Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

6

ПП.350000.000

Разраб.

Волкова Э.Ю.

Провер.

Медведева Т.А.

Н.контр

Попова О.А.

Утвердил

Долгов В.В.

Программная реализация математического моделирования развития эпидемиологической ситуации

Лист.

Листов

79

ДГТУ

Кафедра «ПОВТ и АС»

[Введение 7](#_Toc198770334)

[1 Математическое моделирование инфекционных заболеваний 9](#_Toc198770335)

[1.1 Типы эпидемиологических моделей 9](#_Toc198770336)

[1.2 Детерминистские популяционные модели 11](#_Toc198770337)

[1.3 Численные методы решения СДУ 17](#_Toc198770338)

[1.4 Постановка задачи 19](#_Toc198770339)

[1.5 Выводы по главе 19](#_Toc198770340)

[2 Алгоритмическое конструирование программного средства 20](#_Toc198770341)

[2.1 Структура программного средства 20](#_Toc198770342)

[2.5 Выводы по главе 22](#_Toc198770343)

[3 Программное конструирование 23](#_Toc198770344)

[3.1 Выбор языка программирования и среды разработки 23](#_Toc198770345)

[3.2 Используемые библиотеки 24](#_Toc198770346)

[3.3 Основные классы программного средства 26](#_Toc198770347)

[3.4 Выводы по главе 29](#_Toc198770348)

[4 Тестирование программного средства 30](#_Toc198770349)

[4.1 Инструкция по использованию программного средства 30](#_Toc198770350)

[4.2 Описание загрузки данных из файла 32](#_Toc198770351)

[4.3 Тестирование математических моделей 36](#_Toc198770352)

[4.4 Анализ результатов моделирования 46](#_Toc198770353)

[4.5 Выводы по главе 47](#_Toc198770354)

[Заключение 48](#_Toc198770355)

[Перечень используемых информационных источников 49](#_Toc198770356)

[Приложение А Техническое задание 51](#_Toc198770357)

[Приложение Б Исходный код программного средства 56](#_Toc198770358)

Введение

Математическое моделирование, или преобразование реальных процессов в математические уравнения, — это математическая концепция, которая имеет множество применений. Она применяется во многих областях науки таких как физика, экономика, социология и медицина. Данная работа рассматривает математическое моделирование в области медицинских наук в качестве инструмента для прогнозирования развития эпидемиологической ситуации.

Математические методы впервые были применены к изучению заболеваний в 1760 году Даниэлем Бернулли [2], который использовал их для оценки эффективности различных методов вакцинации от оспы.

В 1840 году Уильям Фарр успешно описал данные по смертности от оспы в Англии и Уэльсе за период с 1837 по 1839 год, используя нормальное распределение. Этот метод был развит Джоном Браунли, который в своей статье «Статистический подход к иммунной защите: теория эпидемий» (1906) сравнил ряды эпидемиологических данных на основе распределения Пирсона.

Хамер и Росс, используя математическое описание распространения заболеваний, смогли решить задачи по выяснению механизмов регулярного повторения эпидемии кори и установлению связи между количеством комаров и возникновением малярии. Их работы, наряду с исследованиями Росса и Хадсона, Сопера, а также Кермака и Маккендрика (1927), стали основой для дальнейших исследований в области математического моделирования эпидемий.

В этих работах впервые был применен «закон действующих масс», согласно которому количество вновь инфицированных прямо пропорционально произведению числа восприимчивых и инфицированных особей. Модель Кермака и Маккендрика положила начало широкому использованию детерминированных SIR-моделей («Susceptible — Infected — Recovered»), в которых с помощью систем дифференциальных или разностных уравнений описывается динамика групп восприимчивых, инфицированных и выздоровевших.

Рассматриваемые в данной работе модели могут спрогнозировать, как прогрессируют инфекционные заболевания. Модели используют базовые допущения или собранные статистические данные для нахождения параметров для различных инфекционных заболеваний, используя их для расчета эффекта различных вмешательств, таких как ввод карантина. [1]

Программная реализация рассматриваемых математических моделей позволяет провести численный анализ различных сценариев, влияние карантинных мер и других факторов. В данной работе рассматриваются и реализуются основные детерминированные модели развития эпидемий: SI, SIR, SIRS, SEIR, SIQR и MSEIR. Эти модели отличаются уровнем детализации и применяются в зависимости от характеристик конкретного заболевания и целей анализа.

# **1 Математическое моделирование инфекционных заболеваний**

В данном разделе описывается понятие математической модели. Рассматриваются конкретные модели для прогнозирования развития эпидемиологической ситуации и численные методы для решения СДУ.

* 1. Типы эпидемиологических моделей

Моделирование эпидемических процессов всегда предполагает использование упрощающих допущений относительно природы описываемого явления. Ключевым требованием к любой математической модели (будь то аппроксимационная, аналитическая, стохастическая или симуляционная) является адекватное отражение существенных связей в эпидемическом процессе с исключением несущественных и случайных факторов. Чем более полно модель охватывает ключевые взаимосвязи, тем выше ее соответствие реальному развитию конкретного эпидемического процесса.

1.1.1 Стохастические модели

Стохастическая модель — это инструмент для оценки распределения вероятностей потенциальных результатов с учётом случайных изменений одного или нескольких входных параметров с течением времени.

В начале XX века ученые-эпидемиологи пришли к выводу, что для описания распространения инфекционных заболеваний в ограниченных популяциях (например, семьях) применительно к болезням, таким как корь, более целесообразным является вероятностный подход, нежели детерминированный.

В 1926 году Маккендрик сформулировал стохастический вариант модели SIR с непрерывным временем для вывода уравнений, описывающих продолжительность эпидемии, применительно к гриппу и малярии. Однако работа Маккендрика не получила широкого признания.

Значительный вклад в развитие моделей с вероятностным описанием внесла работа Гринвуда 1931 года, посвященная изучению вспышки кори. Также важную роль сыграла модель Рида и Фроста, разработанная ими в период с 1928 по 1930 годы, но опубликованная лишь в 1952 году.

В моделях Гринвуда и Рида-Фроста описание количества инфицированных на каждом временном интервале привело к использованию цепи биномиальных распределений, что обусловило появление термина «цепочечно-биномиальные модели».

1.1.2 Детерминированные модели

Детерминированная модель (противоположна модели стохастической) — математическая модель, параметры и переменные которой зависят друг от друга функционально, т.е. не подчинены случайным колебаниям процесса, в связи с чем характер системы в любое время полностью определяется первоначально выбранными условиями.

При работе с большими группами населения, как в случае с туберкулезом, часто используются детерминированные или компартментальные математические модели. В детерминированной модели люди в популяции распределяются по разным подгруппам или компартментам, каждый из которых представляет собой определённую стадию эпидемии.

Скорости перехода из одного класса в другой математически выражаются как производные, поэтому модель формулируется с помощью дифференциальных уравнений [18]. При построении таких моделей необходимо предполагать, что численность населения в группе дифференцируема по времени и что процесс эпидемии является детерминированным. Другими словами, изменения численности населения в группе можно рассчитать, используя только историю, которая использовалась для разработки модели.

1.2 Детерминистские популяционные модели

В популяционных моделях население представляется как совокупность групп. Формирование этих групп обусловлено различным статусом индивидов относительно рассматриваемого заболевания (например, восприимчивые к заболеванию, инфицированные и выздоровевшие).

Важно отметить, что внутри каждой группы индивиды считаются идентичными по своим характеристикам. Изменения численности групп во времени обусловлены рядом следующих процессов:

* переходы индивидов из одной группы в другую вследствие инфицирования, развития заболевания, выявления и лечения заболевших индивидов;
* пополнение групп за счёт иммиграции и рождения индивидов;
* убыль в результате естественной смертности индивидуумов, гибели из–за болезни и эмиграции в другие регионы.

Среди моделей, используемых для описания динамики распространения инфекционных заболеваний в популяциях, наиболее распространены модели SIR. В этих моделях население подразделяется на три группы:

* S (Susceptible) – восприимчивые к заболеванию;
* I (Infected) – инфицированные;
* R (Recovered/Removed) – переболевшие или удаленные из популяции.

Передача инфекции происходит от инфицированных к восприимчивым. Индивиды, переболевшие болезнью, приобретают иммунитет и не могут быть инфицированы повторно.

Математически модели SIR описываются системами дифференциальных уравнений (для непрерывного времени) или разностными уравнениями (для дискретного времени). Эти уравнения отражают изменение численности каждой группы с течением времени.

1.2.1 Классическая SIR-модель

SIR-модель – одна из самых базовых и популярных моделей. Она делит популяцию на три группы:

* S (Susceptible) - восприимчивые к заболеванию;
* I (Infected) - инфицированные;
* R (Recovered/Removed) - переболевшие или удалённые из популяции.

Схема модели SIR показана рисунке 1.1.



Рисунок 1.1 – SIR-модель

Данная модель предложена в 1927 году Уильямом Огильви Кермаком и Эндерсоном Грейем МакКендриком. Она стала классическим фундаментом математической эпидемиологии, на основе которой разработаны и предложены модифицированные модели, рассматривающие дополнительные группы популяции. [2]

Модель описывает изменение численности групп во времени с помощью системы дифференциальных уравнений. Например, стандартной SIR-модели соответствует система (1.1).

, (1.1)

где – скорость передачи инфекции;

– скорость выздоровления;

, , – неизвестные функции, зависящие от времени.

1.2.2 SI-модель

SI-модель является упрощенной версией классической SIR-модели, где отсутствует стадия выздоровления, то есть все, кто заражается, остаются инфицированными. Схема данной модели показана на рисунке 1.2.



Рисунок 1.2 – SI-модель

Появилась данная модель в начале XX века, основаная на работах У. Х. Хаммерса (1920) и Нильса Кристиана Бергера (1932), как часть общего подхода к эпидемиологическому моделированию. SI-модель может использоваться для описания хронических инфекций, таких как ВИЧ. Данной модели соответствует система (1.2).

, (1.2)

где – скорость передачи инфекции;

– скорость выздоровления.

1.2.3 SIRS-модель

SIRS-модель разработана как расширение SIR-модели для инфекций, при которых иммунитет не является постоянным (например, грипп). Схема данной модели показана на рисунке 1.3.

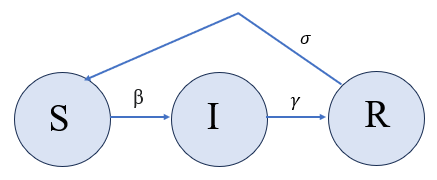


Рисунок 1.3 – SIRS-модель

SIRS-модель применялась активно в 1950–1970-х годах. В этой модели индивиды, выздоровевшие и перешедшие в группу 𝑅, могут потерять иммунитет и снова стать восприимчивыми. Данной модели соответствует система (1.3).

, (1.3)

где – скорость передачи инфекции;

– скорость выздоровления;

– скорость потери иммунитета.

1.2.4 SIQR-модель

SIQR-модель разработана в конце XX — начале XXI века на фоне необходимости учитывать карантинные меры в условиях эпидемий, включающая дополнительное состояние – Q (Quarantined). Схема данной модели представлена на рисунке 1.4.



Рисунок 1.4 – SIQR-модель

Наиболее эффективна при моделировании таких инфекций как COVID-19, SARS, туберкулез, холера и другие заболевания, при которых карантин влияет на распространение. Данной модели соответствует система (1.4).

, (1.4)

где – скорость передачи инфекции;

– скорость выздоровления;

– скорость изоляции инфицированных людей (помещения на карантин);

– скорость выздоровления людей, находящихся на карантине;

– неизвестная функция, зависящая от времени.

1.2.5 SEIR-модель

SEIR-модель включает дополнительное состояние – E (Exposed), или «латентную» фазу, когда человек инфицирован, но еще не заразен. Схема данной модели представлена на рисунке 1.5.



Рисунок 1.5 – SEIR-модель

Модель SEIR полезна для описания инфекций с инкубационным периодом, таких как COVID-19. Предложена в 1930-х–1950-х годах как усовершенствование модели SIR. Активно применялась во второй половине XX века для анализа заболеваний с инкубационным периодом. Данной модели соответствует система (1.5).

, (1.5)

где – скорость передачи инфекции;

– скорость выздоровления;

– обратная величина инкубационного периода;

– неизвестная функция, зависящая от времени.

1.2.6 MSEIR-модель

Данная модель является расширением SEIR-модели, в которой добавляется группа M (Maternal immunity), представляющая новорожденных с пассивным иммунитетом, получивших антитела от матери. Схема данной модели показана на рисунке 1.6.



Рисунок 1.6 – MSEIR-модель

MSEIR-модель наиболее эффективна при моделировании таких инфекций как корь, краснуха, коклюш и другие детские заболевания, при которых материнский иммунитет играет важную роль в первые месяцы жизни младенца. Данной модели соответствует система (1.6).

, (1.6)

где – скорость передачи инфекции;

– скорость выздоровления;

– скорость перехода из инкубационного периода в инфекционный;

– естественная смертность/рождаемость;

– скорость потери материнского иммунитета;

– неизвестная функция, зависящая от времени;

– общее количество населения.

1.3 Численные методы решения СДУ

Представленные в данной работе модели являются системами дифференциальных уравнений и в большей своей массе не имеют аналитического решения их необходимо решать численными методами.

Основными инструментами для численного решения являются:

* метод Эйлера: простой в применении, но обладающий ограниченной точностью.
* метод Рунге-Кутты 4-го порядка: более точный и универсальный метод, применимый к широкому спектру задач.

Данные методы позволяют получить приближенное представление о динамике эпидемии во времени, учитывая различные параметры модели.

1.3.1 Метод Эйлера

Метод Эйлера представляет собой один из простейших численных методов первого порядка, предназначенных для приближенного решения систем обыкновенных дифференциальных уравнений. В его основе лежит идея аппроксимации значения функции в следующий момент времени на основе значения ее производной в текущий момент, представленное в формуле (1.7).

, (1.7)

где – значение функции в текущий момент времени;

– шаг интегрирования;

– правая часть дифференциального уравнения;

– текущий момент времени.

1.3.2 Метод Рунге-Кутта 4-го порядка

Метод Рунге-Кутта 4-го порядка – это распространенный метод численного интегрирования, обеспечивающий баланс между точностью и вычислительной эффективностью. В основе алгоритма лежит вычисление четырех промежуточных значений, используемых для аппроксимации решения дифференциального уравнения представленных в формулах 1.8-1.12.

,(1.8)

,(1.9)

,(1.10)

,(1.11)

,(1.12)

где – значение функции в текущий момент времени;

– шаг интегрирования;

– правая часть дифференциального уравнения;

– текущий момент времени.

1.4 Постановка задачи

Целью данной работы является разработка программной реализации классических и модифицированных моделей математической эпидемиологии с их решением с помощью численных методов.

Для достижения поставленной цели решены следующие задачи:

1. Проведен теоретический анализ существующих моделей распространения инфекционных заболеваний.
2. Сформулированы математические модели (в виде систем обыкновенных дифференциальных уравнений) для каждой рассматриваемой модели.
3. Реализовано программное решение моделей с использованием численного метода Рунге-Кутты 4-го порядка и метода Эйлера.
4. Разработан пользовательский интерфейс, обеспечивающий ввод параметров и визуализацию результатов моделирования.
5. Проведен сравнительный анализ поведения различных моделей при идентичных начальных условиях.
6. Исследованы влияние ключевых параметров (скорость заражения, инкубационный период, миграция и т.п.) на динамику эпидемии.

Данная работа представляет собой создание инструмента для анализа эпидемических ситуаций и возможности принятия решений в сфере общественного здравоохранения.

1.5 Выводы по главе

В данной главе рассмотрены типы математических моделей, проведен теоретический обзор классической SIR-модели и её модификаций. Рассмотрены численные методы для решения СДУ.

2 Алгоритмическое конструирование программного средства

В данном разделе описано алгоритмическое конструирование разработанного программного средства. Приводятся схемы используемых численных методов для решения СДУ.

2.1 Структура программного средства

В графическом интерфейсе разработанного программного средства можно выбрать одну или несколько моделей, ввести для каждой начальные данные, выбрать численный метод для решения ОДУ и экспортировать результаты моделирования в файл формата .xlsx. Описанная структура программного средства показана на рисунке 2.1.

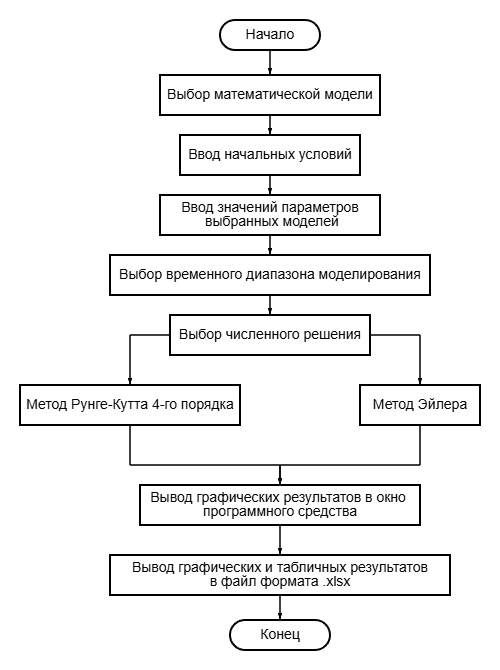


Рисунок 2.1 – Схема программного средства

2.2.1 Структура метода Эйлера

Для метода Эйлера необходимы следующие входные данные: выбранная математическая модель, временной диапазон, начальные условия и параметры модели. Структура метода Эйлера для решения представленных в данной работе СДУ представлена на рисунке 2.2.



Рисунок 2.2 – Схема метода Эйлера

2.2.2 Структура метода Рунге-Кутта 4-го порядка

Для более точного численного решения СДУ с помощью метода Рунге-Кутта 4-го порядка необходимы следующие входные данные: выбранная математическая модель, начальные условия, временной диапазон и параметры модели. Структура метода Рунге-Кутта 4-го для решения представленных в данной работе СДУ представлена на рисунке 2.3.



Рисунок 2.3 – Схема метода Рунге-Кутта 4-го порядка

2.5 Выводы по главе

В данной главе рассмотрена общая схема программного средства, а также схемы численных методов, используемых для решения систем дифференциальных уравнений. Данные схемы используются для дальнейшего решения поставленной задачи.

3 Программное конструирование

В данном разделе обоснован выбор языка программирования, среды разработки и используемых в программном средстве дополнительных библиотек. Описаны основные классы и методы для выполнения поставленной задачи.

* 1. Выбор языка программирования и среды разработки

Для разработки программного средства выбран язык программирования Python версии 3.13.1.

Python – это язык программирования со множеством парадигм, включая императивное, процедурное, структурное, функциональное, асинхронное и объектно-ориентированное программирование, а также метапрограммирование. Задачи обобщенного программирования решаются благодаря динамической типизации. Некоторые аспекты аспектно-ориентированного программирования поддерживаются через использование декораторов, в то время как более глубокая интеграция доступна с помощью дополнительных фреймворков. Реализация таких методик, как контрактное и логическое программирование, возможна с использованием библиотек или расширений. К основным архитектурным особенностям Python относятся динамическая типизация, автоматическое управление памятью, механизм обработки исключений, поддержка многопоточных вычислений с глобальной блокировкой интерпретатора (GIL) и высокоуровневые структуры данных. Язык позволяет разбивать программы на модули, которые могут быть объединены в пакеты. [13]

Python доступен практически на всех известных платформах, от карманных ПК до мейнфреймов. Существуют версии для Microsoft Windows, большинства дистрибутивов UNIX (включая FreeBSD и Linux), Android, Plan 9, Mac OS и macOS, iPhone OS (iOS) версии 2.0 и выше, iPadOS, Palm OS, OS/2, Amiga, HaikuOS, AS/400, OS/390, Windows Mobile и Symbian.

В отличие от многих портируемых систем, Python обеспечивает поддержку технологий, характерных для конкретных платформ (например, Microsoft COM/DCOM). Кроме того, существует специализированная версия Python для виртуальной машины Java под названием Jython, что позволяет исполнять интерпретатор на любой системе, поддерживающей Java, при этом классы Java могут быть использованы непосредственно в Python и написаны на нем. Также есть несколько проектов, обеспечивающих интеграцию с платформой Microsoft .NET, среди которых основными являются IronPython и Python.NET.Средой разработки был выбран Visual Studio Code, т.к. включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса и средства для факторинга. Распространяется бесплатно и позиционируется как «лёгкий» редактор кода.

Для разработки программного средства выбран Visual Studio Code (VS Code) – бесплатный, кроссплатформенный редактор кода от Microsoft, доступный для Windows, Linux и macOS. VS Code, позиционируемый как легковесный инструмент, идеально подходит для веб- и облачной разработки. Он обладает встроенными функциями, такими как отладчик, поддержка Git, подсветка синтаксиса, IntelliSense и инструменты рефакторинга. С 2018 года доступно расширение Python с открытым исходным кодом, значительно расширяющее возможности редактирования, отладки и тестирования Python-кода.

* 1. Используемые библиотеки

Разработанное программное средство содержит несколько библиотек, позволяющих вводить даты с помощью удобного календаря, выводить на экран пользователя графики, выполнять численные операции, импортировать данных из файлов в программу и экспортировать результаты вычислений в файл определенного формата. Используемые в программном средстве библиотеки представлены в таблице 3.1.

Таблица 3.1 – Используемые библиотеки

|  |  |  |
| --- | --- | --- |
| Название библиотеки | Описание | Назначение |
| Tkinter | Событийно-ориентированная графическая библиотека на основе средств Tk (широко распространённая в мире GNU/Linux и других UNIX‐подобных систем, портирована также и на Microsoft Windows). Входит в стандартную библиотеку Python. | Используется для реализации пользовательского интерфейса. |
| Tkcalendar | Модуль Python, предоставляющий виджеты Calendar и DateEntry для Tkinter. | Используется для удобного ввода начальной и конечной даты моделирования эпидемиологической ситуации. |
| Datetime | Модуль, предоставляющий классы для работы с датами и временем. | Используется для получения текущей даты с календаря компьютера. |
| Matplotlib | Библиотека на языке программирования Python для визуализации данных двумерной и трёхмерной графикой. | Используется для создания графических результатов моделирования и вывода их в пользовательский интерфейс. |
| NumPy | Библиотека с открытым исходным кодом для языка программирования Python, включающая поддержку многомерных массивов (включая матрицы) и высокоуровневых математических функций, предназначенных для работы с многомерными массивами. | Используется для создания пустых массивов и временного ряда. |
| pandas | Программная библиотека на языке Python для обработки и анализа данных. Предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами. | Используется для чтения файлов формата .csv и корректной выгрузки данных в программное средство для дальнейшей обработки. |
| zipfile | Данный модуль работаеь с ZIP-файлами, использующими расширения ZIP64. Он поддерживает расшифровку зашифрованных файлов в ZIP-архивах. | Используется для создания архива, содержащего файлы формата .xlsx с результатами по каждой выбранной модели. |

3.3 Основные классы программного средства

Для реализации программного средства выбрана объектно-ориентированная парадигма. В процессе программного конструирования были выделены следующие классы: класс численных методов NumericalMethods, класс математических моделей EpidemicModels и главный класс программного средства EpidemicApp.

NumericalMethods представляет собой класс, содержащий численные методы Эйлера и Рунге-Кутта 4-го порядка для решения систем дифференциальных уравнений. Данный класс содержит методы, приведенные в таблице 3.1.

Таблица 3.1 **–** Методы класса NumericalMethods

|  |  |
| --- | --- |
| Название метода | Функциональное назначение метода |
| euler\_method | метод для нахождения решения системы дифференциальных уравнений с помощью метода Эйлера |
| runge\_kutta\_4 | метод для нахождения решения системы дифференциальных уравнений с помощью метода Рунге-Кутта 4-го порядка |

Класс EpidemicModels содержит методы, реализующие математические модели развития эпидемиологических ситуаций, представленные в данной работе. Содержит методы, приведенные в таблице 3.2.

Таблица 3.2 **–** Методы класса EpidemicModels

|  |  |
| --- | --- |
| Название метода | Функциональное назначение метода |
| sir\_model | метод, реализующий логику SIR-модели |
| si\_model | метод, реализующий логику SI-модели |
| sirs\_model | метод, реализующий логику SIRS-модели |
| siqr\_model | метод, реализующий логику SIQR-модели |
| seir\_model | метод, реализующий логику SEIR-модели |
| mseir\_model | метод, реализующий логику MSEIR-модели |
| run\_si\_model | метод, запускающий SI-модель на указанном диапазоне времени |
| run\_sir\_model | метод, запускающий SIR-модель на указанном диапазоне времени |
| run\_sirs\_model | метод, запускающий SIRS-модель на указанном диапазоне времени |
| run\_seir\_model | метод, запускающий SEIR-модель на указанном диапазоне времени |
| run\_mseir\_model | метод, запускающий MSEIR-модель на указанном диапазоне времени |
| run\_siqr\_model | метод, запускающий SIQR-модель на указанном диапазоне времени |

Класс EpidemicApp содержит методы, реализующие логику работы всей программы, а также загрузку и данных из файла в программу и экспорт результатов в Excel файл. Методы данного класса создают главное окно программного средства, осуществляют ввод данных и запуск моделирования. Содержит методы, приведенные в таблице 3.3.

Таблица 3.3 **–** Методы класса EpidemicApp

|  |  |
| --- | --- |
| Название метода | Функциональное назначение метода |
| create\_widgets | метод создания основного интерфейса |
| create\_models\_tab | метод создания вкладки выбора моделей |
| create\_params\_tab | метод создания вкладки для ввода параметров и начальных условий |
| create\_data\_tab | метод создания вкладки для загрузки и экспорта данных |
| create\_graphs | метод настройки графиков для отображения результатов |
| create\_validate\_func | метод для валидации вводимых данных |
| set\_default\_values | метод, устанавливающий значения по умолчанию |
| update\_model\_selection | метод, ограничивающий и проверяющие количество выбранных моделей |
| run\_simulation | метод, запускающий прогнозирование на основе выбранных моделей |
| load\_csv\_data | метод, загружающие начальные данные из CSV файла |
| process\_csv\_data | метод, обрабатывающий загруженные данные |
| export\_results | метод, экспортирующий результаты моделирование в Excel файл(ы) |

UML-диаграмма классов программного средства изображена на рисунке 3.1.

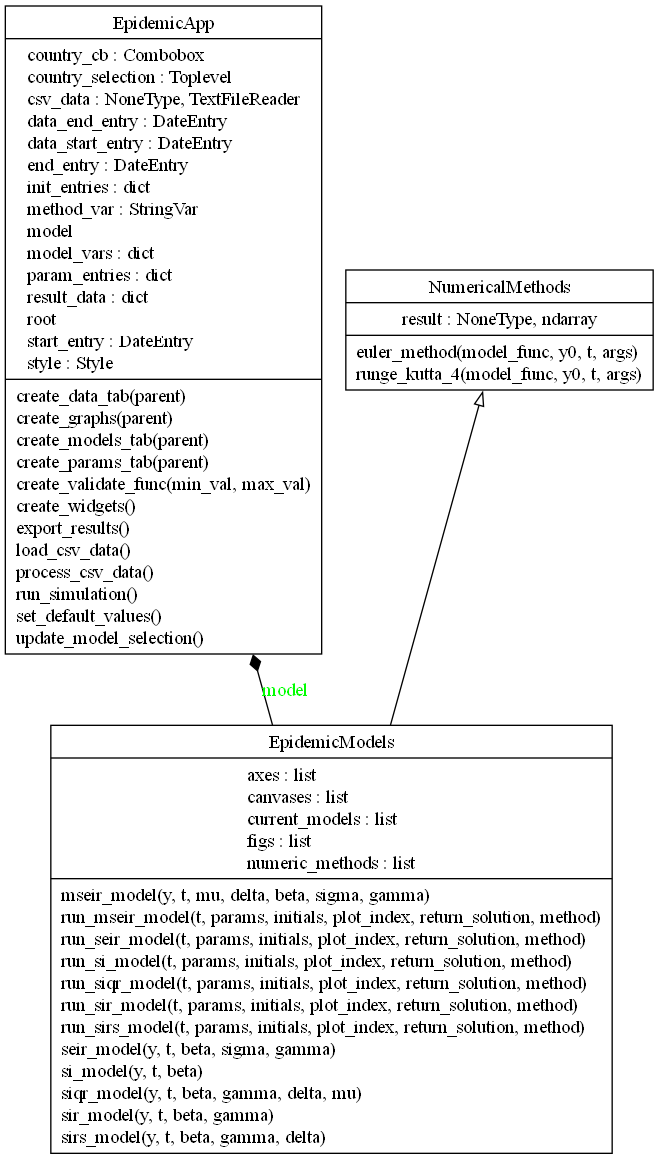
****

Рисунок 3.1 – UML-диаграмма классов программного средства

Полный листинг программы приведен в Приложении В.

3.4 Выводы по главе

В данном разделе обоснован выбор языка программирования и среды разработки для реализации программного средства. Описаны используемые дополнительные библиотеки. Разработаны основные классы и методы программы.

4 Тестирование программного средства

В данном разделе описаны и протестированы контрольные примеры с подробным описанием входных данных и подтверждающими работу программного средства изображениями. Приводится инструкция по использованию программного средства. Проведен анализ результатов моделирования.

4.1 Инструкция по использованию программного средства

Разработанное программное средство предназначено для моделирования развития эпидемиологической ситуации с помощью представленных в данной работе математических моделей: SIR, SI, SIRS, SIQR, SEIR, MSEIR и M-модель. Для начала работы с программой пользователю необходимо выбрать 1-4 модели, выбрать начальную и конечную дату моделирования, а также ввести начальные значения долей населения и значение параметров выбранных моделей. Интерфейс пользователя поддерживает вариант, при котором пользователь может загрузить в программу файл со своими данными.

Запуск моделирования осуществляется через нажатие кнопки «Запустить моделирование». После нажатия в окне программного средства появятся графики выбранных моделей на указанном отрезке времени. Пользователь может сохранить результат моделирования в формате Excel файла, нажав на кнопку «Экспорт в файл». Создаться архив, в котором будут сохранены по отдельности результаты каждой модели. В сохраненных Excel файлах будет 4 листа:

* «Параметры» - лист, в котором сохранены введенные пользователем значения для параметров модели;
* «Начальные условия» - лист, в котором сохранены введенные пользователем начальные значения для долей населения;
* «Решение» - лист, в котором сохранено по дням решение математической модели с помощью численного метода;
* «График» - лист, содержащий результирующий график моделирования развития эпидемиологической ситуации.

Начальное состояние интерфейс программного средства представлен на рисунке 4.1.

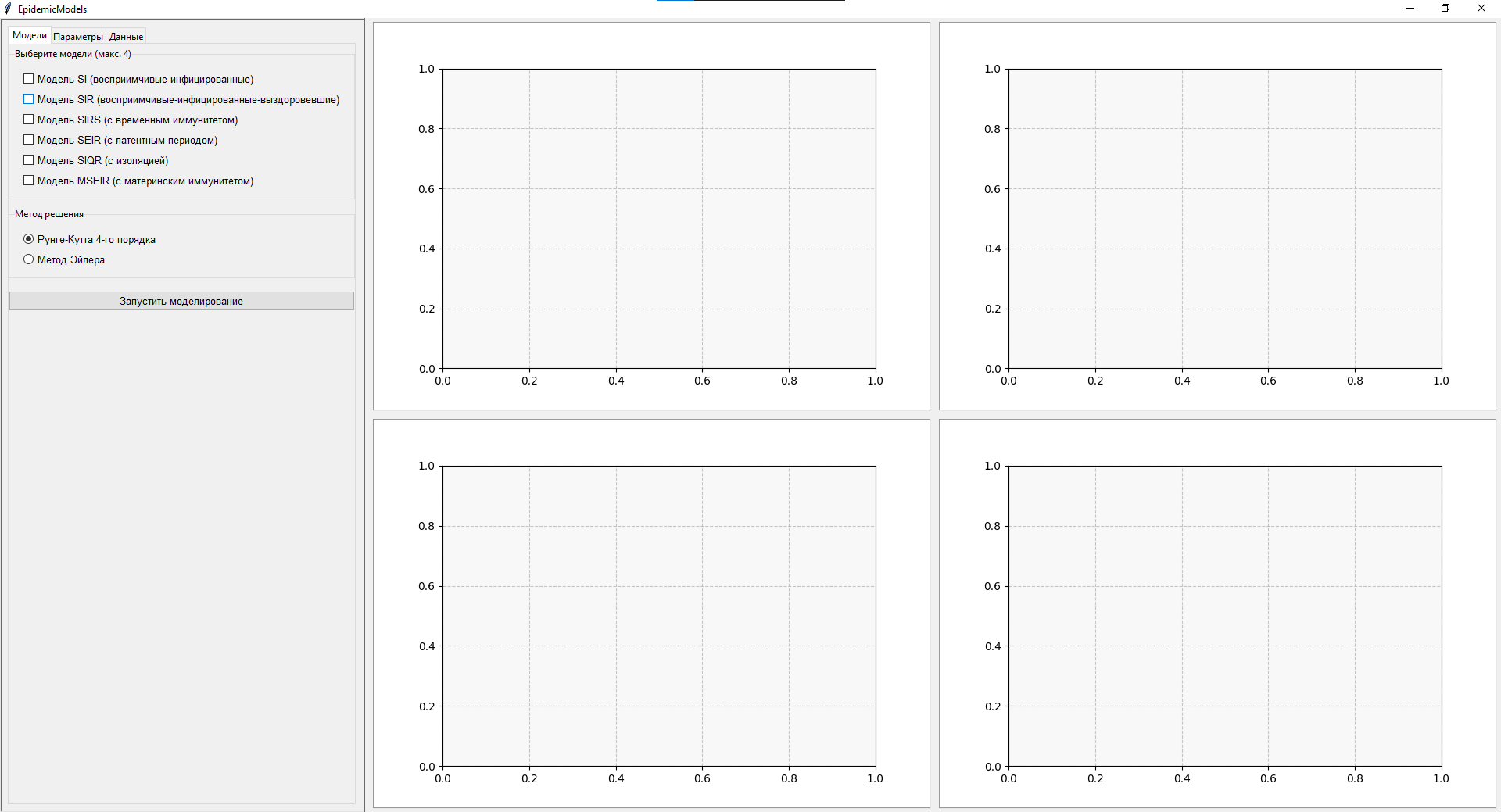


Рисунок 4.1 – Начальное состояние

Конечное состояние интерфейса программного средства показано на рисунке 4.2.

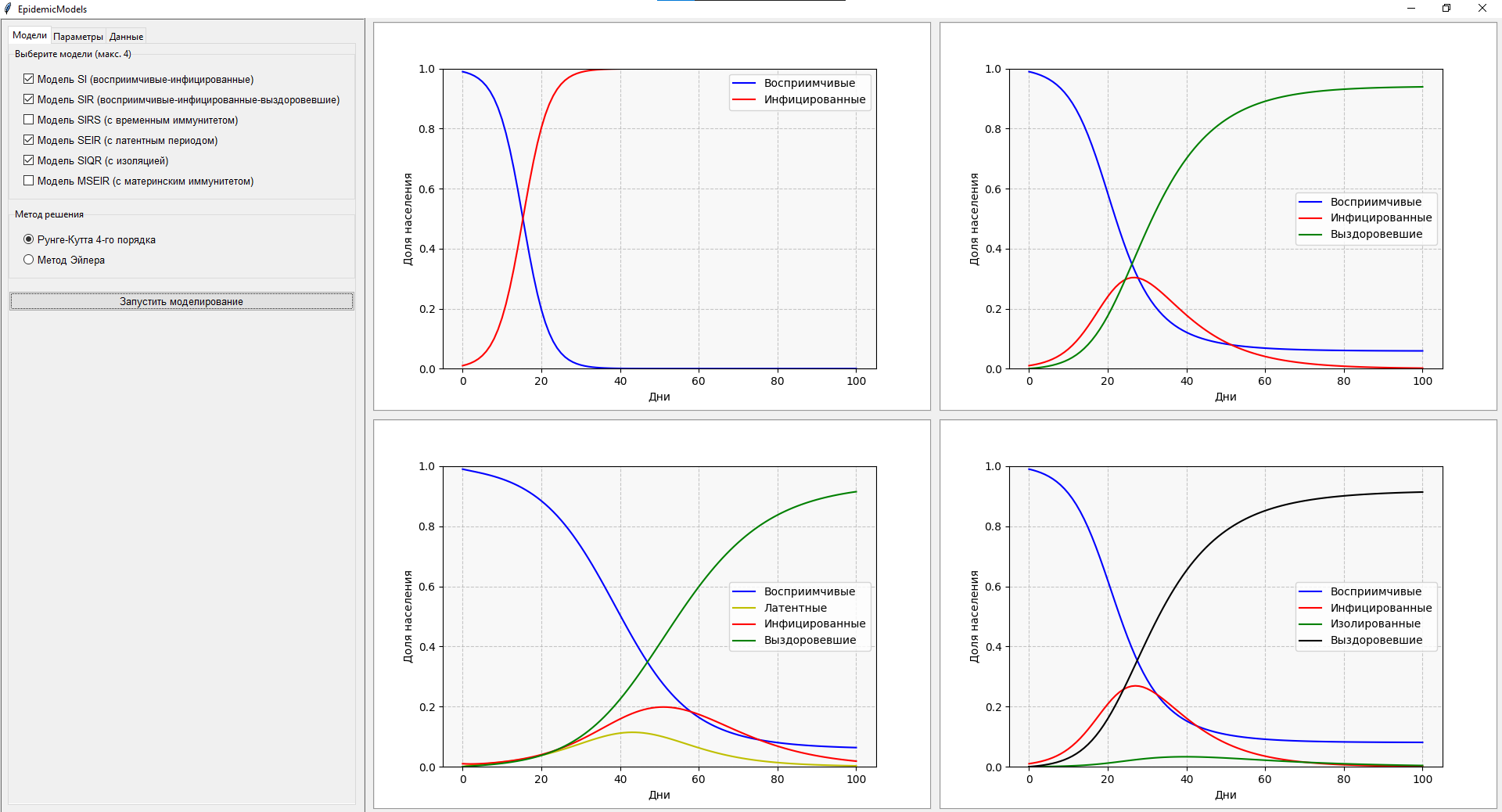


Рисунок 4.2 – Конечное состояние

4.2 Описание загрузки данных из файла

Для загрузки заранее составленных данных для моделирования необходимо перейти на вкладку «Данные» и нажать на кнопку «Загрузить данные из CSV», что показано на рисунке 4.3.



Рисунок 4.3 – Раздел загрузки данных из файла

После этого программа попытается выгрузить необходимые данные из файла: даты, количество населения, восприимчивые, инфицированные и выздоровевшие. Если данные из файла не были успешно выгружены, то на экран пользователя будет выведено уведомление об ошибке с указанием отсутствующего столбца в файле, представленное на рисунке 4.4.



Рисунок 4.4 – Ошибка загрузки данных

Если данные загружены успешно, то программное средство выведен на экран пользователя окно выбора страны, если в файле находятся несколько стран, а также диапазон дат, информация за которые будет считаться начальными данными для моделирования. Данное окно показано на рисунке 4.5.



Рисунок 4.5 – Окно выбора страны и дат

После нажатия на кнопку «Загрузить данные» на экране пользователя появиться уведомление об успехе операции с указанием страны и периода, выбранных пользователем. Пример такого уведомление представлено на рисунке 4.6.



Рисунок 4.6 – Уведомление об успехе загрузки данных

Загруженные данные по долям населения нормализуются в диапазоне от нуля до единицы и распределяются на S (восприимчивых), I (инфицированных) и R (выздоровевших). Пример распределения начальных значений изображено на рисунке 4.7.

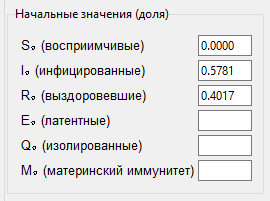


Рисунок 4.7 – Распределение долей населения

Дата начала прогнозирования автоматически определяется как конечная дата выбора начальных данных из файла. Конечная дата прогнозирования отличается от начальной на 3 месяца, что показано на рисунке 4.8.

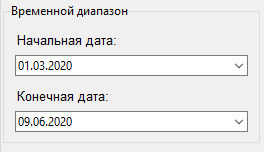


Рисунок 4.8 – Временной диапазон прогнозирования

Пользователь может заранее или после загрузки данных из CSV файла выбрать начальную и конечную дату прогнозирования, примеры которых представлены на рисунках 4.9 и 4.10.

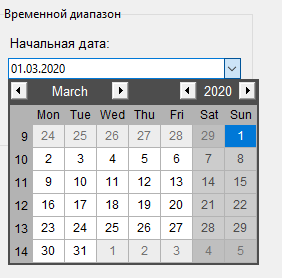


Рисунок 4.9 – Выбор начальной даты

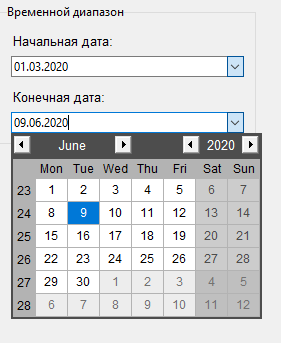


Рисунок 4.10 – Выбор конечной даты

Пользователь может заранее или после загрузки данных из CSV файла ввести значения параметров модели, либо оставить значения по умолчанию, что показаны на рисунке 4.11.

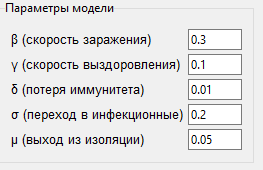


Рисунок 4.11 – Значения параметров по умолчанию

Пользователь может заранее или после загрузки данных из CSV файла выбрать от одной до четырех математических моделей. Если этот выбор превышает ограничение, то на экран будет выведено сообщение о превышении лимита, изображенное на рисунке 4.12.



Рисунок 4.12 – Предупреждение об превышении лимита

Пользователь также может заранее или после выгрузки данных выбрать численный метод: Рунге-Кутта 4-го порядка или Эйлера. Чтобы начать моделирование пользователю необходимо нажать на кнопку «Запустить моделирование».

4.3 Тестирование математических моделей

Поскольку представленные в данной работе модели отличаются друг от друга как параметрами, так и начальными условиями, то каждую из них необходимо тестировать отдельно. Во всех контрольных примерах берется временной диапазон в 60 дней, начальная дата которого 15.05.2025, а конечная 14.07.2025. Все контрольные примеры решаются с помощью численного метода Рунге-Кутта 4-го порядка.

4.3.1 SIR-модель

SIR-модель состоит из трех начальных значений (восприимчивые, инфицированные и выздоровевшие) и двух параметров (скорость заражения и скорость выздоровления). Для первого контрольного примера вводятся следующие начальные данные: , , , , . Результат прогнозирования для введенных значений показан на рисунке 4.13.

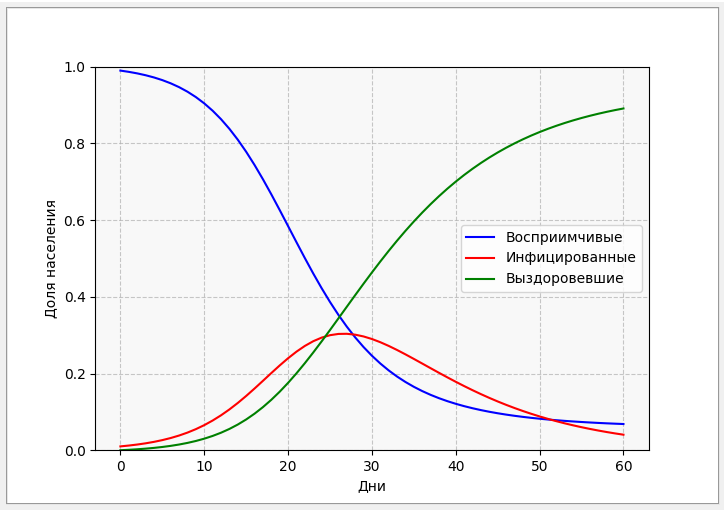


Рисунок 4.13 – Результат SIR-модели для первого контрольного примера с помощью метода Рунге-Кутта 4-го порядка

Для второго контрольного примера вводятся следующие начальные данные: , , , , . Результат прогнозирования для данного примера представлен на рисунке 4.14.

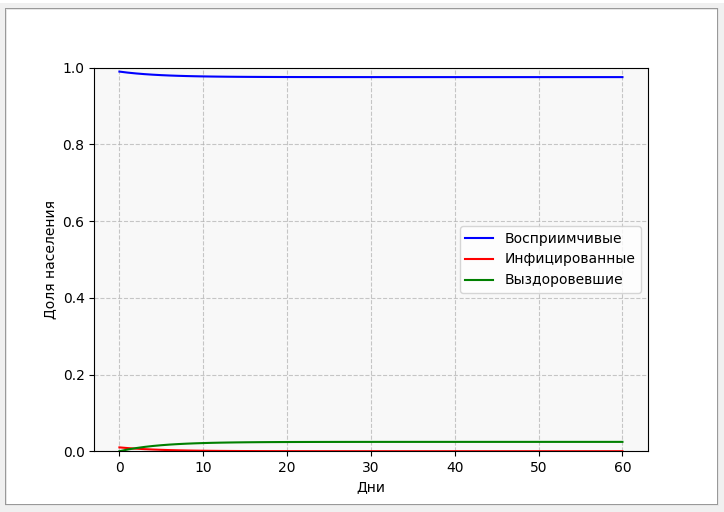


Рисунок 4.14 – Результат SIR-модели для второго контрольного примера с помощью метода Рунге-Кутта 4-го порядка

На графике видно, что инфицированные выздоравливают быстрее, а так как начальное количество данной доли население мало, то эпидемия заканчивается в первые же дни своего появления. Увеличим долю инфицированных до 0.25, а восприимчивых уменьшим до 0.75, оставив остальные параметры без изменения. Результат прогнозирования с измененными данными показан на рисунке 4.15.

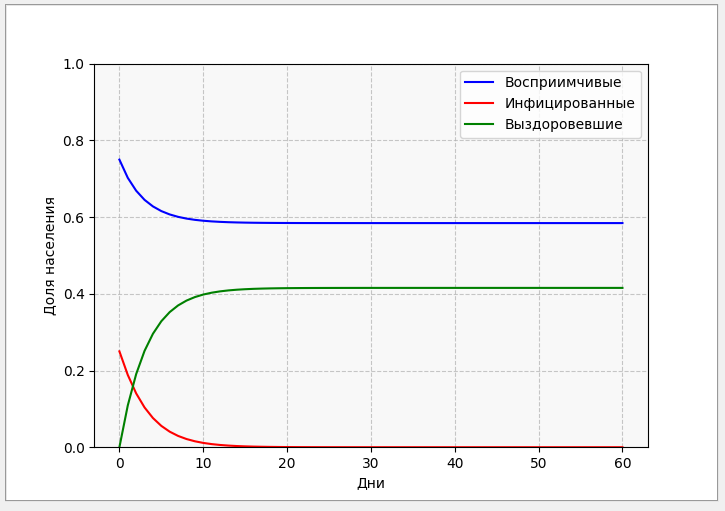


Рисунок 4.15 – Результат прогнозирования с измененными начальными данными для второго контрольного примера

На графике можно увидеть, что несмотря на увеличение количества восприимчивых, эпидемия все равно заканчивается в первый десяток дней. Пересечение графиков, выздоровевших и инфицированных примерно похоже на пересечение этих же графиков в предыдущем варианте второго контрольного примера.

4.3.2 SI-модель

SI-модель состоит из двух начальных условий (восприимчивые и инфицированные) и одного параметра (скорость заражения). Для первого контрольного примера вводятся следующие начальные данные: , , . Результат моделирования изображен на рисунке 4.16.

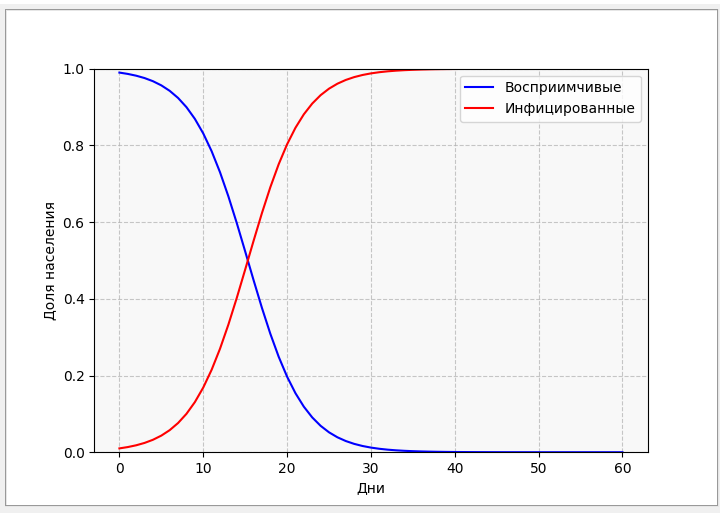


Рисунок 4.16 – Результат SI-модели для первого контрольного примера с помощью метода Рунге-Кутта 4-го порядка

Для второго контрольного примера вводятся следующие начальные данные: , , . Результат прогнозирования показан на рисунке 4.17.

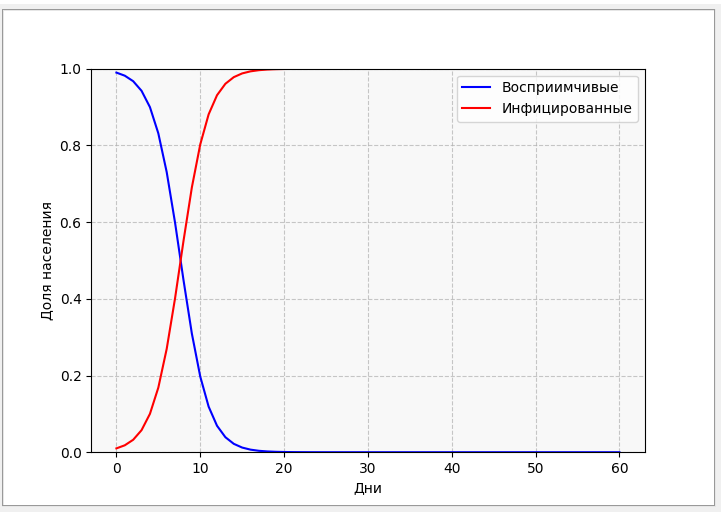


Рисунок 4.17 - Результат SI-модели для второго контрольного примера с помощью метода Рунге-Кутта 4-го порядка

На графике видно, что с увеличением скорости заражения рост инфицированных ускоряется. Количество инфицированных и восприимчивых к болезни наступает в первый десяток дней, когда в предыдущем примере это происходило только во второй десяток.

4.3.3 SIRS-модель

SIRS-модель состоит из трех начальных значений (восприимчивые, инфицированные и выздоровевшие) и трех параметров (скорость заражения, скорость выздоровления и скорость потери иммунитета). Для первого контрольного примера вводятся следующие начальные данные: , , , , , = 0.01. Результат прогнозирования показан на рисунке 4.18.

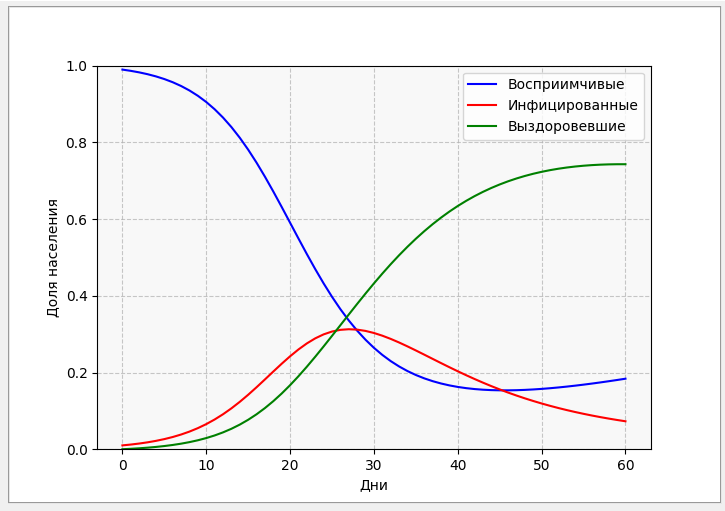


Рисунок 4.18 – Результат SIRS-модели для первого контрольного примера с помощью метода Рунге-Кутта 4-го порядка

Для второго контрольного примера вводятся следующие начальные данные: , , , , , = 0.5. Результат моделирования для второго примера изображен на рисунке 4.19.

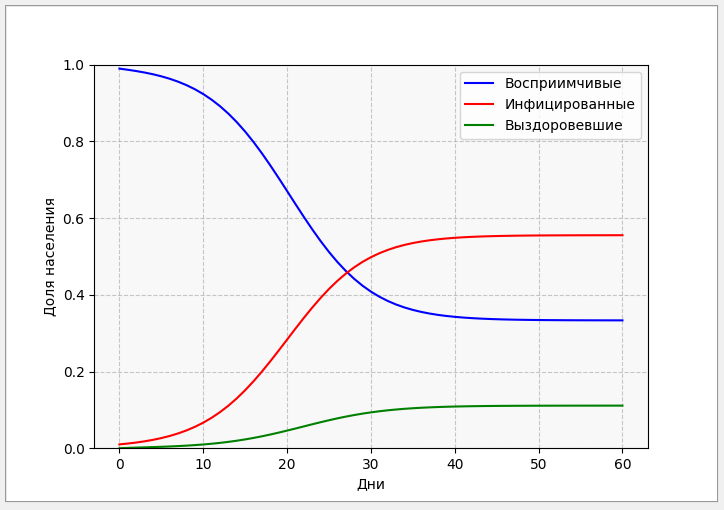


Рисунок 4.19 – Результат SIRS-модели для второго контрольного примера с помощью метода Рунге-Кутта 4-го порядка

На графике видно, что увеличение параметра скорости потери иммунитета изменяет динамику течения эпидемии. Выздоровевшая доля населения быстрее теряет иммунитет, а значит быстрее заболевает повторно. Если в первом контрольном примере инфицированных к 60 дню было около 0.1, то во втором контрольном примере почти 0.6 и их количество практически не изменяется.

4.3.4 SIQR-модель

SIQR-модель состоит из четырех начальных значений (восприимчивые, инфицированные, изолированные и выздоровевшие) и четырех параметров (скорость заражения, скорость выздоровления, скорость перехода на карантин и скорость выздоровления изолированных). Для первого контрольного примера вводятся следующие начальные данные: , ,, , , , = 0.05, = 0.05. Результат для первого контрольного примера представлен на рисунке 4.20.

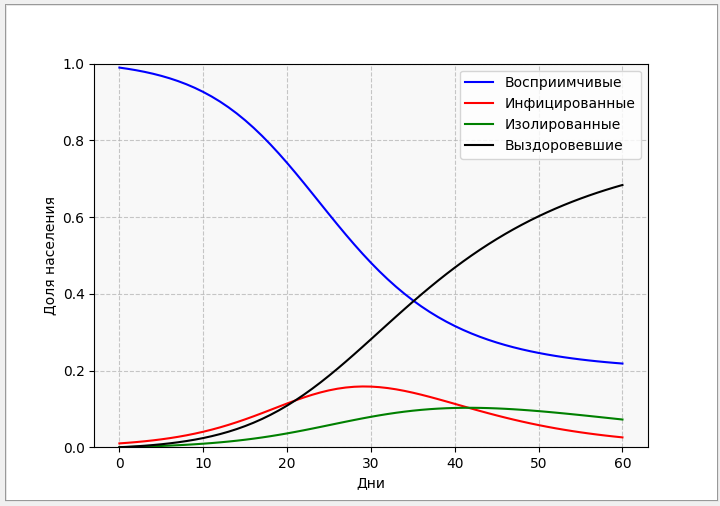


Рисунок 4.20 – Результат SIQR-модели для первого контрольного примера с помощью метода Рунге-Кутта 4-го порядка

Для второго контрольного примера вводятся следующие начальные данные: , ,, , , , = 0.1, = 0.05. Результат прогнозирования показан на рисунке 4.21.

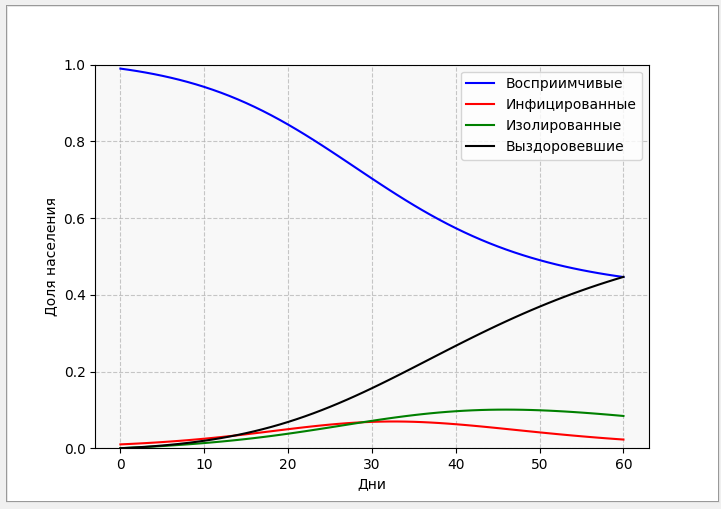


Рисунок 4.21 – Результат SIQR-модели для второго контрольного примера с помощью метода Рунге-Кутта 4-го порядка

На графике видно, что увеличение параметра скорости изоляции инфицированных сильно влияет на течение эпидемии. В первом случае количество инфицированных начинало снижаться с увеличением роста изолированных, но их пик наступил позже и находится в пределах 0.1 доли населения, когда пик инфицированных был примерно 0.7-0.8. Во втором случае наблюдается ситуация, когда изолированных намного больше, чем инфицированных. Количество выздоровевших и восприимчивых совпадает ровно на 60 день, а рост инфицированных снижается до 0.05-0.1.

4.3.5 SEIR-модель

SEIR-модель состоит из четырех начальных значений (восприимчивые, инфицированные, латентные и выздоровевшие) и трех параметров (скорость заражения, скорость выздоровления и скорость выхода из латентного периода). Для первого контрольного примера вводятся следующие начальные данные: , , , , , , . Результат моделирования возможного развития эпидемиологической ситуации представлен на рисунке 4.22.

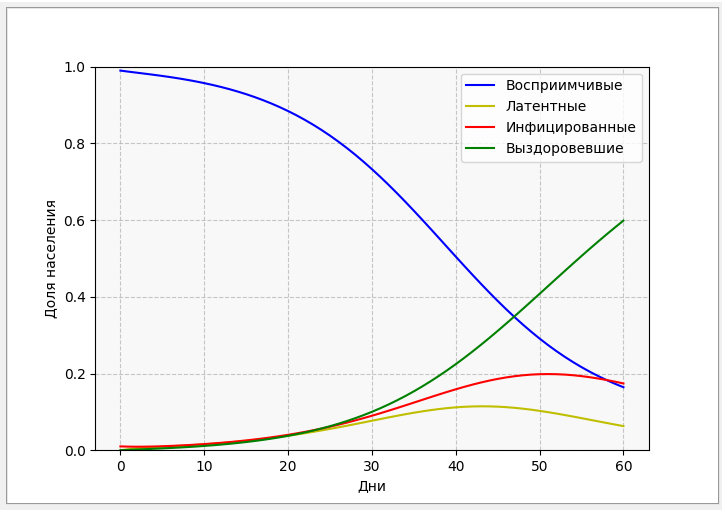


Рисунок 4.22 – Результат SEIR-модели для первого контрольного примера с помощью метода Рунге-Кутта 4-го порядка

Для второго контрольного примера вводятся следующие начальные данные: , , ,, , , . Результат моделирования изображен на рисунке 4.23.

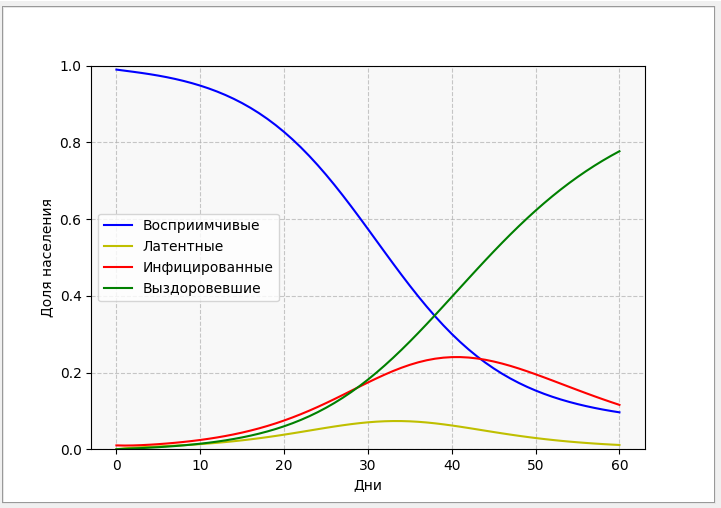


Рисунок 4.23 – Результат SEIR-модели для второго контрольного примера с помощью метода Рунге-Кутта 4-го порядка

На графике видно, что увеличение параметра выхода из латентного состояния способствует резкому росту количества инфицированных. Пик эпидемии наступает раньше, чем в первом случае. Количество инфицированных также становится большем. В первом случае в пике их было не больше 0.2, когда во втором случае их пик приходится на 0.25-0.3

4.3.6 MSEIR-модель

MSEIR-модель состоит из пяти начальных значений (младенцы с материнским иммунитетом, восприимчивые, инфицированные, латентные и выздоровевшие) и пяти параметров (скорость заражения, скорость выздоровления, скорость выхода из латентного периода, естественная смертность/рождаемость и скорость потери материнского иммунитета). Для первого контрольного примера вводятся следующие начальные данные: , , , , , , , , , . Результат моделирования показан на рисунке 4.24.

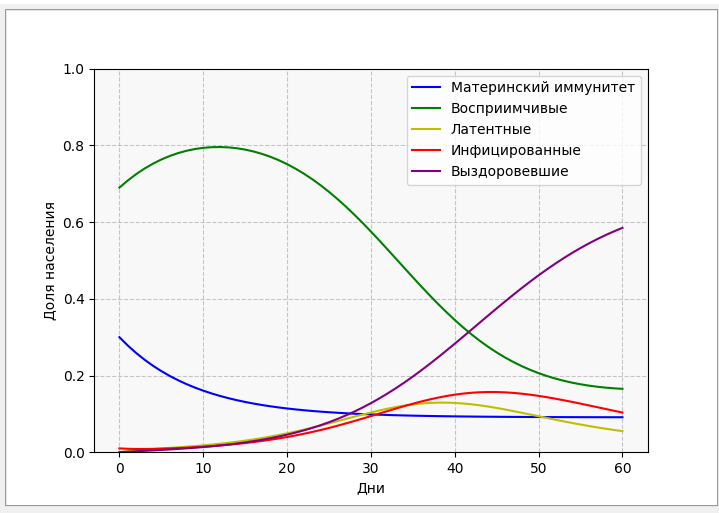


Рисунок 4.24 – Результат MSEIR-модели для первого контрольного примера с помощью метода Рунге-Кутта 4-го порядка

Для второго контрольного примера вводятся следующие начальные данные: , , , , , , , , , . Результат прогнозирования развития эпидемиологической ситуации представлен на рисунке 4.25.

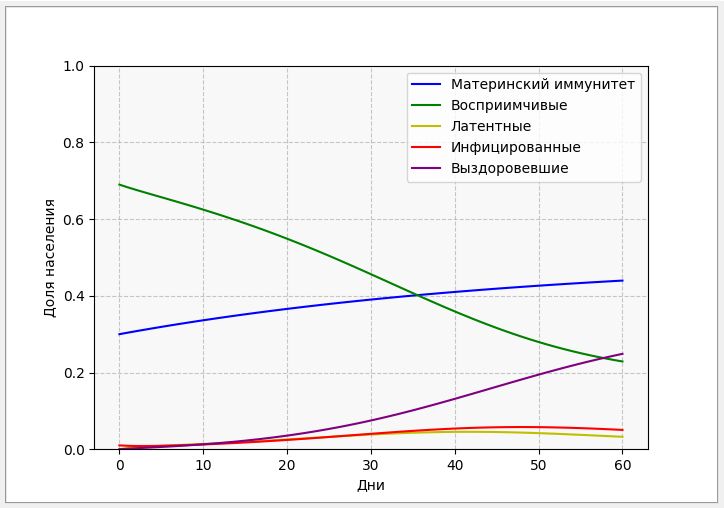


Рисунок 4.25 – Результат MSEIR-модели для второго контрольного примера с помощью метода Рунге-Кутта 4-го порядка

На графике видно, что уменьшение параметра потери материнского иммунитета снижает пик инфицированных, а также способствует сдерживанию эпидемии, так как к 60 дню количество заболевших снижается.

4.4 Анализ результатов моделирования

На основе представленных в данной работе математических моделей и их тестировании можно выделить следующие особенности:

* SIR-модель подходит для моделирования заболеваний, иммунитет для которых постоянен и сохраняется на всю жизнь. К таким заболеваниям относится ветряная оспа;
* SI-модель подходит для моделирования заболеваний, у которых отсутствует этап выздоровления. К таким заболеваниям относится гепатит B и C, а также ВИЧ/СПИД инфекции;
* SIRS-модель позволяет прогнозировать развитие эпидемиологической ситуации для более продолжительного срока, так как она учитывает потерю иммунитета у населения. Данная модель наиболее подходит для моделирования распространения ротавирусных и респираторных вирусов;
* SEIR-модель позволяет моделировать инфекции, у которых есть латентный (инкубационный) период. К таким инфекциям относится COVID-19;
* SIQR-модель подходит для моделирования развития эпидемиологической ситуации, при которой необходимо проанализировать воздействие карантинных мер. Данная модель наиболее полезна при моделировании развития эпидемии COVID-19 и других респираторных инфекций;
* MSEIR-модель подходит для моделирования распространения инфекций наиболее опасных для новорожденных. К таким заболеваниям относится корь.

Подводя итог анализа моделей, стоит отметить, что для каждой инфекции необходимо выбирать наиболее подходящий инструментарий, позволяющий представить полную картину возможного развития эпидемиологической ситуации. В таблице 4.1 приведены инфекции и модели, наиболее подходящие для их моделирования.

Таблица 4.1 **–** Инфекции подходящие для моделирования

|  |  |
| --- | --- |
| Название модели | Наиболее подходящие инфекции для моделирования |
| SIR | Грипп, ветряная оспа, корь, эпидемический паротит |
| SI | ВИЧ, гепатит B/C и другие хронические вирусные инфекции |
| SIRS | Риновирусы, ротавирусные инфекции, стрептококк группы A |
| SIQR | COVID-19, туберкулез, Эбола, другие инфекции с жесткими карантинными мерами |
| SEIR | COVID-19, туберкулез, корь, ветряная оспа |
| MSEIR | Краснуха, корь, ветряная оспа, инфекции с пассивным материнским иммунитетом |

4.5 Выводы по главе

В данной главе протестировано программное средство и реализованные математические модели на различных начальных данных. На основе анализа представленных в данной работе моделей выделены инфекции наиболее подходящие для моделирования развития эпидемиологической ситуации.

Заключение

В ходе работы создано программное средство для математического моделирования динамики эпидемиологической ситуации. Разработанная система предоставляет возможность проведения численных экспериментов с использованием различных параметров модели, анализа распространения инфекции и оценки эффективности мер по ее контролю. Она также позволяет расширить функциональные возможности, в том числе внедрение более сложных моделей, учет пространственных факторов и калибровки параметров по реальным эпидемиологическим данным.

Результаты тестирования подтвердили корректность работы программы и ее пригодность для решения актуальных задач в области эпидемиологии. Созданное программное средство может быть использовано как для научных исследований, так и в качестве вспомогательного средства для принятия решений в сфере управления при возникновении реальной эпидемической угрозы. Оно может также применяться в учебном процессе при изучении математических моделей. Интерфейс с возможностью настройки параметров делает его удобным для демонстрации принципов моделирования, проведения лабораторных работ и самостоятельных студенческих исследований.

Представленное разработанное программное средство и результаты исследования на конференции «Актуальные проблемы науки и техники – 2025» свидетельствует об актуальности проблемы моделирования эпидемиологической ситуации и практической значимости выполненной работы.

# **Перечень используемых информационных источников**

1. Кермак У., МакКендрик А. Математическая теория эпидемий. — 1927.
2. Бернулли Д. Essai d'une nouvelle analyse de la mortalité causée par la petite vérole. — 1760.
3. Diekmann O., Heesterbeek J.A.P. Mathematical Epidemiology of Infectious Diseases. — Wiley, 2000. ISBN 978-0-471-49841-3.
4. Hethcote H.W. The Mathematics of Infectious Diseases. — SIAM Review, 2000. DOI: 10.1137/S0036144500371907.
5. Brauer F., Castillo-Chavez C., Feng Z. Mathematical Models in Epidemiology. — Springer, 2019. ISBN 978-1-4939-9827-3.
6. Ross R. The Prevention of Malaria. — John Murray, 1911.
7. Anderson R.M., May R.M. Infectious Diseases of Humans: Dynamics and Control. — Oxford University Press, 1991. ISBN 978-0-19-854040-3.
8. Keeling M.J., Rohani P. Modeling Infectious Diseases in Humans and Animals. — Princeton University Press, 2008. ISBN 978-0-691-11617-4.
9. Edelstein-Keshet L. Mathematical Models in Biology. — SIAM, 2005. ISBN 978-0-89871-554-5.
10. Murray J.D. Mathematical Biology. — Springer, 2002. ISBN 978-0-387-95223-1.
11. Allen L.J.S. An Introduction to Mathematical Biology. — Pearson Education, 2007. ISBN 978-0-13-045591-1.
12. Python Software Foundation. Python 3.13 Documentation. — https://docs.python.org/3.13/
13. Лутц М. Изучаем Python. — СПб: Символ-Плюс, 2021. ISBN 978-5-6046440-6-4.
14. Гудман Б. Python и анализ данных. — М.: Диалектика, 2022. ISBN 978-5-8459-2531-2.
15. Бутлер М. Программирование на Python: от простого к сложному. — СПб: Питер, 2020. ISBN 978-5-4461-0925-0.
16. Сиденко С. Методы численного анализа. — М.: Физматлит, 2018. ISBN 978-5-9221-1813-9.
17. Кутта М. Введение в численные методы. — М.: Наука, 1987.
18. Guckenheimer J., Holmes P. Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields. — Springer, 2002. ISBN 978-0-387-90819-1.
19. Воронин А.А. Математические модели распространения эпидемий. — М.: МГУ, 2019. ISBN 978-5-19-011784-6.
20. Арсеньев В.В. Математические методы в биологии. — М.: URSS, 2015. ISBN 978-5-9710-2073-0.

**Приложение А Техническое задание**

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Доцент кафедры «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_ Т.А. Медведева  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. | УТВЕРЖДЕНО  Зав. кафедрой «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_ В.В. Долгов  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2025 г. |

**А.1. Общие сведения**

**А.1.1 Наименование программного средства**

Программное средство «EpidemicModels».

**А.1.2 Область применения**

Разработанное программное средство может быть использовано в медицинской сфере для моделирования эпидемиологических ситуаций, прогнозирования их развития и своевременного принятия мер в соответствии с прогнозами. Оно также может использовано для изучения предметов, связанных с математическим моделированием и вычислительной математики.

**А.2 Основание для разработки**

Разработка ведется на основании документа «Учебный план для студентов ВУЗа» направление 02.03.03 «Математическое обеспечение и администрирование информационных систем» кафедры «Программное обеспечение вычислительной техники и автоматизированных систем» факультета «Информатика и вычислительная техника» Донского государственного технического университета.

**А.3 Назначение разработки**

**А.3.1 Функциональное назначение**

Функциональное назначение программного средства заключается в прогнозировании развития эпидемиологической ситуации с помощью математических моделей.

#### **А.3.2 Эксплуатационное назначение**

Эксплуатационное назначение состоит в использовании программного средства на компьютере.

#### **A.4 Требования к программе**

#### **А.4.1 Требования к функциональным характеристикам**

Программное средство должно осуществлять следующий функционал:

* возможность выбора модели (от 1 до 4);
* ввод соответствующих модели параметров;
* возможность ввода реальных данных из файла формата .csv;
* моделирование различных стадий эпидемий;
* ввод даты начала и окончания моделирования;
* возможность выбора численного метода для решения СДУ;
* расчет выбранных моделей с помощью численного метода;
* вывод полученных результатов в графическом виде на экран;
* возможность экспорта результатов в табличном и графическом виде в файл формата .xlsx.

#### **А.4.2 Требования к надежности**

Надежное функционирование программы должно быть обеспечено выполнением Заказчиком совокупности организационно–технических мероприятий, перечень которых приведен ниже:

* организация бесперебойного питания технических средств;
* организация правильного использования модулей;
* защиты программного средства от несанкционированного проникновения;
* копирование (архивирование) данных, необходимых для восстановления работы программного средства;
* регулярным выполнением требований ГОСТ 51188-98. Защита информации. Испытания программных средств на наличие вирусов.

#### **А.4.2.1 Входные данные**

Входными данными являются числовые значения для начальных условий и параметров моделей с помощью клавиатуры или с использованием данных из файла формата .csv, выбор математических моделей, ввод начальной и конечной даты прогнозирования, выбор численного метода.

#### **А.4.2.2 Выходные данные**

Выходные данные являются результаты расчетов математических моделей в графическом виде в интерфейсе и в виде полученных в табличном и графическом виде файла формата .xlsx.

#### **А.4.3 Условия эксплуатации**

Для функционирования программного продукта необходимо соблюдение всех требований и правил эксплуатации компьютерной техники. Для работы с программным средством необходимы навыки работы с персональным компьютером на уровне пользователя. Дополнительных требований и ограничений не вводится.

#### **А.4.4 Требование к составу и параметрам технических средств**

Состав технических средств пользователя:

* процессор не менее 4 ядер и 2.8 ГГц;
* оперативная память объемом не менее 8 Гб.

#### **А.4.5 Требования к исходным кодам и языкам программирования**

Программное средство должно быть реализовано на языке программирования Python, с использованием библиотек matplotlib, pandas и numpy.

#### **А.4.7 Требования к транспортировке и хранению**

Условия транспортирования, места хранения, условия складирования и сроки хранения в различных условиях должны соответствовать требованиям, предъявляемым к носителям информации, на которых будет содержаться данное программное изделие.

Допустимы все способы транспортирования и хранения, не нарушающие целостность используемого носителя данных. Программное средство может храниться на любом носителе информации, имеющее возможность подключения к персональному компьютеру.

#### **А.5 Требования к программной документации**

Программная документация должна состоять из следующих листов:

* титульный лист;
* пояснительная записка к дипломной работе;
* техническое задание по ГОСТ 15.016-2016 ЕСПД;
* исходный код программного средства по ГОСТ Р 51904-2002 ЕСПД.

#### **А.6 Стадии и этапы разработки**

Стадии и этапы разработки программного средства:

* постановка задачи (с 21.04.25 по 27.04.25);
* изучение предметной области (с 27.04.25 по 29.04.25);
* разработка алгоритма решения задачи (с 29.04.25 по 07.05.25);
* разработка программы (с 07.05.25 по 14.05.25);
* тестирование программы (с 14.05.25 по 16.05.25).

#### **А.7 Порядок контроля и приемки**

Порядок и контроль приемки определяются заведующим кафедрой «ПОВТиАС» и основаны на демонстрации знаний технологии и умении создавать программные средства для различных предметных областей.

Главным требованием к приемке является наличие правильно работающего программного средства с тестовым примером и отчета, представленного в печатном виде.

|  |  |
| --- | --- |
| Разработчик технического задания:  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. | /Волкова Эмилия Юрьевна/  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

# **Приложение Б Исходный код программного средства**

***Листинг 1 – Исходный код программного средства***

import tkinter as tk

from tkinter import ttk, messagebox, filedialog

from tkcalendar import DateEntry

from datetime import datetime, timedelta

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import numpy as np

import pandas as pd

import os

import zipfile

from io import BytesIO

import re

class NumericalMethods:

def \_\_init\_\_(self):

self.result = None

def euler\_method(self, model\_func, y0, t, args):

y = np.zeros((len(t), len(y0)))

y[0] = y0

for i in range(1, len(t)):

dt = t[i] - t[i-1]

dy = model\_func(y[i-1], t[i-1], \*args)

y[i] = y[i-1] + dy \* dt

self.result = y.T

def runge\_kutta\_4(self, model\_func, y0, t, args):

y = np.zeros((len(t), len(y0)))

y[0] = y0

for i in range(1, len(t)):

dt = t[i] - t[i-1]

h = dt

k1 = model\_func(y[i-1], t[i-1], \*args)

k2 = model\_func(y[i-1] + 0.5\*h\*k1, t[i-1] + 0.5\*h, \*args)

k3 = model\_func(y[i-1] + 0.5\*h\*k2, t[i-1] + 0.5\*h, \*args)

k4 = model\_func(y[i-1] + h\*k3, t[i-1] + h, \*args)

y[i] = y[i-1] + (h/6.0)\*(k1 + 2\*k2 + 2\*k3 + k4)

self.result = y.T

class EpidemicModels(NumericalMethods):

def \_\_init\_\_(self):

self.current\_models = []

self.numeric\_methods = []

self.axes = []

self.canvases = []

self.figs = []

def sir\_model(self, y, t, beta, gamma):

S, I, R = y

dSdt = -beta \* S \* I

dIdt = beta \* S \* I - gamma \* I

dRdt = gamma \* I

return np.array([dSdt, dIdt, dRdt])

def si\_model(self, y, t, beta):

S, I = y

dSdt = -beta \* S \* I

dIdt = beta \* S \* I

return np.array([dSdt, dIdt])

def sirs\_model(self, y, t, beta, gamma, delta):

S, I, R = y

dSdt = -beta \* S \* I + delta \* R

dIdt = beta \* S \* I - gamma \* I

dRdt = gamma \* I - delta \* R

return np.array([dSdt, dIdt, dRdt])

def siqr\_model(self, y, t, beta, gamma, delta, mu):

S, I, Q, R = y

dSdt = -beta \* S \* I

dIdt = beta \* S \* I - gamma \* I - delta \* I

dQdt = delta \* I - mu \* Q

dRdt = gamma \* I + mu \* Q

return np.array([dSdt, dIdt, dQdt, dRdt])

def seir\_model(self, y, t, beta, sigma, gamma):

S, E, I, R = y

dSdt = -beta \* S \* I

dEdt = beta \* S \* I - sigma \* E

dIdt = sigma \* E - gamma \* I

dRdt = gamma \* I

return np.array([dSdt, dEdt, dIdt, dRdt])

def mseir\_model(self, y, t, mu, delta, beta, sigma, gamma):

M, S, E, I, R = y

N = M + S + E + I + R

dMdt = mu \* N - delta \* M - mu \* M

dSdt = delta \* M - beta \* S \* I / N - mu \* S

dEdt = beta \* S \* I / N - sigma \* E - mu \* E

dIdt = sigma \* E - gamma \* I - mu \* I

dRdt = gamma \* I - mu \* R

return np.array([dMdt, dSdt, dEdt, dIdt, dRdt])

def run\_si\_model(self, t, params, initials, plot\_index, return\_solution=False, method="runge\_kutta"):

y0 = [initials["S0"], initials["I0"]]

if method == "runge\_kutta":

self.runge\_kutta\_4(self.si\_model, y0, t, (params["beta"],))

else:

self.euler\_method(self.si\_model, y0, t, (params["beta"],))

S, I = self.result

ax = self.axes[plot\_index]

ax.clear()

ax.plot(t, S, 'b', label='Восприимчивые')

ax.plot(t, I, 'r', label='Инфицированные')

ax.set\_ylim(0, 1)

ax.set\_xlabel('Дни')

ax.set\_ylabel('Доля населения')

ax.grid(True)

ax.legend()

self.canvases[plot\_index].draw()

if return\_solution:

return {"S": S, "I": I}

def run\_sir\_model(self, t, params, initials, plot\_index, return\_solution=False, method="runge\_kutta"):

y0 = [initials["S0"], initials["I0"], initials["R0"]]

if method == "runge\_kutta":

self.runge\_kutta\_4(self.sir\_model, y0, t, (params["beta"], params["gamma"]))

else:

self.euler\_method(self.sir\_model, y0, t, (params["beta"], params["gamma"]))

S, I, R = self.result

ax = self.axes[plot\_index]

ax.clear()

ax.plot(t, S, 'b', label='Восприимчивые')

ax.plot(t, I, 'r', label='Инфицированные')

ax.plot(t, R, 'g', label='Выздоровевшие')

ax.set\_ylim(0, 1)

ax.set\_xlabel('Дни')

ax.set\_ylabel('Доля населения')

ax.grid(True)

ax.legend()

self.canvases[plot\_index].draw()

if return\_solution:

return {"S": S, "I": I, "R": R}

def run\_sirs\_model(self, t, params, initials, plot\_index, return\_solution=False, method="runge\_kutta"):

y0 = [initials["S0"], initials["I0"], initials["R0"]]

if method == "runge\_kutta":

self.runge\_kutta\_4(self.sirs\_model, y0, t, (params["beta"], params["gamma"], params["delta"]))

else:

self.euler\_method(self.sirs\_model, y0, t, (params["beta"], params["gamma"], params["delta"]))

S, I, R = self.result

ax = self.axes[plot\_index]

ax.clear()

ax.plot(t, S, 'b', label='Восприимчивые')

ax.plot(t, I, 'r', label='Инфицированные')

ax.plot(t, R, 'g', label='Выздоровевшие')

ax.set\_ylim(0, 1)

ax.set\_xlabel('Дни')

ax.set\_ylabel('Доля населения')

ax.grid(True)

ax.legend()

self.canvases[plot\_index].draw()

if return\_solution:

return {"S": S, "I": I, "R": R}

def run\_seir\_model(self, t, params, initials, plot\_index, return\_solution=False, method="runge\_kutta"):

y0 = [initials["S0"], initials["E0"], initials["I0"], initials["R0"]]

if method == "runge\_kutta":

self.runge\_kutta\_4(self.seir\_model, y0, t, (params["beta"], params["sigma"], params["gamma"]))

else:

self.euler\_method(self.seir\_model, y0, t, (params["beta"], params["sigma"], params["gamma"]))

S, E, I, R = self.result

ax = self.axes[plot\_index]

ax.clear()

ax.plot(t, S, 'b', label='Восприимчивые')

ax.plot(t, E, 'y', label='Латентные')

ax.plot(t, I, 'r', label='Инфицированные')

ax.plot(t, R, 'g', label='Выздоровевшие')

ax.set\_ylim(0, 1)

ax.set\_xlabel('Дни')

ax.set\_ylabel('Доля населения')

ax.grid(True)

ax.legend()

self.canvases[plot\_index].draw()

if return\_solution:

return {"S": S, "E": E, "I": I, "R": R}

def run\_mseir\_model(self, t, params, initials, plot\_index, return\_solution=False, method="runge\_kutta"):

y0 = [initials["M0"], initials["S0"], initials["E0"], initials["I0"], initials["R0"]]

if method == "runge\_kutta":

self.runge\_kutta\_4(self.mseir\_model, y0, t,

(params["mu"], params["delta"], params["beta"], params["sigma"], params["gamma"]))

else:

self.euler\_method(self.mseir\_model, y0, t,

(params["mu"], params["delta"], params["beta"], params["sigma"], params["gamma"]))

M, S, E, I, R = self.result

ax = self.axes[plot\_index]

ax.clear()

ax.plot(t, M, 'b', label='Материнский иммунитет')

ax.plot(t, S, 'g', label='Восприимчивые')

ax.plot(t, E, 'y', label='Латентные')

ax.plot(t, I, 'r', label='Инфицированные')

ax.plot(t, R, 'purple', label='Выздоровевшие')

ax.set\_ylim(0, 1)

ax.set\_xlabel('Дни')

ax.set\_ylabel('Доля населения')

ax.grid(True)

ax.legend()

self.canvases[plot\_index].draw()

if return\_solution:

return {"M": M, "S": S, "E": E, "I": I, "R": R}

def run\_siqr\_model(self, t, params, initials, plot\_index, return\_solution=False, method="runge\_kutta"):

y0 = [initials["S0"], initials["I0"], initials["Q0"], initials["R0"]]

if method == "runge\_kutta":

self.runge\_kutta\_4(self.siqr\_model, y0, t,

(params["beta"], params["gamma"], params["delta"], params["mu"]))

else:

self.euler\_method(self.siqr\_model, y0, t,

(params["beta"], params["gamma"], params["delta"], params["mu"]))

S, I, Q, R = self.result

ax = self.axes[plot\_index]

ax.clear()

ax.plot(t, S, 'b', label='Восприимчивые')

ax.plot(t, I, 'r', label='Инфицированные')

ax.plot(t, Q, 'g', label='Изолированные')

ax.plot(t, R, 'k', label='Выздоровевшие')

ax.set\_ylim(0, 1)

ax.set\_xlabel('Дни')

ax.set\_ylabel('Доля населения')

ax.grid(True)

ax.legend()

self.canvases[plot\_index].draw()

if return\_solution:

return {"S": S, "I": I, "Q": Q, "R": R}

class EpidemicApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Эпидемиологическое моделирование")

self.root.geometry("1200x800")

self.root.minsize(1000, 700)

self.style = ttk.Style()

self.style.configure('TFrame', background='#f0f0f0')

self.style.configure('TLabel', background='#f0f0f0', font=('Arial', 10))

self.style.configure('TButton', font=('Arial', 10))

self.style.configure('TCheckbutton', background='#f0f0f0', font=('Arial', 10))

self.style.configure('TRadiobutton', background='#f0f0f0', font=('Arial', 10))

self.model = EpidemicModels()

self.result\_data = {}

self.create\_widgets()

self.set\_default\_values()

def create\_widgets(self):

main\_paned = ttk.PanedWindow(self.root, orient=tk.HORIZONTAL)

main\_paned.pack(fill=tk.BOTH, expand=True)

control\_frame = ttk.Frame(main\_paned, width=350, relief=tk.RIDGE, padding=10)

main\_paned.add(control\_frame, weight=0)

graph\_frame = ttk.Frame(main\_paned)

main\_paned.add(graph\_frame, weight=1)

control\_notebook = ttk.Notebook(control\_frame)

control\_notebook.pack(fill=tk.BOTH, expand=True)

models\_tab = ttk.Frame(control\_notebook)

control\_notebook.add(models\_tab, text="Модели")

self.create\_models\_tab(models\_tab)

params\_tab = ttk.Frame(control\_notebook)

control\_notebook.add(params\_tab, text="Параметры")

self.create\_params\_tab(params\_tab)

data\_tab = ttk.Frame(control\_notebook)

control\_notebook.add(data\_tab, text="Данные")

self.create\_data\_tab(data\_tab)

self.create\_graphs(graph\_frame)

def create\_models\_tab(self, parent):

models\_group = ttk.LabelFrame(parent, text="Выберите модели (макс. 4)", padding=10)

models\_group.pack(fill=tk.BOTH, pady=5)

self.model\_vars = {}

models = [

('SI', 'Модель SI (восприимчивые-инфицированные)'),

('SIR', 'Модель SIR (восприимчивые-инфицированные-выздоровевшие)'),

('SIRS', 'Модель SIRS (с временным иммунитетом)'),

('SEIR', 'Модель SEIR (с латентным периодом)'),

('SIQR', 'Модель SIQR (с изоляцией)'),

('MSEIR', 'Модель MSEIR (с материнским иммунитетом)')

]

for model\_code, model\_desc in models:

var = tk.BooleanVar()

cb = ttk.Checkbutton(models\_group, text=model\_desc, variable=var,

command=self.update\_model\_selection)

cb.pack(anchor='w', padx=5, pady=2)

self.model\_vars[model\_code] = var

method\_group = ttk.LabelFrame(parent, text="Метод решения", padding=10)

method\_group.pack(fill=tk.BOTH, pady=5)

self.method\_var = tk.StringVar(value="runge\_kutta")

ttk.Radiobutton(method\_group, text="Рунге-Кутта 4-го порядка",

variable=self.method\_var, value="runge\_kutta").pack(anchor='w', padx=5, pady=2)

ttk.Radiobutton(method\_group, text="Метод Эйлера",

variable=self.method\_var, value="euler").pack(anchor='w', padx=5, pady=2)

ttk.Button(parent, text="Запустить моделирование",

command=self.run\_simulation).pack(fill=tk.X, pady=10)

def create\_params\_tab(self, parent):

canvas = tk.Canvas(parent)

scrollbar = ttk.Scrollbar(parent, orient="vertical", command=canvas.yview)

scrollable\_frame = ttk.Frame(canvas)

scrollable\_frame.bind(

"<Configure>",

lambda e: canvas.configure(

scrollregion=canvas.bbox("all")

)

)

canvas.create\_window((0, 0), window=scrollable\_frame, anchor="nw")

canvas.configure(yscrollcommand=scrollbar.set)

canvas.pack(side="left", fill="both", expand=True)

scrollbar.pack(side="right", fill="y")

params\_group = ttk.LabelFrame(scrollable\_frame, text="Параметры модели", padding=10)

params\_group.pack(fill=tk.X, pady=5)

self.param\_entries = {}

params = [

("beta", "β (скорость заражения)", 0.0, 1.0),

("gamma", "γ (скорость выздоровления)", 0.0, 1.0),

("delta", "δ (потеря иммунитета)", 0.0, 1.0),

("sigma", "σ (переход в инфекционные)", 0.0, 1.0),

("mu", "μ (выход из изоляции)", 0.0, 1.0)

]

for param\_code, param\_desc, min\_val, max\_val in params:

row = ttk.Frame(params\_group)

row.pack(fill=tk.X, pady=2)

ttk.Label(row, text=param\_desc, width=25).pack(side=tk.LEFT)

entry = ttk.Entry(row, width=8)

entry.pack(side=tk.RIGHT)

self.param\_entries[param\_code] = entry

initials\_group = ttk.LabelFrame(scrollable\_frame, text="Начальные значения (доля)", padding=10)

initials\_group.pack(fill=tk.X, pady=5)

self.init\_entries = {}

initials = [

("S0", "S₀ (восприимчивые)", 0.0, 1.0),

("I0", "I₀ (инфицированные)", 0.0, 1.0),

("R0", "R₀ (выздоровевшие)", 0.0, 1.0),

("E0", "E₀ (латентные)", 0.0, 1.0),

("Q0", "Q₀ (изолированные)", 0.0, 1.0),

("M0", "M₀ (материнский иммунитет)", 0.0, 1.0)

]

for init\_code, init\_desc, min\_val, max\_val in initials:

row = ttk.Frame(initials\_group)

row.pack(fill=tk.X, pady=2)

ttk.Label(row, text=init\_desc, width=25).pack(side=tk.LEFT)

entry = ttk.Entry(row, width=8)

entry.pack(side=tk.RIGHT)

self.init\_entries[init\_code] = entry

time\_group = ttk.LabelFrame(scrollable\_frame, text="Временной диапазон", padding=10)

time\_group.pack(fill=tk.X, pady=5)

ttk.Label(time\_group, text="Начальная дата:").pack(anchor='w', pady=(0, 5))

self.start\_entry = DateEntry(time\_group, date\_pattern='dd.mm.yyyy')

self.start\_entry.pack(fill=tk.X, pady=(0, 10))

ttk.Label(time\_group, text="Конечная дата:").pack(anchor='w', pady=(0, 5))

self.end\_entry = DateEntry(time\_group, date\_pattern='dd.mm.yyyy')

self.end\_entry.pack(fill=tk.X)

def create\_data\_tab(self, parent):

load\_group = ttk.LabelFrame(parent, text="Загрузка данных", padding=10)

load\_group.pack(fill=tk.BOTH, pady=5, expand=True)

ttk.Button(load\_group, text="Загрузить данные из CSV",

command=self.load\_csv\_data).pack(fill=tk.X, pady=5)

export\_group = ttk.LabelFrame(parent, text="Экспорт результатов", padding=10)

export\_group.pack(fill=tk.BOTH, pady=5)

ttk.Button(export\_group, text="Экспорт в Excel и ZIP",

command=self.export\_results).pack(fill=tk.X, pady=5)

def create\_graphs(self, parent):

self.model.axes = []

self.model.canvases = []

self.model.figs = []

for i in range(4):

fig = plt.Figure(figsize=(6, 4), dpi=100)

ax = fig.add\_subplot(111)

ax.grid(True, linestyle='--', alpha=0.7)

ax.set\_facecolor('#f8f8f8')

canvas = FigureCanvasTkAgg(fig, master=parent)

canvas\_widget = canvas.get\_tk\_widget()

canvas\_widget.grid(row=i//2, column=i%2, padx=5, pady=5, sticky='nsew')

canvas\_widget.config(borderwidth=2, relief=tk.GROOVE)

parent.grid\_rowconfigure(i//2, weight=1)

parent.grid\_columnconfigure(i%2, weight=1)

self.model.figs.append(fig)

self.model.axes.append(ax)

self.model.canvases.append(canvas)

def create\_validate\_func(self, min\_val, max\_val):

def validate(value):

if value == "":

return True

try:

num = float(value)

return min\_val <= num <= max\_val

except ValueError:

return False

return validate

def set\_default\_values(self):

self.param\_entries["beta"].insert(0, "0.3")

self.param\_entries["gamma"].insert(0, "0.1")

self.param\_entries["delta"].insert(0, "0.01")

self.param\_entries["sigma"].insert(0, "0.2")

self.param\_entries["mu"].insert(0, "0.05")

self.init\_entries["S0"].insert(0, "0.99")

self.init\_entries["I0"].insert(0, "0.01")

self.init\_entries["R0"].insert(0, "0.0")

self.init\_entries["E0"].insert(0, "0.0")

self.init\_entries["Q0"].insert(0, "0.0")

self.init\_entries["M0"].insert(0, "0.0")

today = datetime.now()

self.start\_entry.set\_date(today)

self.end\_entry.set\_date(today + timedelta(days=100))

def update\_model\_selection(self):

selected = sum(var.get() for var in self.model\_vars.values())

if selected > 4:

for model\_code, var in reversed(self.model\_vars.items()):

if var.get():

var.set(False)

messagebox.showwarning("Предупреждение", "Можно выбрать не более 4 моделей одновременно")

break

def run\_simulation(self):

selected\_models = [name for name, var in self.model\_vars.items() if var.get()]

if not selected\_models:

messagebox.showwarning("Ошибка", "Выберите хотя бы одну модель")

return

try:

params = {k: float(e.get()) if e.get() else 0.0 for k, e in self.param\_entries.items()}

initials = {k: float(e.get()) if e.get() else 0.0 for k, e in self.init\_entries.items()}

except ValueError:

messagebox.showerror("Ошибка", "Некорректные значения параметров")

return

if self.end\_entry.get\_date() <= self.start\_entry.get\_date():

messagebox.showerror("Ошибка", "Конечная дата должна быть позже начальной")

return

self.result\_data.clear()

days = (self.end\_entry.get\_date() - self.start\_entry.get\_date()).days

t = np.linspace(0, days, days + 1)

method = self.method\_var.get()

for ax in self.model.axes:

ax.clear()

for i, model\_code in enumerate(selected\_models):

if i >= 4:

break

if model\_code == "SI":

sol = self.model.run\_si\_model(t, params, initials, i, return\_solution=True, method=method)

elif model\_code == "SIR":

sol = self.model.run\_sir\_model(t, params, initials, i, return\_solution=True, method=method)

elif model\_code == "SIRS":

sol = self.model.run\_sirs\_model(t, params, initials, i, return\_solution=True, method=method)

elif model\_code == "SEIR":

sol = self.model.run\_seir\_model(t, params, initials, i, return\_solution=True, method=method)

elif model\_code == "SIQR":

sol = self.model.run\_siqr\_model(t, params, initials, i, return\_solution=True, method=method)

elif model\_code == "MSEIR":

sol = self.model.run\_mseir\_model(t, params, initials, i, return\_solution=True, method=method)

if sol:

self.result\_data[model\_code] = pd.DataFrame(sol, index=t)

for canvas in self.model.canvases:

canvas.draw()

def load\_csv\_data(self):

path = filedialog.askopenfilename(filetypes=[("CSV files", "\*.csv")])

if not path:

return

try:

df = pd.read\_csv(path)

df["Date"] = pd.to\_datetime(df["Date"])

df.sort\_values("Date", inplace=True)

self.csv\_data = df

countries = sorted(df["Country/Region"].dropna().unique())

self.country\_selection = tk.Toplevel(self.root)

self.country\_selection.title("Выбор страны и дат")

self.country\_selection.geometry("400x300")

ttk.Label(self.country\_selection, text="Выберите страну:").pack(pady=(10, 5))

self.country\_cb = ttk.Combobox(self.country\_selection, values=countries, state="readonly")

self.country\_cb.pack(pady=5)

ttk.Label(self.country\_selection, text="Выберите диапазон дат для начальных данных:").pack(pady=(10, 5))

date\_frame = ttk.Frame(self.country\_selection)

date\_frame.pack(pady=5)

ttk.Label(date\_frame, text="С:").pack(side=tk.LEFT)

self.data\_start\_entry = DateEntry(date\_frame, date\_pattern='dd.mm.yyyy')

self.data\_start\_entry.pack(side=tk.LEFT, padx=5)

ttk.Label(date\_frame, text="По:").pack(side=tk.LEFT)

self.data\_end\_entry = DateEntry(date\_frame, date\_pattern='dd.mm.yyyy')

self.data\_end\_entry.pack(side=tk.LEFT, padx=5)

min\_date = df["Date"].min().to\_pydatetime()

max\_date = df["Date"].max().to\_pydatetime()

self.data\_start\_entry.set\_date(min\_date)

self.data\_end\_entry.set\_date(max\_date)

ttk.Button(self.country\_selection, text="Загрузить данные",

command=self.process\_csv\_data).pack(pady=10)

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось загрузить файл: {str(e)}")

def process\_csv\_data(self):

country = self.country\_cb.get()

if not country:

messagebox.showwarning("Ошибка", "Выберите страну")

return

try:

start\_date = self.data\_start\_entry.get\_date()

end\_date = self.data\_end\_entry.get\_date()

if start\_date > end\_date:

messagebox.showerror("Ошибка", "Начальная дата не может быть позже конечной")

return

df\_country = self.csv\_data[

(self.csv\_data["Country/Region"] == country) &

(self.csv\_data["Date"] >= pd.to\_datetime(start\_date)) &

(self.csv\_data["Date"] <= pd.to\_datetime(end\_date))

].copy()

if df\_country.empty:

messagebox.showerror("Ошибка", "Нет данных для выбранного диапазона дат")

return

latest = df\_country.iloc[-1]

total = latest["Confirmed"] + latest["Recovered"] + latest["Deaths"]

if total == 0:

total = 1

S0 = 1 - (latest["Confirmed"] + latest["Recovered"] + latest["Deaths"]) / total

I0 = latest["Confirmed"] / total

R0 = latest["Recovered"] / total

for entry in self.init\_entries.values():

entry.delete(0, tk.END)

self.init\_entries["S0"].insert(0, f"{S0:.4f}")

self.init\_entries["I0"].insert(0, f"{I0:.4f}")

self.init\_entries["R0"].insert(0, f"{R0:.4f}")

self.start\_entry.set\_date(end\_date)

self.end\_entry.set\_date(end\_date + timedelta(days=100))

self.country\_selection.destroy()

messagebox.showinfo("Успех", f"Данные для {country} за период {start\_date.strftime('%d.%m.%Y')}-{end\_date.strftime('%d.%m.%Y')} успешно загружены")

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось обработать данные: {str(e)}")

def export\_results(self):

if not self.result\_data:

messagebox.showwarning("Нет данных", "Сначала выполните моделирование")

return

save\_path = filedialog.asksaveasfilename(

defaultextension=".zip",

filetypes=[("ZIP files", "\*.zip")],

initialfile="epidemic\_results.zip"

)

if not save\_path:

return

try:

mem\_zip = BytesIO()

with zipfile.ZipFile(mem\_zip, mode='w', compression=zipfile.ZIP\_DEFLATED) as zf:

for model\_name, df in self.result\_data.items():

excel\_buffer = BytesIO()

with pd.ExcelWriter(excel\_buffer, engine='xlsxwriter') as writer:

initials\_df = pd.DataFrame({

'Параметр': ['S₀ (восприимчивые)', 'I₀ (инфицированные)', 'R₀ (выздоровевшие)',

'E₀ (латентные)', 'Q₀ (изолированные)', 'M₀ (материнский иммунитет)'],

'Значение': [self.init\_entries["S0"].get(), self.init\_entries["I0"].get(),

self.init\_entries["R0"].get(), self.init\_entries["E0"].get(),

self.init\_entries["Q0"].get(), self.init\_entries["M0"].get()]

})

initials\_df.to\_excel(writer, sheet\_name='Начальные данные', index=False)

params\_df = pd.DataFrame({

'Параметр': ['β (скорость заражения)', 'γ (скорость выздоровления)',

'δ (потеря иммунитета)', 'σ (переход в инфекционные)',

'μ (выход из изоляции)'],

'Значение': [self.param\_entries["beta"].get(), self.param\_entries["gamma"].get(),

self.param\_entries["delta"].get(), self.param\_entries["sigma"].get(),

self.param\_entries["mu"].get()]

})

params\_df.to\_excel(writer, sheet\_name='Параметры', index=False)

method\_info = pd.DataFrame({

'Информация': ['Метод решения:', 'Выбранная модель:'],

'Значение': [self.method\_var.get().replace("runge\_kutta", "Рунге-Кутта 4-го порядка").replace("euler", "Метод Эйлера"),

model\_name]

})

method\_info.to\_excel(writer, sheet\_name='Решение', startrow=0, index=False, header=False)

df.to\_excel(writer, sheet\_name='Решение', startrow=3, index=True)

worksheet = writer.sheets['Решение']

worksheet.write(3, 0, 'Дни')

graph\_sheet = writer.book.add\_worksheet('График')

chart = writer.book.add\_chart({'type': 'line'})

max\_row = len(df) + 4

categories = f"='Решение'!$A$5:$A${max\_row}"

for i, col in enumerate(df.columns, 1):

chart.add\_series({

'name': f"='Решение'!${chr(66+i)}$4",

'categories': categories,

'values': f"='Решение'!${chr(66+i)}$5:${chr(66+i)}${max\_row}",

})

chart.set\_x\_axis({'name': 'Дни'})

chart.set\_y\_axis({'name': 'Доля населения'})

chart.set\_title({'name': f'Модель {model\_name} ({method\_info.iloc[0,1]})'})

graph\_sheet.insert\_chart('B2', chart, {'x\_scale': 2, 'y\_scale': 2})

zf.writestr(f"{model\_name}.xlsx", excel\_buffer.getvalue())

with open(save\_path, "wb") as f:

f.write(mem\_zip.getvalue())

messagebox.showinfo("Экспорт завершён", f"Файлы успешно сохранены в архив:\n{save\_path}")

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось экспортировать данные: {str(e)}")

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = EpidemicApp(root)

root.mainloop()