве.
7. Смоделированные данные – результаты численного моделирования, используемые в качестве истинных значений для формирования обучающей выборки.
8. Целевая переменная – выходной параметр модели, подлежащий прогнозированию; в данной работе – это коэффициент связи между антеннами.
9. Конфигурация размещения – конкретный набор координат и ориентаций антенн на летательном аппарате, рассматриваемый как потенциальное инженерное решение.
10. Прогноз коэффициента связи – значение коэффициента связи между антеннами, вычисленное нейронной сетью без необходимости проведения физического моделирования.

11. Фюзеляж – корпус летательного аппарата, на поверхности которого размещаются антенны.
12. CAD/CAE-среда – инженерная система автоматизированного проектирования и анализа, используемая для трёхмерного моделирования и компоновки оборудования.

## Приложение Б. Образцы оформления компоновочной схемы

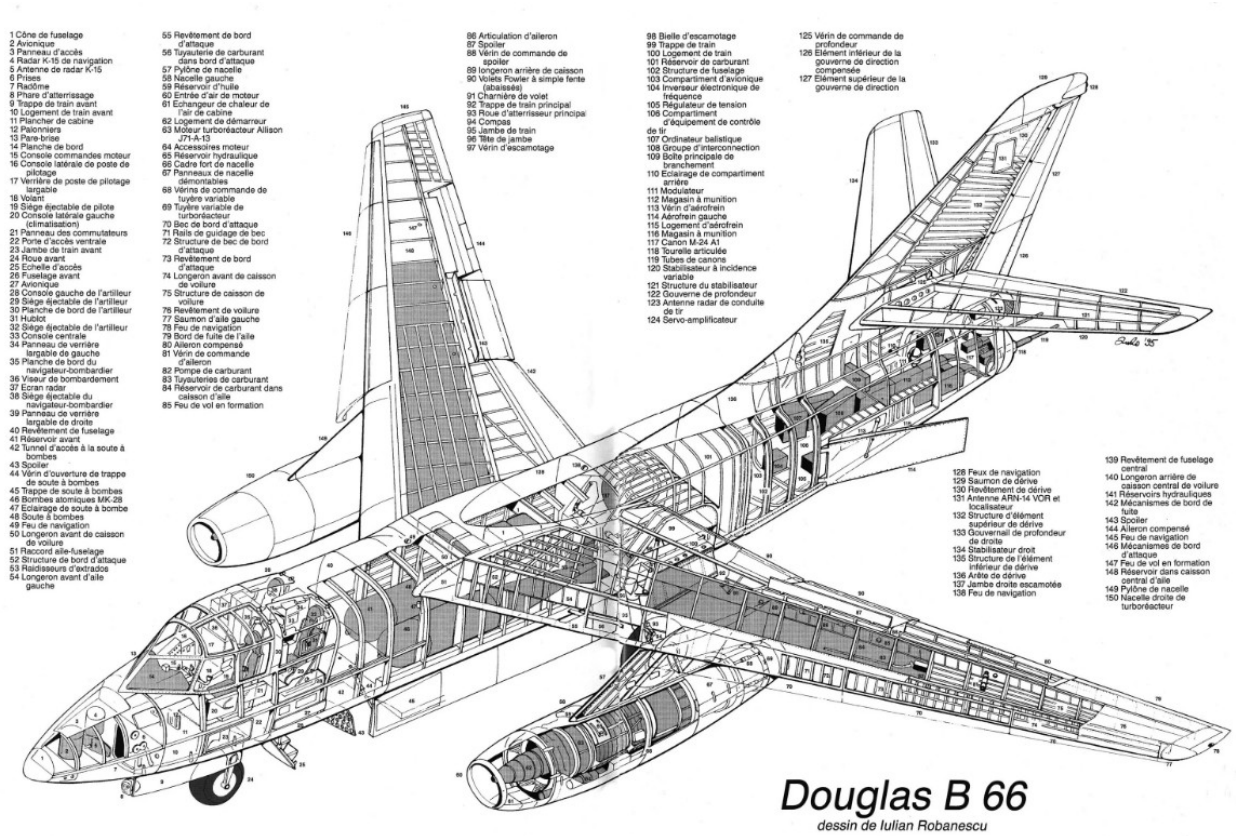


Рис. 1. Первый пример компоновочной схемы летательного аппарата

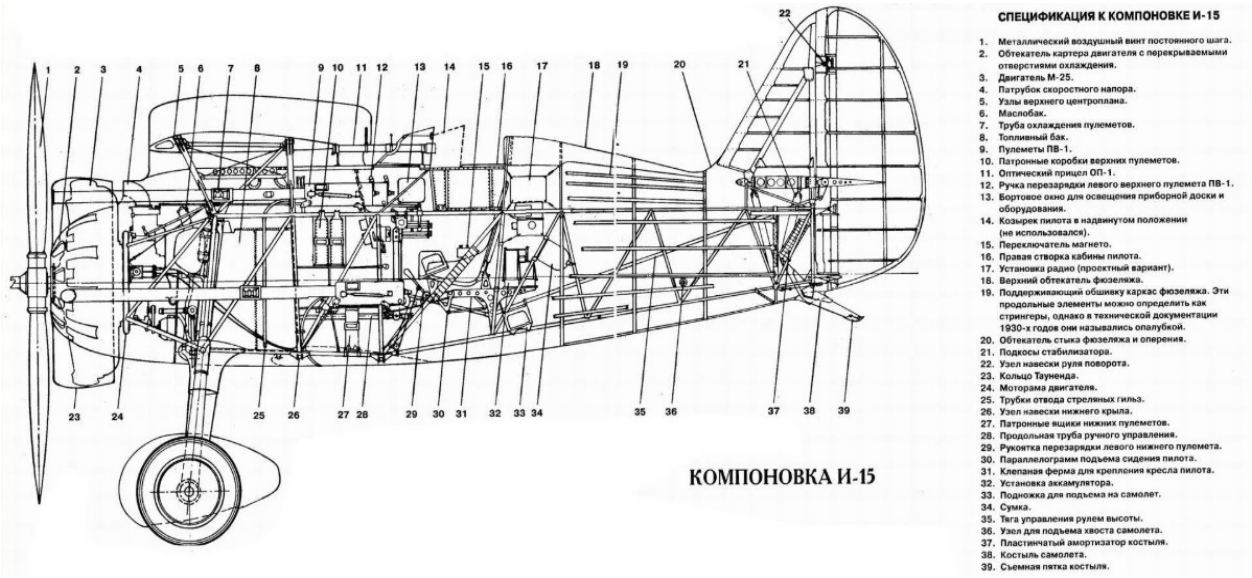


Рис. 2. Второй пример компоновочной схемы летательного аппарата

## ПРИЛОЖЕНИЕ В. Листинг программы.

```
import sys
import os
import json
import random
import numpy as np
import pymysql
import math
import pyqtgraph as pg
import pyqtgraph.opengl as gl
from PyQt5 import QtWidgets, QtGui, QtCore
from decimal import Decimal
from PyQt5.QtGui import QVector3D
from sklearn.neural_network import MLPRegressor

def predict_ks(freq_rx, freq_tx, dist_m, max_dist=None):
    if freq_rx != freq_tx:
        return None
    if dist_m >= 4.0:
        return -20.0
    elif dist_m >= 3.0:
        return round(random.uniform(-20.0, -15.0), 2)
    elif dist_m >= 2.0:
        return round(random.uniform(-15.0, -10.0), 2)
    elif dist_m >= 1.0:
        return round(random.uniform(-10.0,  -5.0), 2)
    else:
        return round(random.uniform(-5.0,   0.0), 2)

DB_HOST = 'localhost'
DB_PORT = 3306
DB_USER = 'root'
DB_PASSWORD = ''
DB_NAME = 'vkr'

def get_aircrafts():
    conn = pymysql.connect(host=DB_HOST, port=DB_PORT, user=DB_USER,
                           password=DB_PASSWORD, db=DB_NAME, charset='utf8')
    cursor = conn.cursor()
    cursor.execute("""
        SELECT `ID ЛА`, `Модель ЛА`, `Длина фюзеляжа`, `Радиус фюзеляжа`,
            `Материал фюзеляжа`, `Электропроводность материала`, `Диэлектрическая проницаемость`
        FROM `Летательный аппарат`;
    """)
    rows = cursor.fetchall()
    conn.close()
    return rows

def get_antennas_full():
    conn = pymysql.connect(host=DB_HOST, port=DB_PORT, user=DB_USER,
                           password=DB_PASSWORD, db=DB_NAME, charset='utf8')
    cursor = conn.cursor()
    cursor.execute("""
        SELECT `ID антенны`, `Тип антенны`, `Роль антенны`, `Рабочая частота`,
            `Усиление`, `Входное сопротивление`, `Диаграмма направленности`
        FROM `Антенна`;
    """)
    rows = cursor.fetchall()
    conn.close()
    return rows
```

```
class AddAntennaDialog(QtWidgets.QDialog):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setWindowTitle("Добавление антенны")
        self.setFixedSize(300, 400)

        self.is_modified = False

        layout = QtWidgets.QVBoxLayout(self)
        self.type_input     = QtWidgets.QLineEdit()
        self.role_input     = QtWidgets.QLineEdit()
        self.freq_input     = QtWidgets.QLineEdit()
        self.gain_input     = QtWidgets.QLineEdit()
        self.inp_res_input  = QtWidgets.QLineEdit()
        self.pattern_input  = QtWidgets.QLineEdit()

        for w in (self.type_input, self.role_input, self.freq_input,
                  self.gain_input, self.inp_res_input, self.pattern_input):
            w.textChanged.connect(self.mark_modified)

        layout.addWidget(QtWidgets.QLabel("Тип антенны:"))
        layout.addWidget(self.type_input)
        layout.addWidget(QtWidgets.QLabel("Роль антенны:"))
        layout.addWidget(self.role_input)
        layout.addWidget(QtWidgets.QLabel("Рабочая частота:"))
        layout.addWidget(self.freq_input)
        layout.addWidget(QtWidgets.QLabel("Усиление:"))
        layout.addWidget(self.gain_input)
        layout.addWidget(QtWidgets.QLabel("Входное сопротивление:"))
        layout.addWidget(self.inp_res_input)
        layout.addWidget(QtWidgets.QLabel("Диаграмма направленности:"))
        layout.addWidget(self.pattern_input)

        btn_layout = QtWidgets.QHBoxLayout()
        self.add_button = QtWidgets.QPushButton("Добавить")
        self.add_button.clicked.connect(self.add_antenna_to_db)
        self.cancel_button = QtWidgets.QPushButton("Закрыть")
        self.cancel_button.clicked.connect(self.confirm_close)
        btn_layout.addWidget(self.add_button)
        btn_layout.addWidget(self.cancel_button)
        layout.addLayout(btn_layout)

    def mark_modified(self, _):
        self.is_modified = True

    def add_antenna_to_db(self):
        type_ = self.type_input.text()
        role = self.role_input.text()
        freq = self.freq_input.text()
        gain = self.gain_input.text()
        inp_res = self.inp_res_input.text()
        pattern = self.pattern_input.text()

        if not all([type_, role, freq, gain, inp_res, pattern]):
            QtWidgets.QMessageBox.warning(self, "Ошибка", "Все поля должны быть заполнены.")
            return

        try:
            conn = pymysql.connect(host=DB_HOST, port=DB_PORT, user=DB_USER,
                                   password=DB_PASSWORD, db=DB_NAME, charset='utf8')
            cursor = conn.cursor()
```

```
            cursor.execute("""
                INSERT INTO `Антенна` (`Тип антенны`, `Роль антенны`, `Рабочая частота`,
                                       `Усиление`, `Входное сопротивление`, `Диаграмма направленности`)
                VALUES (%s, %s, %s, %s, %s, %s);
            """, (type_, role, float(freq), float(gain), float(inp_res), pattern))
            conn.commit()
            conn.close()
            QtWidgets.QMessageBox.information(self, "Успех", "Антенна добавлена.")
            self.close()
        except pymysql.Error as e:
            QtWidgets.QMessageBox.critical(self, "Ошибка", f"Ошибка базы данных: {e}")

    def confirm_close(self):
        if not self.is_modified:
            self.close()
            return
        msg = QtWidgets.QMessageBox(self)
        msg.setWindowTitle("Подтверждение")
        msg.setText("Изменения не будут сохранены. Закрыть окно?")
        yes = msg.addButton("Да",  QtWidgets.QMessageBox.YesRole)
        no  = msg.addButton("Нет", QtWidgets.QMessageBox.NoRole)
        msg.exec_()
        if msg.clickedButton() == yes:
            self.close()

class SelectAntennaDialog(QtWidgets.QDialog):
    def __init__(self, parent=None, mode="edit"):
        super().__init__(parent)
        self.setWindowTitle("Выбор антенны")
        self.setFixedSize(850, 600)
        self.mode = mode
        layout = QtWidgets.QVBoxLayout(self)

        self.ant_list = QtWidgets.QListWidget()
        self.ant_list.setSelectionMode(QtWidgets.QAbstractItemView.SingleSelection)
        layout.addWidget(self.ant_list)

        antennas = get_antennas_full()
        for ant_id, ant_type, ant_role, freq, gain, inp_res, pattern in antennas:
            text = (f"{ant_id}: {ant_type} ({ant_role}) | "
                    f"Частота = {freq:.2f} МГц | Усиление = {gain:.2f} дБ | "
                    f"Вх. сопротивление = {inp_res:.2f} Ω | Диагр. = {pattern}")
            item = QtWidgets.QListWidgetItem(text)
            item.setData(QtCore.Qt.UserRole, (ant_id, ant_type, ant_role, freq, gain, inp_res, pattern))
            self.ant_list.addItem(item)

        btn_layout = QtWidgets.QHBoxLayout()
        if self.mode == "edit":
            self.action_button = QtWidgets.QPushButton("Редактировать")
            self.action_button.clicked.connect(self.open_edit_dialog)
        elif self.mode == "delete":
            self.action_button = QtWidgets.QPushButton("Удалить")
            self.action_button.clicked.connect(self.delete_antenna)
        self.close_button = QtWidgets.QPushButton("Закрыть")
        self.close_button.clicked.connect(self.close)
        btn_layout.addWidget(self.action_button)
        btn_layout.addWidget(self.close_button)
        layout.addLayout(btn_layout)

        self.ant_list.itemSelectionChanged.connect(self.toggle_action_button)
        self.action_button.setEnabled(False)
```

```
    def toggle_action_button(self):
        self.action_button.setEnabled(len(self.ant_list.selectedItems()) == 1)

    def open_edit_dialog(self):
        selected = self.ant_list.selectedItems()[0]
        ant_data = selected.data(QtCore.Qt.UserRole)
        edit_dialog = EditAntennaDialog(ant_data, self)
        edit_dialog.exec_()

    def delete_antenna(self):
        selected = self.ant_list.selectedItems()[0]
        ant_id = selected.data(QtCore.Qt.UserRole)[0]

        msg = QtWidgets.QMessageBox(self)
        msg.setWindowTitle("Подтверждение")
        msg.setText(f"Вы уверены, что хотите удалить антенну {ant_id}?")
        yes = msg.addButton("Да", QtWidgets.QMessageBox.YesRole)
        no = msg.addButton("Нет", QtWidgets.QMessageBox.NoRole)
        msg.exec_()

        if msg.clickedButton() == yes:
            try:
                conn = pymysql.connect(host=DB_HOST, port=DB_PORT, user=DB_USER,
                                       password=DB_PASSWORD, db=DB_NAME, charset='utf8')
                cursor = conn.cursor()
                cursor.execute("DELETE FROM `Антенна` WHERE `ID антенны` = %s", (ant_id,))
                conn.commit()
                conn.close()
                QtWidgets.QMessageBox.information(self, "Успех", "Антенна удалена.")
                self.ant_list.takeItem(self.ant_list.row(selected))
            except pymysql.Error as e:
                QtWidgets.QMessageBox.critical(self, "Ошибка", f"Ошибка базы данных: {e}")


class EditAntennaDialog(QtWidgets.QDialog):
    def __init__(self, ant_data, parent=None):
        super().__init__(parent)
        self.setWindowTitle("Редактирование антенны")
        self.setFixedSize(300, 400)
        self.ant_id       = ant_data[0]
        self.original_data = ant_data
        self.is_modified   = False

        layout = QtWidgets.QVBoxLayout(self)
        self.type_input    = QtWidgets.QLineEdit(ant_data[1])
        self.role_input    = QtWidgets.QLineEdit(ant_data[2])
        self.freq_input    = QtWidgets.QLineEdit(str(ant_data[3]))
        self.gain_input    = QtWidgets.QLineEdit(str(ant_data[4]))
        self.inp_res_input = QtWidgets.QLineEdit(str(ant_data[5]))
        self.pattern_input = QtWidgets.QLineEdit(ant_data[6])

        for w in (self.type_input, self.role_input, self.freq_input,
                  self.gain_input, self.inp_res_input, self.pattern_input):
            w.textChanged.connect(self.mark_modified)
            w.textChanged.connect(self.check_changes)

        layout.addWidget(QtWidgets.QLabel("Тип антенны:"))
        layout.addWidget(self.type_input)
        layout.addWidget(QtWidgets.QLabel("Роль антенны:"))
        layout.addWidget(self.role_input)
        layout.addWidget(QtWidgets.QLabel("Рабочая частота:"))
        layout.addWidget(self.freq_input)
```

```
        layout.addWidget(QtWidgets.QLabel("Усиление:"))
        layout.addWidget(self.gain_input)
        layout.addWidget(QtWidgets.QLabel("Входное сопротивление:"))
        layout.addWidget(self.inp_res_input)
        layout.addWidget(QtWidgets.QLabel("Диаграмма направленности:"))
        layout.addWidget(self.pattern_input)

        btn_layout = QtWidgets.QHBoxLayout()
        self.update_button = QtWidgets.QPushButton("Изменить")
        self.update_button.clicked.connect(self.update_antenna_in_db)
        self.cancel_button = QtWidgets.QPushButton("Закрыть")
        self.cancel_button.clicked.connect(self.confirm_close)
        btn_layout.addWidget(self.update_button)
        btn_layout.addWidget(self.cancel_button)
        layout.addLayout(btn_layout)

        self.update_button.setEnabled(False)

        for input_field in [self.type_input, self.role_input, self.freq_input, self.gain_input, self.inp_res_input,
                            self.pattern_input]:
            input_field.textChanged.connect(self.check_changes)
        self.update_button.setEnabled(False)

    def mark_modified(self, _):
        self.is_modified = True

    def check_changes(self):
        od = self.original_data
        has = (
            self.type_input.text()    != str(od[1]) or
            self.role_input.text()    != str(od[2]) or
            self.freq_input.text()    != str(od[3]) or
            self.gain_input.text()    != str(od[4]) or
            self.inp_res_input.text() != str(od[5]) or
            self.pattern_input.text() != str(od[6])
        )
        self.update_button.setEnabled(has)

    def update_antenna_in_db(self):
        if not all([self.type_input.text(), self.role_input.text(), self.freq_input.text(), self.gain_input.text(),
                    self.inp_res_input.text(), self.pattern_input.text()]):
            QtWidgets.QMessageBox.warning(self, "Ошибка", "Все поля должны быть заполнены.")
            return

        try:
            conn = pymysql.connect(host=DB_HOST, port=DB_PORT, user=DB_USER,
                                   password=DB_PASSWORD, db=DB_NAME, charset='utf8')
            cursor = conn.cursor()
            cursor.execute("""
                UPDATE `Антенна`
                SET `Тип антенны` = %s,
                    `Роль антенны` = %s,
                    `Рабочая частота` = %s,
                    `Усиление` = %s,
                    `Входное сопротивление` = %s,
                    `Диаграмма направленности` = %s
                WHERE `ID антенны` = %s;
            """, (self.type_input.text(), self.role_input.text(), float(self.freq_input.text()),
                  float(self.gain_input.text()), float(self.inp_res_input.text()), self.pattern_input.text(),
                  self.ant_id))
            conn.commit()
            conn.close()
```

```
            QtWidgets.QMessageBox.information(self, "Успех", "Антенна обновлена.")
            self.close()
        except pymysql.Error as e:
            QtWidgets.QMessageBox.critical(self, "Ошибка", f"Ошибка базы данных: {e}")

    def confirm_close(self):
        if not self.is_modified:
            self.close()
            return
        msg = QtWidgets.QMessageBox(self)
        msg.setWindowTitle("Подтверждение")
        msg.setText("Изменения не будут сохранены. Закрыть окно?")
        yes = msg.addButton("Да",  QtWidgets.QMessageBox.YesRole)
        no  = msg.addButton("Нет", QtWidgets.QMessageBox.NoRole)
        msg.exec_()
        if msg.clickedButton() == yes:
            self.close()


class AddAircraftDialog(QtWidgets.QDialog):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setWindowTitle("Добавление БПЛА")
        self.setFixedSize(300, 400)
        self.is_modified = False

        layout = QtWidgets.QVBoxLayout(self)
        self.model_input       = QtWidgets.QLineEdit()
        self.length_input      = QtWidgets.QLineEdit()
        self.radius_input      = QtWidgets.QLineEdit()
        self.material_input    = QtWidgets.QLineEdit()
        self.conductivity_input= QtWidgets.QLineEdit()
        self.permittivity_input= QtWidgets.QLineEdit()

        for w in (self.model_input, self.length_input, self.radius_input,
                  self.material_input, self.conductivity_input, self.permittivity_input):
            w.textChanged.connect(self.mark_modified)

        layout.addWidget(QtWidgets.QLabel("Модель ЛА:"))
        layout.addWidget(self.model_input)
        layout.addWidget(QtWidgets.QLabel("Длина фюзеляжа:"))
        layout.addWidget(self.length_input)
        layout.addWidget(QtWidgets.QLabel("Радиус фюзеляжа:"))
        layout.addWidget(self.radius_input)
        layout.addWidget(QtWidgets.QLabel("Материал фюзеляжа:"))
        layout.addWidget(self.material_input)
        layout.addWidget(QtWidgets.QLabel("Электропроводность материала:"))
        layout.addWidget(self.conductivity_input)
        layout.addWidget(QtWidgets.QLabel("Диэлектрическая проницаемость:"))
        layout.addWidget(self.permittivity_input)

        btn_layout = QtWidgets.QHBoxLayout()
        self.add_button = QtWidgets.QPushButton("Добавить")
        self.add_button.clicked.connect(self.add_aircraft_to_db)
        self.cancel_button = QtWidgets.QPushButton("Закрыть")
        self.cancel_button.clicked.connect(self.confirm_close)
        btn_layout.addWidget(self.add_button)
        btn_layout.addWidget(self.cancel_button)
        layout.addLayout(btn_layout)

    def mark_modified(self, _):
        self.is_modified = True
```

```
    def add_aircraft_to_db(self):
        model = self.model_input.text()
        length = self.length_input.text()
        radius = self.radius_input.text()
        material = self.material_input.text()
        conductivity = self.conductivity_input.text()
        permittivity = self.permittivity_input.text()

        if not all([model, length, radius, material, conductivity, permittivity]):
            QtWidgets.QMessageBox.warning(self, "Ошибка", "Все поля должны быть заполнены.")
            return

        try:
            conn = pymysql.connect(host=DB_HOST, port=DB_PORT, user=DB_USER,
                                   password=DB_PASSWORD, db=DB_NAME, charset='utf8')
            cursor = conn.cursor()
            cursor.execute("""
                INSERT INTO `Летательный аппарат` (`Модель ЛА`, `Длина фюзеляжа`, `Радиус фюзеляжа`,
                                                   `Материал фюзеляжа`, `Электропроводность материала`, `Диэлектрическая
проницаемость`)
                VALUES (%s, %s, %s, %s, %s, %s);
            """, (model, float(length), float(radius), material, float(conductivity), float(permittivity)))
            conn.commit()
            conn.close()
            QtWidgets.QMessageBox.information(self, "Успех", "БПЛА добавлен.")
            self.close()
        except pymysql.Error as e:
            QtWidgets.QMessageBox.critical(self, "Ошибка", f"Ошибка базы данных: {e}")

    def confirm_close(self):
        if not self.is_modified:
            self.close()
            return
        msg = QtWidgets.QMessageBox(self)
        msg.setWindowTitle("Подтверждение")
        msg.setText("Изменения не будут сохранены. Закрыть окно?")
        yes = msg.addButton("Да",  QtWidgets.QMessageBox.YesRole)
        no  = msg.addButton("Нет", QtWidgets.QMessageBox.NoRole)
        msg.exec_()
        if msg.clickedButton() == yes:
            self.close()


class SelectAircraftDialog(QtWidgets.QDialog):
    def __init__(self, parent=None, mode="edit"):
        super().__init__(parent)
        self.setWindowTitle("Выбор БПЛА")
        self.setFixedSize(850, 600)
        self.mode = mode
        layout = QtWidgets.QVBoxLayout(self)

        self.aircraft_list = QtWidgets.QListWidget()
        self.aircraft_list.setSelectionMode(QtWidgets.QAbstractItemView.SingleSelection)
        layout.addWidget(self.aircraft_list)

        aircrafts = get_aircrafts()
        for la_id, la_model, length, radius, material, conductivity, permittivity in aircrafts:
            text = (f"{la_id}: {la_model} | "
                    f"Длина = {length:.2f} м | Радиус = {radius:.2f} м | "
                    f"Материал = {material} | Проводимость = {conductivity:.2f} С/м | "
                    f"Диэл. проницаемость = {permittivity:.2f}")
```

```
            item = QtWidgets.QListWidgetItem(text)
            item.setData(QtCore.Qt.UserRole, (la_id, la_model, length, radius, material, conductivity, permittivity))
            self.aircraft_list.addItem(item)

        btn_layout = QtWidgets.QHBoxLayout()
        if self.mode == "edit":
            self.action_button = QtWidgets.QPushButton("Редактировать")
            self.action_button.clicked.connect(self.open_edit_dialog)
        elif self.mode == "delete":
            self.action_button = QtWidgets.QPushButton("Удалить")
            self.action_button.clicked.connect(self.delete_aircraft)
        self.close_button = QtWidgets.QPushButton("Закрыть")
        self.close_button.clicked.connect(self.close)
        btn_layout.addWidget(self.action_button)
        btn_layout.addWidget(self.close_button)
        layout.addLayout(btn_layout)

        self.aircraft_list.itemSelectionChanged.connect(self.toggle_action_button)
        self.action_button.setEnabled(False)

    def toggle_action_button(self):
        self.action_button.setEnabled(len(self.aircraft_list.selectedItems()) == 1)

    def open_edit_dialog(self):
        selected = self.aircraft_list.selectedItems()[0]
        aircraft_data = selected.data(QtCore.Qt.UserRole)
        edit_dialog = EditAircraftDialog(aircraft_data, self)
        edit_dialog.exec_()

    def delete_aircraft(self):
        selected = self.aircraft_list.selectedItems()[0]
        la_id = selected.data(QtCore.Qt.UserRole)[0]

        msg = QtWidgets.QMessageBox(self)
        msg.setWindowTitle("Подтверждение")
        msg.setText(f"Вы уверены, что хотите удалить БПЛА {la_id}?")
        yes = msg.addButton("Да", QtWidgets.QMessageBox.YesRole)
        no = msg.addButton("Нет", QtWidgets.QMessageBox.NoRole)
        msg.exec_()

        if msg.clickedButton() == yes:
            try:
                conn = pymysql.connect(host=DB_HOST, port=DB_PORT, user=DB_USER,
                                       password=DB_PASSWORD, db=DB_NAME, charset='utf8')
                cursor = conn.cursor()
                cursor.execute("DELETE FROM `Летательный аппарат` WHERE `ID ЛА` = %s", (la_id,))
                conn.commit()
                conn.close()
                QtWidgets.QMessageBox.information(self, "Успех", "БПЛА удален.")
                self.aircraft_list.takeItem(self.aircraft_list.row(selected))
            except pymysql.Error as e:
                QtWidgets.QMessageBox.critical(self, "Ошибка", f"Ошибка базы данных: {e}")


class EditAircraftDialog(QtWidgets.QDialog):
    def __init__(self, aircraft_data, parent=None):
        super().__init__(parent)
        self.setWindowTitle("Редактирование БПЛА")
        self.setFixedSize(300, 400)
        self.la_id         = aircraft_data[0]
        self.original_data = aircraft_data
        self.is_modified   = False
```

```
        layout = QtWidgets.QVBoxLayout(self)
        self.model_input       = QtWidgets.QLineEdit(aircraft_data[1])
        self.length_input      = QtWidgets.QLineEdit(str(aircraft_data[2]))
        self.radius_input      = QtWidgets.QLineEdit(str(aircraft_data[3]))
        self.material_input    = QtWidgets.QLineEdit(aircraft_data[4])
        self.cond_input        = QtWidgets.QLineEdit(str(aircraft_data[5]))
        self.permittivity_input = QtWidgets.QLineEdit(str(aircraft_data[6]))

        for w in (self.model_input, self.length_input, self.radius_input,
                  self.material_input, self.cond_input, self.permittivity_input):
            w.textChanged.connect(self.mark_modified)
            w.textChanged.connect(self.check_changes)

        for label, widget in [
            ("Модель ЛА:",       self.model_input),
            ("Длина фюзеляжа:",  self.length_input),
            ("Радиус фюзеляжа:", self.radius_input),
            ("Материал фюзеляжа:", self.material_input),
            ("Электропроводность:", self.cond_input),
            ("Диэлектрическая проницаемость:", self.permittivity_input),
        ]:
            layout.addWidget(QtWidgets.QLabel(label))
            layout.addWidget(widget)

        btn_layout = QtWidgets.QHBoxLayout()
        self.update_button = QtWidgets.QPushButton("Изменить")
        self.update_button.clicked.connect(self.update_aircraft_in_db)
        self.cancel_button = QtWidgets.QPushButton("Закрыть")
        self.cancel_button.clicked.connect(self.confirm_close)
        btn_layout.addWidget(self.update_button)
        btn_layout.addWidget(self.cancel_button)
        layout.addLayout(btn_layout)

        self.update_button.setEnabled(False)

        for w in [self.model_input, self.length_input, self.radius_input,
                  self.material_input, self.cond_input, self.permittivity_input]:
            w.textChanged.connect(self.check_changes)
        self.update_button.setEnabled(False)

    def mark_modified(self, _):
        self.is_modified = True

    def check_changes(self):
        od = self.original_data
        has = (
            self.model_input.text()       != str(od[1]) or
            self.length_input.text()      != str(od[2]) or
            self.radius_input.text()      != str(od[3]) or
            self.material_input.text()    != str(od[4]) or
            self.cond_input.text()        != str(od[5]) or
            self.permittivity_input.text() != str(od[6])
        )
        self.update_button.setEnabled(has)

    def update_aircraft_in_db(self):
        if not all([self.model_input.text(), self.length_input.text(),
                    self.radius_input.text(), self.material_input.text(),
                    self.cond_input.text(), self.permittivity_input.text()]):
            QtWidgets.QMessageBox.warning(self, "Ошибка", "Все поля должны быть заполнены.")
            return
```

```
        try:
            conn = pymysql.connect(host=DB_HOST, port=DB_PORT, user=DB_USER,
                                   password=DB_PASSWORD, db=DB_NAME, charset='utf8')
            cursor = conn.cursor()
            cursor.execute("""
                UPDATE `Летательный аппарат`
                SET `Модель ЛА` = %s,
                    `Длина фюзеляжа` = %s,
                    `Радиус фюзеляжа` = %s,
                    `Материал фюзеляжа` = %s,
                    `Электропроводность материала` = %s,
                    `Диэлектрическая проницаемость` = %s
                WHERE `ID ЛА` = %s;
            """, (
                self.model_input.text(),
                float(self.length_input.text()),
                float(self.radius_input.text()),
                self.material_input.text(),
                float(self.cond_input.text()),
                float(self.permittivity_input.text()),
                self.la_id
            ))
            conn.commit()
            conn.close()
            QtWidgets.QMessageBox.information(self, "Успех", "БПЛА обновлён.")
            self.close()
        except pymysql.Error as e:
            QtWidgets.QMessageBox.critical(self, "Ошибка БД", str(e))

    def confirm_close(self):
        if not self.is_modified:
            self.close()
            return
        msg = QtWidgets.QMessageBox(self)
        msg.setWindowTitle("Подтверждение")
        msg.setText("Изменения не будут сохранены. Закрыть окно?")
        yes = msg.addButton("Да",  QtWidgets.QMessageBox.YesRole)
        no  = msg.addButton("Нет", QtWidgets.QMessageBox.NoRole)
        msg.exec_()
        if msg.clickedButton() == yes:
            self.close()

class DataLoaderDialog(QtWidgets.QDialog):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setWindowTitle("Загрузка данных")
        self.resize(800, 800)

        main_layout = QtWidgets.QVBoxLayout(self)

        self.la_list = QtWidgets.QListWidget()
        self.la_list.setSelectionMode(QtWidgets.QAbstractItemView.SingleSelection)
        self.la_list.setSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

        for la_id, la_model, length, radius, material, conductivity, permittivity in get_aircrafts():
            item = QtWidgets.QListWidgetItem(
                f"{la_id}: {la_model} (Длина = {length:.2f}, Радиус = {radius:.2f}, Материал = {material})"
            )
            item.setData(QtCore.Qt.UserRole,
                         (la_id, length, radius, material, conductivity, permittivity))
            self.la_list.addItem(item)
```

```
        self.ant_list = QtWidgets.QListWidget()
        self.ant_list.setSelectionMode(QtWidgets.QAbstractItemView.MultiSelection)
        self.ant_list.setSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)

        for ant_id, ant_type, ant_role, freq, gain, inp_res, pattern in get_antennas_full():
            text = (f"{ant_id}: {ant_type} ({ant_role}) | Частота = {freq:.2f} МГц | "
                    f"Усиление = {gain:.2f} дБ | Сопротивление = {inp_res:.2f} Ω")
            item = QtWidgets.QListWidgetItem(text)
            item.setData(QtCore.Qt.UserRole,
                         (ant_id, ant_type, ant_role, freq, gain, inp_res, pattern))
            self.ant_list.addItem(item)

        la_label = QtWidgets.QLabel("Выберите одну модель ЛА:")
        ant_label = QtWidgets.QLabel("Выберите одну или более пару антенн передатчик/приемник:")

        self.load_button = QtWidgets.QPushButton("Загрузить")
        self.load_button.setEnabled(False)
        self.load_button.clicked.connect(self.accept)

        self.la_list.itemSelectionChanged.connect(self.check_selection)
        self.ant_list.itemSelectionChanged.connect(self.check_selection)

        main_layout.addWidget(la_label)
        main_layout.addWidget(self.la_list, stretch=1)
        main_layout.addWidget(ant_label)
        main_layout.addWidget(self.ant_list, stretch=2)

        main_layout.addWidget(self.load_button)

        qr = self.frameGeometry()
        cp = QtWidgets.QDesktopWidget().availableGeometry().center()
        qr.moveCenter(cp)
        self.move(qr.topLeft())

    def check_selection(self):
        la_selected = (len(self.la_list.selectedItems()) == 1)
        ant_selected = (len(self.ant_list.selectedItems()) >= 1)
        self.load_button.setEnabled(la_selected and ant_selected)

    def get_selection(self):
        la_item = self.la_list.selectedItems()[0]
        la_id, length, radius, material, conductivity, permittivity = la_item.data(QtCore.Qt.UserRole)
        ant_data = [item.data(QtCore.Qt.UserRole) for item in self.ant_list.selectedItems()]
        return la_id, length, radius, material, conductivity, permittivity, ant_data

class Model3DWidget(QtWidgets.QWidget):

    position_clicked = QtCore.pyqtSignal(int)

    def __init__(self, length, radius, material, conductivity, permittivity, num_positions, parent=None):
        super().__init__(parent)
        self.setMinimumSize(500, 500)

        self.length = float(length)
        self.radius = float(radius)
        self.material = material
        self.conductivity = conductivity
        self.permittivity = permittivity
        self.num_positions = int(num_positions)

        self.positions = []
```

```
        self.position_spheres = []
        self.used_positions = set()

        self.init_ui()

    def add_background_grid(self):
        size = 50
        step = 0.5
        y_level = -self.radius * 18
        lines = []
        z_range = np.arange(-size, size + step, step)
        for z in z_range:
            lines.append([[-size, y_level, z], [size, y_level, z]])

        x_range = np.arange(-size, size + step, step)
        for x in x_range:
            lines.append([[x, y_level, -size], [x, y_level, size]])

        pos = np.array(lines).reshape(-1, 3)

        color = np.tile([0.5, 0.5, 0.5, 1.0], (len(pos), 1))

        grid = gl.GLLinePlotItem(pos=pos, color=color, width=1.0, antialias=True, mode='lines')
        grid.setGLOptions('opaque')
        self.view.addItem(grid)

    def init_ui(self):
        layout = QtWidgets.QVBoxLayout(self)

        self.view = gl.GLViewWidget()
        self.view.opts['distance'] = max(self.length, self.radius) * 1.5
        self.view.setBackgroundColor('w')
        self.view.setCameraPosition(pos=QVector3D(self.length * 0.1, 0, 0))
        layout.addWidget(self.view)
        self.add_background_grid()
        self._add_fuselage_with_components()
        self._generate_install_positions_spaced()
        self.view.mousePressEvent = self._mouse_press_event

    def _add_fuselage_with_components(self):
        material_colors = {
            'Углепластик': (0.4, 0.4, 0.4, 1),
            'Алюминий':    (0.95, 0.95, 0.95, 1),
            'Титан':       (0.8, 0.8, 0.85, 1),
            'Сталь':       (0.85, 0.85, 0.85, 1),
            'Композит':    (0.5, 0.8, 0.6, 1),
        }
        color = material_colors.get(self.material, (0.7, 0.7, 0.7, 1))
        fuselage = self.create_cylinder_mesh(self.length, self.radius, color=color)
        fuselage.rotate(90, 0, 1, 0)
        fuselage.setGLOptions('opaque')
        self.view.addItem(fuselage)

        nose_length = self.radius * 1.2
        nose = self.create_cone_mesh(self.radius, nose_length, color=color)
        nose.translate(self.length / 2 + nose_length / 2, 0, 0)
        nose.setGLOptions('opaque')
        self.view.addItem(nose)

        tail_disk = self.create_disk_mesh(self.radius, color=color)
        tail_disk.translate(-self.length / 2, 0, 0)
        tail_disk.setGLOptions('opaque')
```

```
        self.view.addItem(tail_disk)

        wing_span  = self.length * 0.7
        wing_depth = self.radius * 0.05
        wing_width = self.length * 0.2

        left_wing = self.create_box_mesh(wing_width, wing_span, wing_depth, color=color)
        left_wing.translate(0, -wing_span / 2, -wing_depth / 2)
        left_wing.setGLOptions('opaque')
        self.view.addItem(left_wing)

        right_wing = self.create_box_mesh(wing_width, wing_span, wing_depth, color=color)
        right_wing.translate(0, wing_span / 2, -wing_depth / 2)
        right_wing.setGLOptions('opaque')
        self.view.addItem(right_wing)

    def create_cylinder_mesh(self, length, radius, color=(0.5, 0.5, 0.8, 1), slices=32):
        length = float(length)
        radius = float(radius)
        verts = []
        faces = []

        for i in range(slices + 1):
            theta = 2 * np.pi * i / slices
            x = radius * np.cos(theta)
            y = radius * np.sin(theta)
            verts.append([x, y, -length / 2])
            verts.append([x, y,  length / 2])

        for i in range(slices):
            idx = 2 * i
            faces.append([idx,   idx + 2, idx + 1])
            faces.append([idx + 1, idx + 2, idx + 3])

        mesh = gl.MeshData(vertexes=np.array(verts), faces=np.array(faces))
        md = gl.GLMeshItem(meshdata=mesh, smooth=True, color=color, shader='shaded')
        return md

    def create_cone_mesh(self, radius, length, color=(0.5, 0.5, 0.8, 1), slices=32):
        verts = []
        faces = []

        verts.append([length / 2, 0, 0])

        for i in range(slices):
            theta = 2 * np.pi * i / slices
            y = radius * np.cos(theta)
            z = radius * np.sin(theta)
            verts.append([-length / 2, y, z])

        for i in range(1, slices):
            faces.append([0, i, i + 1])
        faces.append([0, slices, 1])

        base_center_idx = len(verts)
        verts.append([-length / 2, 0, 0])
        for i in range(1, slices):
            faces.append([base_center_idx, i + 1, i])
        faces.append([base_center_idx, 1, slices])

        mesh = gl.MeshData(vertexes=np.array(verts), faces=np.array(faces))
        md = gl.GLMeshItem(meshdata=mesh, smooth=True, color=color, shader='shaded')
```

```
    return md

def create_disk_mesh(self, radius, color=(0.5, 0.5, 0.8, 1), slices=32):
    verts = []
    faces = []

    verts.append([0, 0, 0])
    for i in range(slices):
        theta = 2 * np.pi * i / slices
        y = radius * np.cos(theta)
        z = radius * np.sin(theta)
        verts.append([0, y, z])

    for i in range(1, slices):
        faces.append([0, i, i + 1])
    faces.append([0, slices, 1])

    mesh = gl.MeshData(vertexes=np.array(verts), faces=np.array(faces))
    md = gl.GLMeshItem(meshdata=mesh, smooth=True, color=color, shader='shaded')
    return md

def create_box_mesh(self, dx, dy, dz, color=(0.5, 0.5, 0.8, 1)):
    verts = np.array([
        [-dx / 2, -dy / 2, -dz / 2],
        [ dx / 2, -dy / 2, -dz / 2],
        [ dx / 2,  dy / 2, -dz / 2],
        [-dx / 2,  dy / 2, -dz / 2],
        [-dx / 2, -dy / 2,  dz / 2],
        [ dx / 2, -dy / 2,  dz / 2],
        [ dx / 2,  dy / 2,  dz / 2],
        [-dx / 2,  dy / 2,  dz / 2],
    ], dtype=np.float32)

    faces = np.array([
        [0, 1, 2], [0, 2, 3],
        [4, 5, 6], [4, 6, 7],
        [0, 1, 5], [0, 5, 4],
        [2, 3, 7], [2, 7, 6],
        [1, 2, 6], [1, 6, 5],
        [0, 3, 7], [0, 7, 4],
    ], dtype=np.int32)

    mesh = gl.MeshData(vertexes=verts, faces=faces)
    md = gl.GLMeshItem(meshdata=mesh, smooth=True, color=color, shader='shaded')
    return md

def _generate_install_positions_spaced(self):
    self.positions = []
    self.position_spheres = []
    self.used_positions = set()

    sphere_radius = self.radius * 0.08
    min_dist = 2 * sphere_radius

    attempts_limit = 1000
    attempts = 0

    while len(self.positions) < self.num_positions and attempts < attempts_limit:
        attempts += 1

        theta = random.uniform(0, 2 * np.pi)
        x = self.length * random.uniform(-0.4, 0.4)
```

```
            y = self.radius * np.cos(theta)
            z = self.radius * np.sin(theta)
            candidate = np.array([x, y, z])

            ok = True
            for existing in self.positions:
                if np.linalg.norm(candidate - np.array(existing)) < min_dist:
                    ok = False
                    break

            if not ok:
                continue

            self.positions.append(tuple(candidate))

            md = gl.MeshData.sphere(rows=10, cols=10, radius=sphere_radius)
            sphere = gl.GLMeshItem(meshdata=md, smooth=True, color=(1, 0, 0, 0.6))
            sphere.translate(candidate[0], candidate[1], candidate[2])
            sphere.setVisible(False)
            self.view.addItem(sphere)
            self.position_spheres.append(sphere)

        while len(self.positions) < self.num_positions:
            theta = random.uniform(0, 2 * np.pi)
            x = self.length * random.uniform(-0.4, 0.4)
            y = self.radius * np.cos(theta)
            z = self.radius * np.sin(theta)
            candidate = np.array([x, y, z])

            self.positions.append(tuple(candidate))

            md = gl.MeshData.sphere(rows=10, cols=10, radius=sphere_radius)
            sphere = gl.GLMeshItem(meshdata=md, smooth=True, color=(1, 0, 0, 0.6))
            sphere.translate(candidate[0], candidate[1], candidate[2])
            sphere.setVisible(False)
            self.view.addItem(sphere)
            self.position_spheres.append(sphere)

    def highlight_positions(self):
        for idx, sph in enumerate(self.position_spheres):
            sph.setVisible(idx not in self.used_positions)

    def hide_positions(self):
        for sph in self.position_spheres:
            sph.setVisible(False)

    def _mouse_press_event(self, ev):
        if ev.button() == QtCore.Qt.RightButton:
            pos2d = ev.pos()
            w, h = self.view.width(), self.view.height()

            for idx, sph in enumerate(self.position_spheres):
                if idx in self.used_positions:
                    continue

                center3d = np.array(self.positions[idx] + (1.0,))
                try:
                    proj_mat = self.view.projectionMatrix()
                    view_mat = self.view.viewMatrix()
                except Exception:
                    proj_mat = None
                    view_mat = None
```

```
            if proj_mat is not None and view_mat is not None:
                mvp = proj_mat * view_mat
                v = QtGui.QVector4D(center3d[0], center3d[1], center3d[2], 1.0)
                tp = mvp.map(v)
                if abs(tp.w()) < 1e-6:
                    continue
                ndc_x = tp.x() / tp.w()
                ndc_y = tp.y() / tp.w()
                screen_x = (ndc_x * 0.5 + 0.5) * w
                screen_y = (1.0 - (ndc_y * 0.5 + 0.5)) * h

                offset3d = np.array([center3d[0], center3d[1] + self.radius * 0.08, center3d[2], 1.0])
                v_off = QtGui.QVector4D(offset3d[0], offset3d[1], offset3d[2], 1.0)
                tp_off = mvp.map(v_off)
                if abs(tp_off.w()) < 1e-6:
                    screen_radius = 20
                else:
                    ndc_off_x = tp_off.x() / tp_off.w()
                    ndc_off_y = tp_off.y() / tp_off.w()
                    screen_x_off = (ndc_off_x * 0.5 + 0.5) * w
                    screen_y_off = (1.0 - (ndc_off_y * 0.5 + 0.5)) * h
                    screen_radius = np.hypot(screen_x_off - screen_x, screen_y_off - screen_y)

                dist = np.hypot(pos2d.x() - screen_x, pos2d.y() - screen_y)
                if dist <= screen_radius:
                    sph.setVisible(False)
                    self.used_positions.add(idx)
                    self.position_clicked.emit(idx)
                    return
            else:
                cx, cy = w / 2, h / 2
                dx = abs(pos2d.x() - cx)
                dy = abs(pos2d.y() - cy)
                if dx < 40 and dy < 40:
                    sph.setVisible(False)
                    self.used_positions.add(idx)
                    self.position_clicked.emit(idx)
                    return

        return

    gl.GLViewWidget.mousePressEvent(self.view, ev)

class WorkTab(QtWidgets.QWidget):
    def __init__(self, parent=None, la_id=None, length=1.0, radius=1.0,
                 material='', conductivity=0.0, permittivity=0.0, ant_data=None):
        super().__init__(parent)
        self.la_id = la_id
        self.length = float(length)
        self.radius = float(radius)
        self.material = material
        self.conductivity = conductivity
        self.permittivity = permittivity
        self.ant_data = ant_data or []
        self.placed_antennas = []
        self.dirty = False
        self.cached_ks = None
        self.init_ui()

    def init_ui(self):
        main_layout = QtWidgets.QVBoxLayout(self)
```

```
top_layout = QtWidgets.QHBoxLayout()

num_positions = len(self.ant_data)
self.model_view = Model3DWidget(self.length, self.radius,
                          self.material, self.conductivity,
                          self.permittivity, num_positions, self)
self.model_view.position_clicked.connect(self.on_position_clicked)

right_panel = QtWidgets.QVBoxLayout()

self.ant_list_widget = QtWidgets.QListWidget()
self.ant_list_widget.setFixedWidth(650)
self.ant_list_widget.setSelectionMode(QtWidgets.QAbstractItemView.SingleSelection)
for (ant_id, ant_type, ant_role, freq, gain, inp_res, pattern) in self.ant_data:
    text = (f"{ant_id}: {ant_type} ({ant_role}) | Частота = {freq} МГц | "
            f"Усиление = {gain} дБ | Сопротивление = {inp_res} Ω")
    item = QtWidgets.QListWidgetItem(text)
    item.setData(QtCore.Qt.UserRole, (ant_id, ant_type, ant_role, freq, gain, inp_res, pattern))
    self.ant_list_widget.addItem(item)

self.ant_list_widget.itemSelectionChanged.connect(self.on_ant_selected)

self.pattern_plot = pg.PlotWidget(title="Диаграмма направленности")
self.pattern_plot.setFixedWidth(650)
self.pattern_plot.setBackground('w')
self.pattern_plot.hideAxis('bottom')
self.pattern_plot.hideAxis('left')

right_panel.addWidget(QtWidgets.QLabel("Доступные антенны:"))
right_panel.addWidget(self.ant_list_widget, stretch=1)
right_panel.addWidget(self.pattern_plot, stretch=1)
right_panel.addStretch()

self.place_button = QtWidgets.QPushButton("Начать размещение")
self.place_button.setEnabled(False)
self.place_button.clicked.connect(self.activate_position_selection)

self.finish_button = QtWidgets.QPushButton("Завершить размещение")
self.finish_button.setEnabled(False)
self.finish_button.clicked.connect(self.finish_placement)

btn_layout = QtWidgets.QHBoxLayout()
btn_layout.addWidget(self.place_button)
btn_layout.addWidget(self.finish_button)
btn_layout.addStretch()
right_panel.addLayout(btn_layout)

top_layout.addWidget(self.model_view, stretch=1)
top_layout.addLayout(right_panel, stretch=0)

bottom_layout = QtWidgets.QHBoxLayout()
self.predict_button = QtWidgets.QPushButton("Прогноз коэффициента связи")
self.predict_button.setEnabled(False)
self.predict_button.clicked.connect(self.show_prediction)

self.remove_one_button = QtWidgets.QPushButton("Убрать антенну")
self.remove_one_button.setEnabled(False)
self.remove_one_button.clicked.connect(self.remove_one)

self.remove_all_button = QtWidgets.QPushButton("Убрать все антенны")
self.remove_all_button.setEnabled(False)
```

```
        self.remove_all_button.clicked.connect(self.remove_all)

        bottom_layout.addWidget(self.predict_button)
        bottom_layout.addWidget(self.remove_one_button)
        bottom_layout.addWidget(self.remove_all_button)
        bottom_layout.addStretch()

        main_layout.addLayout(top_layout)
        main_layout.addLayout(bottom_layout)

    def on_ant_selected(self):
        items = self.ant_list_widget.selectedItems()
        self.place_button.setEnabled(len(items) == 1)
        if items:
            self.load_pattern_from_db(items[0])

    def load_pattern_from_db(self, item):
        ant_id, ant_type, ant_role, freq, gain, inp_res, pattern_json = item.data(QtCore.Qt.UserRole)
        self.pattern_plot.clear()

        angles = np.linspace(0, 360, 361)
        theta = np.deg2rad(angles)

        ant_type_lower = ant_type.lower()
        freq_val = float(freq)
        sigma = 15 * (1000.0 / freq_val)

        if "диполь" in ant_type_lower:
            degree = 1.0 + (freq_val / 1000.0)
            values = gain * (np.abs(np.cos(theta)) ** degree)

        elif "штыревая" in ant_type_lower:
            values = gain + 1.0 * np.cos(2 * theta)

        elif "щелевая" in ant_type_lower:
            values = gain * np.exp(-((angles - 0) ** 2) / (2 * (sigma ** 2)))

        elif "вибратор" in ant_type_lower:
            expo = 5 * (1000.0 / freq_val)
            values = gain * (np.abs(np.cos(theta)) ** expo)

        elif "зеркальная" in ant_type_lower:
            expo = 5 * (1000.0 / freq_val)
            values = gain * (np.abs(np.sin(theta)) ** expo)

        elif "рупорная" in ant_type_lower:
            sigma_horn = 5 * (1000.0 / freq_val)
            values = gain * np.exp(-((angles - 0) ** 2) / (2 * (sigma_horn ** 2)))

        elif "кольцевая" in ant_type_lower:
            expo = 2 * (freq_val / 1000.0)
            values = gain * (np.abs(np.sin(2 * theta)) ** expo)

        elif "петлевая" in ant_type_lower:
            expo = 2 * (freq_val / 1000.0)
            values = gain * (np.abs(np.sin(4 * theta)) ** expo)

        elif "спиральная" in ant_type_lower:
            values = gain * (0.5 + 0.5 * np.cos(3 * theta * (freq_val / 1000.0)))

        elif "панельная" in ant_type_lower:
            expo = 1.5 * (freq_val / 1000.0)
```

```
            values = gain * (np.abs(np.cos(theta)) ** expo)

        elif "микрополосковая" in ant_type_lower:
            sigma_micro = 20 * (1000.0 / freq_val)
            values = gain * np.exp(-((angles - 0) ** 2) / (2 * (sigma_micro ** 2)))

        elif "логопериодическ" in ant_type_lower:
            sigma_log = 15 * (1000.0 / freq_val)
            values = gain * np.exp(-((angles - 0) ** 2) / (2 * (sigma_log ** 2)))

        elif "конусная" in ant_type_lower:
            values = gain * (0.3 + 0.7 * np.abs(np.sin(3 * theta * (freq_val / 1000.0))))

        elif "проволочная" in ant_type_lower:
            sigma_wire = 8 * (1000.0 / freq_val)
            values = gain * np.exp(-((angles - 0) ** 2) / (2 * (sigma_wire ** 2)))

        elif "фазированная" in ant_type_lower:
            expo = 40 * (freq_val / 1000.0)
            values = gain * (np.abs(np.cos(theta)) ** expo)

        else:
            values = gain * np.ones_like(angles)

        pen = pg.mkPen('b', width=2)
        self.pattern_plot.plot(angles, values, pen=pen)
        self.pattern_plot.setLabel('left', "Усиление, dB")
        self.pattern_plot.setLabel('bottom', "Угол, °")

    def activate_position_selection(self):
        self.model_view.highlight_positions()
        self.finish_button.setEnabled(True)
        QtWidgets.QMessageBox.information(
            self,
            "Инструкция",
            "Выберите одну антенну из списка, а затем кликните правой кнопкой мыши по одной из подсвеченных сфер на
3D–модели, чтобы установить антенну.\n"
            "Когда закончите размещение – нажмите «Завершить размещение»."
        )

    def finish_placement(self):
        self.model_view.hide_positions()
        self.finish_button.setEnabled(False)

    def on_position_clicked(self, pos_idx):
        self.place_antenna(pos_idx)

    def place_antenna(self, pos_idx):
        ant_item = self.ant_list_widget.selectedItems()[0]
        ant_id, ant_type, ant_role, freq, gain, inp_res, pattern_json = ant_item.data(QtCore.Qt.UserRole)

        occupied_indices = [pos for (_, _, pos, _, _, _) in self.placed_antennas]
        if pos_idx in occupied_indices:
            QtWidgets.QMessageBox.information(self, "Ошибка", "Эта позиция уже занята!")
            return

        coordinates = self.model_view.positions[pos_idx]
        x, y, z = coordinates

        self.placed_antennas.append((ant_id, ant_role, pos_idx, x, y, z))
        self.dirty = True
        self.cached_ks = None
```

```
        self.model_view.position_spheres[pos_idx].setVisible(False)

        self.predict_button.setEnabled(True)
        self.remove_one_button.setEnabled(True)
        self.remove_all_button.setEnabled(True)

        QtWidgets.QMessageBox.information(
            self,
            "Антенна размещена",
            f"Антенна {ant_id} ({ant_type}, {ant_role}) размещена в позиции {pos_idx + 1}."
        )

    def remove_one(self):
        if not self.placed_antennas:
            return

        items = [f"Антенна {ant_id} ({ant_role}) на позиции {pos_idx + 1}"
                 for ant_id, ant_role, pos_idx, _, _, _ in self.placed_antennas]
        dlg = QtWidgets.QInputDialog(self)
        dlg.setWindowTitle("Удаление размещенной антенны с 3D-модели")
        dlg.setLabelText("Выберите антенну, которую хотите убрать:")
        dlg.setComboBoxItems(items)
        dlg.setComboBoxEditable(False)

        dlg.setFixedSize(400, 200)

        dlg.setOkButtonText("OK")
        dlg.setCancelButtonText("Отмена")

        if dlg.exec_() == QtWidgets.QDialog.Accepted:
            choice = dlg.textValue()
            idx = items.index(choice)
            removed = self.placed_antennas.pop(idx)
            self.dirty = True
            self.cached_ks = None

            pos_idx = removed[2]
            if pos_idx in self.model_view.used_positions:
                self.model_view.used_positions.remove(pos_idx)
            self.model_view.hide_positions()
            self.finish_button.setEnabled(False)

            QtWidgets.QMessageBox.information(
                self,
                "Антенна удалена с 3D-модели",
                f"Антенна {removed[0]} удалена из позиции {pos_idx + 1}."
            )

        if not self.placed_antennas:
            self.predict_button.setEnabled(False)
            self.remove_one_button.setEnabled(False)
            self.remove_all_button.setEnabled(False)

    def remove_all(self):
        if not self.placed_antennas:
            return

        self.placed_antennas.clear()
        self.dirty = True
        self.cached_ks = None

        self.model_view.used_positions.clear()
```

```
        self.model_view.hide_positions()

        self.predict_button.setEnabled(False)
        self.remove_one_button.setEnabled(False)
        self.remove_all_button.setEnabled(False)
        self.finish_button.setEnabled(False)

        QtWidgets.QMessageBox.information(self, "Удаление всех размещенных антенн", "Все антенны были убраны")

    def show_prediction(self):
        param_map = {aid: freq for aid, _, _, freq, _, _, _ in self.ant_data}
        rx_list, tx_list = [], []
        for ant_id, ant_role, pos_idx, _, _, _ in self.placed_antennas:
            role = ant_role.lower()
            freq = param_map.get(ant_id)
            if freq is None:
                continue
            if "приемник" in role:
                rx_list.append((ant_id, pos_idx, freq))
            elif "передатчик" in role:
                tx_list.append((ant_id, pos_idx, freq))

        if not rx_list or not tx_list:
            QtWidgets.QMessageBox.information(
                self, "Невозможно рассчитать",
                "Для прогноза коэффициента связи нужно как минимум\n"
                "одна антенна-«приемник» и одна антенна-«передатчик»."
            )
            return

        if self.cached_ks is not None:
            coeff = self.cached_ks
        else:
            n_rx, n_tx = len(rx_list), len(tx_list)
            pos3d = self.model_view.positions
            max_d = 2 * self.length
            coeff = [[None] * n_tx for _ in range(n_rx)]
            for i, (_, rx_pos, f_rx) in enumerate(rx_list):
                for j, (_, tx_pos, f_tx) in enumerate(tx_list):
                    d = math.dist(pos3d[rx_pos], pos3d[tx_pos])
                    coeff[i][j] = predict_ks(f_rx, f_tx, d, max_d)
            self.cached_ks = coeff

        dialog = QtWidgets.QDialog(self)
        dialog.setWindowTitle("Матрица коэффициентов связи")
        dialog.resize(1500, 950)
        outer = QtWidgets.QVBoxLayout(dialog)

        table = QtWidgets.QTableWidget(dialog)
        table.setRowCount(len(rx_list))
        table.setColumnCount(len(tx_list))
        table.setVerticalHeaderLabels([str(r[0]) for r in rx_list])
        table.setHorizontalHeaderLabels([str(t[0]) for t in tx_list])

        for i in range(len(rx_list)):
            for j in range(len(tx_list)):
                val = coeff[i][j]
                text = "-" if val is None else f"{val:.2f}"
                item = QtWidgets.QTableWidgetItem(text)
                item.setTextAlignment(QtCore.Qt.AlignCenter)
                table.setItem(i, j, item)
```

```
        table.horizontalHeader().setStyleSheet("""
            QHeaderView::section {
                border-bottom: 2px solid #444;
                border-right: 1px solid #ccc;
                padding: 4px;
                background-color: #f0f0f0;
            }
        """)
        table.verticalHeader().setStyleSheet("""
            QHeaderView::section {
                border-right: 2px solid #444;
                border-bottom: 1px solid #ccc;
                padding: 4px;
                background-color: #f0f0f0;
            }
        """)
        table.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Stretch)
        table.verticalHeader().setSectionResizeMode(QtWidgets.QHeaderView.ResizeToContents)

        grid = QtWidgets.QGridLayout()
        lbl_tx = QtWidgets.QLabel("Передатчик", dialog)
        font_tx = lbl_tx.font()
        font_tx.setPointSize(14)
        font_tx.setBold(True)
        lbl_tx.setFont(font_tx)
        lbl_tx.setAlignment(QtCore.Qt.AlignCenter)

        lbl_rx = QtWidgets.QLabel("Приемник", dialog)
        font_rx = lbl_rx.font()
        font_rx.setPointSize(14)
        font_rx.setBold(True)
        lbl_rx.setFont(font_rx)
        lbl_rx.setAlignment(QtCore.Qt.AlignCenter)

        grid.addWidget(lbl_tx, 0, 1, alignment=QtCore.Qt.AlignCenter)
        grid.addWidget(lbl_rx, 1, 0, alignment=QtCore.Qt.AlignCenter)
        grid.addWidget(table, 1, 1)
        grid.setColumnStretch(0, 0)
        grid.setColumnStretch(1, 1)
        grid.setRowStretch(0, 0)
        grid.setRowStretch(1, 1)

        outer.addLayout(grid)

        btn_layout = QtWidgets.QHBoxLayout()
        btn_layout.addStretch()
        btn_coords = QtWidgets.QPushButton("Координаты размещения", dialog)
        btn_coords.clicked.connect(self.show_coordinates)
        btn_close = QtWidgets.QPushButton("Закрыть", dialog)
        btn_close.clicked.connect(dialog.accept)
        btn_layout.addWidget(btn_coords)
        btn_layout.addWidget(btn_close)

        outer.addLayout(btn_layout)
        dialog.exec_()

    def show_coordinates(self):
        dialog = QtWidgets.QDialog(self)
        dialog.setWindowTitle("Координаты размещения антенн")
        dialog.resize(600, 400)

        layout = QtWidgets.QVBoxLayout(dialog)
```

```
        table = QtWidgets.QTableWidget(dialog)
        table.setRowCount(len(self.placed_antennas))
        table.setColumnCount(3)
        table.setHorizontalHeaderLabels(["Координаты X", "Координаты Y", "Координаты Z"])

        for i, (ant_id, _, _, x, y, z) in enumerate(self.placed_antennas):
            table.setItem(i, 0, QtWidgets.QTableWidgetItem(f"{x:.2f}"))
            table.setItem(i, 1, QtWidgets.QTableWidgetItem(f"{y:.2f}"))
            table.setItem(i, 2, QtWidgets.QTableWidgetItem(f"{z:.2f}"))
            for j in range(3):
                item = table.item(i, j)
                item.setTextAlignment(QtCore.Qt.AlignCenter)

        ids = [str(ant_id) for ant_id, _, _, _, _, _ in self.placed_antennas]
        table.setVerticalHeaderLabels(ids)
        table.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

        layout.addWidget(table)

        btn_close = QtWidgets.QPushButton("Закрыть", dialog)
        btn_close.clicked.connect(dialog.accept)
        layout.addWidget(btn_close, alignment=QtCore.Qt.AlignRight)

        dialog.exec_()

class MainWindow(QtWidgets.QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Автоматизированное размещение антенн на летательном аппарате на основе нейронных
сетей")
        self.showMaximized()

        self.mlp_model = MLPRegressor(
            hidden_layer_sizes = (64, 32),
            activation='relu',
            solver='adam',
            random_state = 42,
            max_iter = 1
        )

        self.tabs = QtWidgets.QTabWidget()
        self.tabs.setTabsClosable(True)
        self.setCentralWidget(self.tabs)
        self.tabs.tabCloseRequested.connect(self.close_tab)

        self.init_menu()

        self.new_tab()

    def init_menu(self):
        menubar = self.menuBar()

        file_menu = menubar.addMenu("Файл")
        open_action    = QtWidgets.QAction("Открыть", self)
        save_action    = QtWidgets.QAction("Сохранить", self)
        save_as_action = QtWidgets.QAction("Сохранить как", self)
        new_tab_action = QtWidgets.QAction("Новая вкладка", self)

        file_menu.addAction(open_action)
        file_menu.addAction(save_action)
        file_menu.addAction(save_as_action)
```

```
        file_menu.addSeparator()
        file_menu.addAction(new_tab_action)

        open_action.triggered.connect(self.open_project)
        save_action.triggered.connect(self.save_project)
        save_as_action.triggered.connect(self.save_as_project)
        new_tab_action.triggered.connect(self.new_tab)

        database_menu = menubar.addMenu("База данных")

        add_antenna_action = QtWidgets.QAction("Добавить антенну", self)
        add_antenna_action.triggered.connect(self.open_add_antenna)
        database_menu.addAction(add_antenna_action)

        edit_antenna_action = QtWidgets.QAction("Редактировать антенну", self)
        edit_antenna_action.triggered.connect(self.open_edit_antenna)
        database_menu.addAction(edit_antenna_action)

        delete_antenna_action = QtWidgets.QAction("Удалить антенну", self)
        delete_antenna_action.triggered.connect(self.open_delete_antenna)
        database_menu.addAction(delete_antenna_action)

        add_aircraft_action = QtWidgets.QAction("Добавить БПЛА", self)
        add_aircraft_action.triggered.connect(self.open_add_aircraft)
        database_menu.addAction(add_aircraft_action)

        edit_aircraft_action = QtWidgets.QAction("Редактировать БПЛА", self)
        edit_aircraft_action.triggered.connect(self.open_edit_aircraft)
        database_menu.addAction(edit_aircraft_action)

        delete_aircraft_action = QtWidgets.QAction("Удалить БПЛА", self)
        delete_aircraft_action.triggered.connect(self.open_delete_aircraft)
        database_menu.addAction(delete_aircraft_action)


        help_menu = menubar.addMenu("Справка")
        author_action = QtWidgets.QAction("Автор", self)
        help_action   = QtWidgets.QAction("Помощь", self)

        help_menu.addAction(author_action)
        help_menu.addAction(help_action)

        author_action.triggered.connect(self.show_author_info)
        help_action.triggered.connect(self.show_help_info)

    def open_add_antenna(self):
        dialog = AddAntennaDialog(self)
        dialog.exec_()

    def open_edit_antenna(self):
        dialog = SelectAntennaDialog(self, mode="edit")
        dialog.exec_()

    def open_delete_antenna(self):
        dialog = SelectAntennaDialog(self, mode="delete")
        dialog.exec_()

    def open_add_aircraft(self):
        dialog = AddAircraftDialog(self)
        dialog.exec_()

    def open_edit_aircraft(self):
```

```
        dialog = SelectAircraftDialog(self, mode="edit")
        dialog.exec_()

    def open_delete_aircraft(self):
        dialog = SelectAircraftDialog(self, mode="delete")
        dialog.exec_()

    def save_project(self):
        current_widget = self.tabs.currentWidget()
        if not current_widget:
            return False

        desktop = os.path.join(os.path.expanduser("~"), "Desktop")
        filename = f"ЛА_{current_widget.la_id}.mars"
        path = os.path.join(desktop, filename)

        if os.path.exists(path):
            mb = QtWidgets.QMessageBox(self)
            mb.setWindowTitle("Файл уже существует")
            mb.setText(f"Файл «{filename}» уже есть на рабочем столе.\nЗаменить его?")
            btn_yes = mb.addButton("Да", QtWidgets.QMessageBox.YesRole)
            btn_no = mb.addButton("Нет", QtWidgets.QMessageBox.NoRole)
            btn_cancel = mb.addButton("Отмена", QtWidgets.QMessageBox.RejectRole)
            mb.exec_()

            if mb.clickedButton() == btn_cancel:
                return False
            if mb.clickedButton() == btn_no:
                return self.save_as_project()

        self._write_project_file(path, current_widget)
        current_widget.dirty = False
        QtWidgets.QMessageBox.information(self, "Сохранить", f"Проект сохранён:\n{path}")
        return True

    def save_as_project(self):
        current_widget = self.tabs.currentWidget()
        if not current_widget:
            return False

        path, _ = QtWidgets.QFileDialog.getSaveFileName(
            self, "Сохранить как", "", "Проект (*.mars);;Все файлы (*)"
        )
        if not path:
            return False

        if not path.lower().endswith(".mars"):
            path += ".mars"
        self._write_project_file(path, current_widget)
        current_widget.dirty = False
        QtWidgets.QMessageBox.information(self, "Сохранить как", f"Проект сохранён:\n{path}")
        return True

    def open_project(self):
        path, _ = QtWidgets.QFileDialog.getOpenFileName(
            self, "Открыть проект", "", "Проект (*.mars);;Все файлы (*)"
        )
        if not path:
            return

        with open(path, "r", encoding="utf-8-sig") as f:
            data = json.load(f)
```

```
        la_id        = data['la_id']
        length       = float(data.get('length', 0))
        radius       = float(data.get('radius', 0))
        material     = data.get('material', '')
        conductivity = float(data.get('conductivity', 0))
        permittivity = float(data.get('permittivity', 0))
        ant_data_raw = data.get('ant_data', [])
        placed       = data.get('placed', [])
        cached_ks    = data.get('cached_ks', None)

        tab = WorkTab(self, la_id, length, radius,
                      material, conductivity, permittivity, ant_data_raw)

        tab.placed_antennas = [tuple(x) for x in placed]
        tab.cached_ks = cached_ks
        if tab.placed_antennas:
            tab.predict_button.setEnabled(True)
            tab.remove_one_button.setEnabled(True)
            tab.remove_all_button.setEnabled(True)
            for entry in tab.placed_antennas:
                pos_idx = entry[2]
                tab.model_view.position_spheres[pos_idx].setVisible(False)

        tab.dirty = False

        idx = self.tabs.addTab(tab, f"ЛА {la_id}")
        self.tabs.setCurrentIndex(idx)

    def _write_project_file(self, path, widget):
        def safe_float(x):
            return float(x) if isinstance(x, (int, float, Decimal)) else x

        data = {
            'la_id':        widget.la_id,
            'length':       safe_float(widget.length),
            'radius':       safe_float(widget.radius),
            'material':     widget.material,
            'conductivity': safe_float(widget.conductivity),
            'permittivity': safe_float(widget.permittivity),
            'ant_data':     widget.ant_data,
            'placed':       widget.placed_antennas,
            'cached_ks':    widget.cached_ks
        }
        with open(path, "w", encoding="utf-8") as f:
            json.dump(data, f, ensure_ascii=False, indent=4)

    def new_tab(self):
        dlg = DataLoaderDialog(self)
        if dlg.exec_() == QtWidgets.QDialog.Accepted:
            la_id, length, radius, material, conductivity, permittivity, ant_data = dlg.get_selection()
            tab = WorkTab(
                self,
                la_id,
                length, radius,
                material, conductivity, permittivity,
                ant_data
            )
            idx = self.tabs.addTab(tab, f"ЛА {la_id}")
            self.tabs.setCurrentIndex(idx)
        else:
            if self.tabs.count() == 0:
```

```
            self.close()

    def close_tab(self, index):
        widget = self.tabs.widget(index)
        if widget and getattr(widget, 'dirty', False):
            mb = QtWidgets.QMessageBox(self)
            mb.setWindowTitle("Сохранение")
            mb.setText("Вкладка не сохранена. Сохранить перед закрытием?")
            btn_yes = mb.addButton("Да", QtWidgets.QMessageBox.YesRole)
            btn_no = mb.addButton("Нет", QtWidgets.QMessageBox.NoRole)
            btn_cancel = mb.addButton("Отмена", QtWidgets.QMessageBox.RejectRole)
            mb.exec_()

            if mb.clickedButton() == btn_cancel:
                return
            elif mb.clickedButton() == btn_no:
                self.tabs.removeTab(index)
                return
            else:
                if self.save_project():
                    self.tabs.removeTab(index)
                return

        self.tabs.removeTab(index)

    def show_author_info(self):
        QtWidgets.QMessageBox.information(
            self, "Автор",
            "Данное ПО разработано студентом КНИТУ-КАИ.\n"
            "В случае вопросов/предложений – напишите на:\n"
            "AmanullinMF@stud.kai.ru"
        )

    def show_help_info(self):
        QtWidgets.QMessageBox.information(
            self, "Помощь",
            "1. Файл → Новая вкладка → выбрать ЛА и антенны.\n"
            "2. Выбрать одну модель ЛА и одну или более антенн.\n"
            "3. Нажать «Загрузить».\n"
            "4. В окне размещения – нажать «Разместить», щёлкнуть правой кнопкой мыши по одной из подсвеченных
сфер.\n"
            "   Для завершения режима размещения нажмите «Завершить размещение».\n"
            "5. После хотя бы одной установки – «Прогноз коэффициента связи».\n"
            "6. Можно удалить одну или все антенны.\n"
            "7. Файл → Сохранить / Сохранить как / Открыть для работы с проектами.\n"
        )

def main():
    app = QtWidgets.QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())

if __name__ == "__main__":
    main()
```