Санкт–Петербургский государственный университет

Косьянов Никита Олегович

Выпускная квалификационная работа

Применение управления с прогнозирующими моделями для задачи SIIR

Уровень образования: магистратура
Направление 01.04.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5504.2021 «Исследование операций и системный анализ»

Научный руководитель:
доцент, кафедра математической теории игр
и статистических решений, к. ф.-м. н.
Губар Елена Алексеевна

Рецензент:
ведущий инженер IT-отдела,
ООО «МНИЦ СиПБ»,
Тихонова Наталья Владимировна

Санкт-Петербург
2023 г.

Содержание

Введение 3

Обзор литературы 4

Постановка задачи 6

Глава 1. Эпидемические модели 7

1.1. Модель SIIR 7

1.2. Модель SIIRS 9

1.3. Модель SWIIRS 11

1.4. Линеаризация задачи SIIR 12

1.5. Управляемый параметр скорости заражения 15

Глава 2. Управление с прогнозированием модели 19

2.1. Построение УМП для линейных систем 19

Глава 3. Реализация УПМ в Matlab 22

3.1. Моделирование эпидемических задач 22

3.2. Подключение MPC-контроллера 23

Глава 4. Вычислительные эксперименты 25

4.1. Эксперимент 1 25

4.2. Эксперимент 2 27

4.3. Эксперимент 3 30

Заключение 33

Список литературы 34

Приложение 1 37

Введение

На сегодняшний день учеными-биологами обнаружено и описано большое количество различных видов вирусов. Инфекционные заболевания, вызываемые вирусами, остаются одной из главных проблем современного мира. По оценкам ВОЗ [25], только вспышки гриппа приводят к 500 тысячам смертей по всему миру. Однако помимо гриппа в человеческой популяции циркулирует множество других вирусов (коронавирусы, туберкулёз, малярия, корь и др.), каждый из которых в той или иной мере оказывает влияние на экономику государств и предприятий. Например, из-за пандемии COVID-19 многим организациям и образовательным учреждениям пришлось перейти на дистанционный или смешанный режим работы, что могло привести к дополнительным финансовым затратам для предприятий. Поэтому одной важной задачей систем здравоохранения является разработка эффективных стратегий по борьбе с распространением вирусов, а также поиск способов контроля количества инфицированных. В связи с этим в современном мире возникают, как задачи моделирования эпидемических процессов, так и задачи поиска оптимальных и эффективных стратегий, позволяющих снизить ущерб от эпидемий и пандемий.

Для решения первой задачи в настоящей работе используются актуальные варианты управляемых компартментных эпидемических модели. Современные исследования показывают, что такие модели могут использоваться для оценки числа зараженных и выздоровевших в популяции. За основу были взяты модели [11], которые учитывают циркуляцию сразу нескольких видов вирусов в популяции, а группа инфицированных подразделяется на несколько подгрупп. В рамках текущего научного исследования изучаемые модели были дополнены возможностью управления числом связей между субъектами модели. Для полученной задачи на данный момент критерии оптимальности решения ещё не получены. Именно поэтому для поставленной задачи был выбран алгоритм поиска эффективных управлений - управление с прогнозированием модели (Model Predictive Control, УПМ).

Использование метода управления с прогнозированием модели позволяет решить вторую из поставленных задач, а именно разработку стратегий

по снижению распространения вируса. Сегодня УПМ успешно применяется в управлении технологическими процессами, однако по мнению автора, его концепции также применимы и к эпидемическим процессам. Согласно различным исследованиям, метод показал высокую эффективность в задачах автоматизации химических, энергетических и производственных процессов. Главным преимуществом метода является создание собственных моделей прогнозирования процесса, благодаря которым полученные управляющие сигналы являются эффективными на протяжении интервала времени, превышающем интервал времени управления. По этой причине, контроллеры, использующие метод управления с прогнозированием модели устойчивы шумам и могут адаптироваться к изменениям процесса.

Выпускная квалификационная работа имеет следующую структуру. В главе 1 рассматриваются основные эпидемические модели распространения вирусов, как задачи оптимального управления. В этой же главе проводится линеаризация данных моделей, которая необходима при поиске решения MPC-контроллером, а также модификация параметра отвечающего за интенсивность заражения. Во второй главе описан алгоритм метода управления с прогнозированием модели и представлен пример прогнозирования линейной системы. Глава 3 посвящена компьютерному моделированию описанных задач, а также реализации метода с помощью MPC-контроллеров доступных в языке программирования Matlab. В рамках главы 4 проводится серия численных экспериментов, в ходе которой определяется эффективность полученных решений, а также определяются эффективные управления и траектории модифицированных задач. В заключении формулируются выводы о применимости данного подхода, а также проводится анализ результатов исследования.

Обзор литературы

Можно выделить четыре группы источников, на которые опирается данное исследование:

1. научные статьи и доклады, посвященные моделям распространения вирусов и связанным с ними задачам;

2. научные публикации с применением УПМ в задачах поиска эффективных управлений;

3. сайты, занимающиеся сбором статистики количества заражений и выздоровлений;

4. документация языков программирования Matlab и Scilab.

Первые статьи в области эпидемического моделирования с использованием компартментных моделей появились ещё в 1920-х годах [14]. Гораздо позднее эти модели были модифицированы и стали изучаться как системы с управлением [15]. После этого были получены необходимые условия оптимальности с помощью принципа максимума Понтрягина для задач с несколькими вирусами [10, 11]. После пандемии COVID-19, вызванной вирусом SARS-CoV-2, появилось множество статистической информации [23, 24, 25], которая была использована для улучшения имеющихся эпидемических моделей [2, 9]. В ходе изучения статистических данных удалось обнаружить, что число контактов и частота заражений в значительной степени зависит от мер, вводимых государствами. По этой причине было принято решение внести модификацию в изучаемые модели, добавив возможность управления количеством контактов.

Для решения сформулированной задачи был выбран метод управления с прогнозированием модели. УПМ был предложен в 1950-х годах американским математиком Ричардом Беллманом при разработке оптимального управления динамическими системами. Более подробно метод был изложен в научных статьях 1980-х годов [4, 5, 7, 16]. Изначально подход применялся в обрабатывающей и добывающей промышленности [6, 17]. Благодаря развитию методов компьютерного моделирования и нейронных сетей [18, 21], УМП стал успешно применяться и для более сложных задач [13, 20], таких как управление транспортными средствами [1] и роботами [1], менеджментом электрических сетей [19], а также для контроля сложных химических процессов [3].

Управление с прогнозированием модели реализовано на разных языках программирования, но наиболее популярными считаются библиотеки для

Matlab и Python. В языке программирования Matlab метод реализуется с помощью объекта MPC-контроллер. На сайте компании MathWorks представлено множество примеров реализации УПМ [22], с параллельным и последовательным подключением нескольких MPC-контроллеров. Для долгосрочного планирования в эпидемических задач были использованы последовательно подключенные контроллеры MPC.

Постановка задачи

Целью данной научно-исследовательской работы является создание пакета программ, который применяя метод управления с прогнозированием модели (УПМ), находит эффективные решения эпидемических задач. Разработанный комплекс программ может быть использован для снижения ущерба, связанного с распространением инфекционных заболеваний. Для достижения поставленной цели были решены следующие задачи:

1) изучены современные исследования в области эпидемических моделей и применения метода УПМ;

2) собрана информация о случаях заражения и выздоровления в период эпидемии COVID-19;

3) описаны математические модели эпидемических процессов и сформулированы соответствующие им задачи оптимального управления;

4) проведена модификация изученных моделей, путём ввода возможности управления числом контактов в популяции;

5) разработана компьютерная программа, которая находит эффективное управление в сформулированных задачах с помощью прогнозирования модели; и разработана программа поиска решения методом УПМ;

6) выявлены основные преимущества и недостатки подхода УПМ.

Глава 1. Эпидемические модели

Эпидемические модели – это математические модели, которые применяются при анализе распространения инфекционных заболеваний в популяциях. Компартментные эпидемические модели основаны на представлении популяции в виде нескольких компартментов (групп), каждый из которых соответствует определённой категории индивидов. Такие модели позволяют понять, как быстро распространяется заболевание, а также с помощью каких мер можно снизить распространение вируса.

1.1 Модель SIIR

SIIR (Susceptible-Infected-Recovered) модель – двухвирусная модель эпидемии [11], в которой популяция разделяется на четыре группы:

$$N(t) = n_S(t) + n_{I_1}(t) + n_{I_2}(t) + n_R(t), t \in [t_0, T]. \quad (1)$$

Здесь $N(t)$ – общая численность популяции, $n_S(t)$ – число восприимчивых, $n_{I_1}(t)$ – число инфицированных вирусом V_1, $n_{I_2}(t)$ – число инфицированных вирусом V_2, а $n_R(t)$ – число выздоровевших. Отрезок $[t_0, T]$ – временной интервал, на котором рассматривается эпидемический процесс. Восприимчивые – это группа людей, которые могут заболеть в будущем, но сейчас не заражены. Инфицированные – индивиды, которые на текущий момент времени являются носителями одного из вирусов. Выздоровевшие – группа людей, которые уже была заражены, но выздоровели и имеют иммунитет к повторному заражению. Пусть $S(t), I_1(t), I_2(t), R(t)$ – доли восприимчивых, инфицированных и выздоровевших в популяции. То есть

$$S(t) = \frac{n_S(t)}{N(t)};\ I_1(t) = \frac{n_{I_1}(t)}{N(t)};\ I_2(t) = \frac{n_{I_2}(t)}{N(t)};\ R(t) = \frac{n_R(t)}{N(t)}.$$

Тогда вместо равенства (1) можно записать эквивалентное равенство:

$$1 = S(t) + I_1(t) + I_2(t) + R(t), t \in [t_0, T].$$

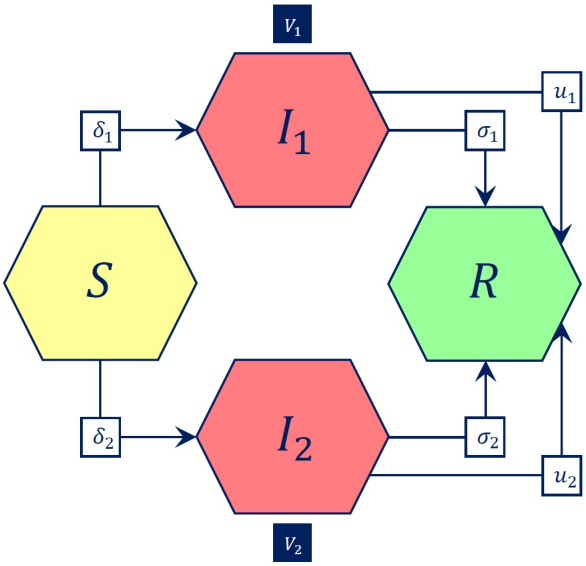

Рис. 1: Схема модели SIIR

На рисунке 1 представлена модель SIIR с двумя вирусами. Для данной модели справедливо следующее допущение: если представитель популяции заразился одним из вирусов, то другим вирусом он заразиться не может. Изменение состояний системы SIIR описывается следующей системой дифференциальных уравнений:

$$\begin{cases} \dot{S}(t) = -\delta_1 S(t) I_1(t) - \delta_2 S(t) I_2(t), \\ \dot{I}_1(t) = (\delta_1 S(t) - \sigma_1 - u_1) I_1(t), \\ \dot{I}_2(t) = (\delta_2 S(t) - \sigma_2 - u_2) I_2(t), \\ \dot{R}(t) = (\sigma_1 + u_1) I_1(t) + (\sigma_2 + u_2) I_2(t). \end{cases} \tag{2}$$

где δ_i – интенсивность заражения вирусом V_i, а σ_i – интенсивность выздоровления, $i = \overline{1,2}$. Интенсивность заражения δ_i представляет собой обобщенный параметр и является произведением среднего числа контактов l на вероятность передачи вируса при данном количестве контакте δ_{i_0}:

$$\delta_i = l\delta_{i_0}.$$

В качестве сокращения числа инфицированных в городской популяции обычно применяют медицинское лечение. Эти меры профилактики можно интерпретировать как функции управления $u = (u_1, u_2)$, где u_i - доля ин-

фицированных, находящихся на интенсивном лечении. При этом функции управления должны принадлежать U – множеству допустимых управлений, которое является подмножеством кусочно-непрерывных функций и удовлетворяет условию:

$$0 \leq u_i \leq u_{max} < 1, \quad i = \overline{1,2}.$$

Общие затраты, связанные с вирусом можно представить как сумму $f_i(I_i)$ – экономического ущерба от заболевания индивидов и $h_i(u_i))$ – стоимости мер, направленных на снижение числа заболевших, $i = \overline{1,2}$. Также предполагается, что существует экономический доход $g(R)$ связанный с выздоровлением инфицированных. По смыслу задачи функции f, h, g – неубывающие, к тому же будем считать, что они дифференцируемы. Тогда суммарные затраты на отрезке $[t_0, T]$ определяются как:

$$\int_{t_0}^{T} [f_1(I_1(t)) + f_2(I_2(t)) - g(R(t)) + h_1(u_1(t)) + h_2(u_2(t))]\, dt \to \min_{u \in U} \qquad (3)$$

Важно отметить, что в моделях SIIR по истечении определенного количества времени процесс эпидемии обычно заканчивается, из-за того, что согласно системе (2) доля восприимчивых $S(t)$ является невозрастающей функцией.

1.2 Модель SIIRS

Модели SIIR, рассмотренные в предыдущем разделе, лучше всего подходят для описания эпидемических процессов, в которых вирусы мутируют медленно. Из-за малого числа мутаций, индивиды могут приобрести иммунитет к вирусу на всю жизнь. Однако существуют вирусы, которые мутируют быстро, и для них были разработаны модели SIIRS (Susceptible-Infected-Recovered-Susceptible) (см. рис. 2), учитывающие снижение иммунитета в популяции с течением времени.

Система дифференциальных уравнений, с помощью которой описыва-

ется динамика изменения состояний, имеет вид:

$$\begin{cases} \dot{S}(t) = \gamma R(t) - \delta_1 S(t) I_1(t) - \delta_2 S(t) I_2(t), \\ \dot{I}_1(t) = (\delta_1 S(t) - \sigma_1 - u_1) I_1(t), \\ \dot{I}_2(t) = (\delta_2 S(t) - \sigma_2 - u_2) I_2(t), \\ \dot{R}(t) = (\sigma_1 + u_1) I_1(t) + (\sigma_2 + u_2) I_2(t) - \gamma R(t). \end{cases} \tag{4}$$

Здесь параметры $\delta_i, \sigma_i, i = \overline{1,2}$, как и в модели SIIR, отвечают за интенсивность заражения и выздоровления, u_i – доля инфицированных, находящихся на медикаментозном лечении, а γ – интенсивность снижения иммунитета у представителей популяции. Целевой функционал для таких задач ничем не отличается от (3).

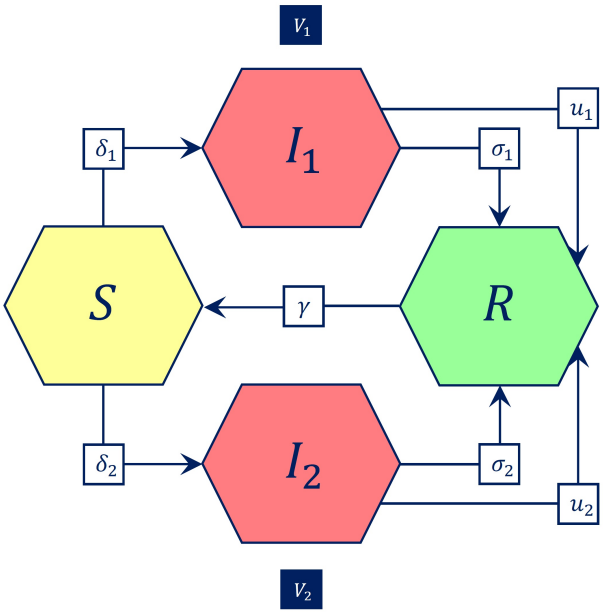

Рис. 2: Схема модели SIIRS

Заметим, что из-за постепенного снижения иммунитета процесс эпидемии в задаче SIIRS может продолжаться бесконечно. По этой причине при решении задач (3-4) необходимо выбирать большие значения для T. Однако в таком случае существует риск (в связи с высокой частотой мутаций у вирусов) того, что параметры модели перестанут быть актуальными. Именно поэтому при решении таких задач могут оказаться эффективными методы, которые

могут адаптироваться к изменениям процесса, благодаря прогнозированию его поведения.

1.3 Модель SWIIRS

В 2020 году, была предложена модель SWIIRS (Susceptible-Warned-Infected-Recovered-Susceptible) [12], которая позволяет учитывать информированность граждан об эпидемии. В данной модели популяция подразделяется на пять групп, вместо четырёх. Дополнительная группа индивидов W – предупрежденные (см. рис. 3). Такой подход позволяет точнее описывать процессы эпидемии в крупных странах, поскольку скорость распространения вируса зачастую зависит не только от характеристик самого вируса, но и от поведения представителей популяции.

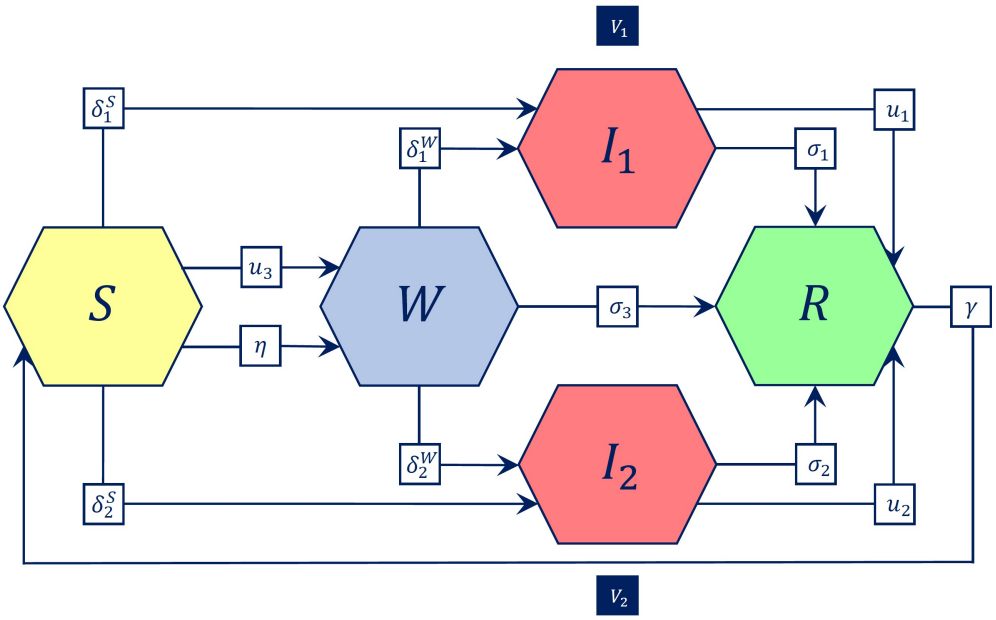

Рис. 3: Схема модели SWIIRS

Изменения популяционных групп для управляемой модели SWIIRS описываются системой из пяти дифференциальных уравнений:

$$\begin{cases} \dot{S}(t) = \gamma R(t) - \delta_1^S S(t) I_1(t) - \delta_2^S S(t) I_2(t) - (\eta W(t) + u_3) S(t), \\ \dot{W}(t) = (\eta W(t) + u_3) S(t) - (\delta_1^W I_1(t) + \delta_2^W I_2(t) + \sigma_3) W(t), \\ \dot{I}_1(t) = (\delta_1^S S(t) + \delta_1^W W(t) - \sigma_1 - u_1) I_1(t), \\ \dot{I}_2(t) = (\delta_2^S S(t) + \delta_2^W W(t) - \sigma_2 - u_2) I_2(t), \\ \dot{R}(t) = (\sigma_1 + u_1) I_1(t) + (\sigma_2 + u_2) I_2(t) + \sigma_3 W(t) - \gamma R(t). \end{cases} \tag{5}$$

В системе (5) константы δ_i^S и δ_i^W описывают интенсивность заражения для групп S и W соответственно, а σ_i – интенсивность выздоровления $i = \overline{1,2}$. Значение σ_3 отвечает за вероятность того, что осведомленный индивид приобретёт иммунитет к вирусам (например обратится в медицинское учреждение за вакциной). Интенсивность снижения иммунитета описывается с помощью γ. За информирование (предупреждение) представителей популяции отвечают два параметра:

1) u_3 – доля восприимчивых индивидов, информируемых через средства массовой информации;

2) η – вероятность информирования восприимчивого индивида при общении с предупрежденным.

1.4 Линеаризация задачи SIIR

В современных программных пакетах управление с прогнозированием модели реализуется для нелинейных управляемых систем с помощью промежуточного решения вспомогательной задачи линеаризации. Опишем алгоритм линеаризации задачи SIIR. Рассмотрим задачу оптимального управления (2-3). Предположим, что начальные состояния для всех состояний $S(t_0)$, $I_1(t_0)$, $I_2(t_0)$, $R(t_0)$ и управление $u(t_0)$ известны. Обозначим за

$$SIIR_{t_0} = \begin{pmatrix} S^0 \\ I_1^0 \\ I_2^0 \\ R^0 \end{pmatrix};\ u^0 = u(t_0). \tag{6}$$

Разобьём исходный отрезок времени $[t_0, T]$ на k равных частей. Тогда целевой функционал (3) может быть записан как:

$$J = \int_{t_0}^{T} [f_1(I_1(t)) + f_2(I_2(t)) - g(R(t)) + h_1(u_1(t)) + h_2(u_2(t))]dt =$$
$$= J_1 + \ldots + J_k, \text{ где}$$
$$J_i = \int_{t_{i-1}}^{t_i} [f_1(I_1(t)) + f_2(I_2(t)) - g(R(t)) + h_1(u_1(t)) + h_2(u_2(t))]dt,$$

здесь $t_i = \frac{iT}{k}, i = \overline{1,k}$.

Очевидно, что $\min\limits_{u\in U} J_1 + \min\limits_{u\in U} J_2 + ... + \min\limits_{u\in U} J_k \leq \min\limits_{u\in U} J$. Тогда вместо исходной задачи (2-3) можно рассматривать k задач с целевой функцией вида:

$$J_i \to \min_{u\in U}. \tag{7}$$

Начальные условия для первой задачи соответствуют (6), а для каждой последующей определяются итеративно подстановкой найденного решения. То есть

$$SI_1I_2R(t_i) = \begin{pmatrix} S(t_i) \\ I_1(t_i) \\ I_2(t_i) \\ R(t_i) \end{pmatrix} = \begin{pmatrix} S^i \\ I_1^i \\ I_2^i \\ R^i \end{pmatrix},\ u^i = u(t_i),\ i = \overline{1,k} - \text{номер итерации.}$$

Для удобства переобозначим $\begin{pmatrix} S \\ I_1 \\ I_2 \\ R \end{pmatrix}$ как вектор $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$. Система

(2) также справедлива и для начальных условий, значит

$$\begin{aligned}
\dot{x}_1^i &= f_1(x^i, u^i, t) = -(\delta_1 x_2^i + \delta_2 x_3^i)x_1^i; \\
\dot{x}_2^i &= f_2(x^i, u^i, t) = (\delta_1 x_1^i - \sigma_1 - u_1^i)x_2^i; \\
\dot{x}_3^i &= f_3(x^i, u^i, t) = (\delta_2 x_1^i - \sigma_2 - u_2^i)x_3^i; \\
\dot{x}_4^i &= f_4(x^i, u^i, t) = (\sigma_1 + u_1^i)x_2^i + (\sigma_2 + u_2^i)x_3^i.
\end{aligned} \tag{8}$$

Рассмотрим некоторое возмущение этой системы:

1) $\widetilde{x}(t) = x^i(t) + \Delta x(t)$, где $\Delta x(t)$ – вектор возмущений состояния;

2) $\widetilde{u}(t) = u^i(t) + \Delta u(t)$, где $\Delta u(t)$ – вектор возмущений управления;

Тогда получаем, что $\dot{x}^i(t) + \Delta\dot{x}(t) = f(\widetilde{x}, \widetilde{u}, t)$. Правые части системы (8) можно разложить в ряд Тейлора:

$$\begin{gathered}
\dot{x}^i(t) + \Delta\dot{x}(t) = \\
= f(x^i(t), u^i(t), t) + \frac{df}{dx}\bigg|_{x^i,u^i} \Delta x(t) + \frac{df}{du}\bigg|_{x^i,u^i} \Delta u(t) + O((\Delta x)^2, (\Delta u)^2).
\end{gathered} \tag{9}$$

Здесь $O((\Delta x)^2, (\Delta u)^2)$ – остаточный член погрешности второго порядка малости. Из формулы (9) вычтем (8) и получим, что

$$\Delta\dot{x}(t) = \underbrace{\frac{df}{dx}\bigg|_{x^i,u^i}}_{A^i(t)} \Delta x(t) + \underbrace{\frac{df}{du}\bigg|_{x^i,u^i}}_{B^i(t)} \Delta u(t).$$

Полученная система дифференциальных уравнений является линейной. Дополнительно можно вывести расчётные формулы для матриц $A^i(t)$ и $B^i(t)$:

$$A^i(t) = \begin{pmatrix}
-(\delta_1 x_2^i + \delta_2 x_3^i) & -\delta_1 x_1^i & -\delta_2 x_1^i & 0 \\
\delta_1 x_2^i & \delta_1 x_1^i - \sigma_1 - u_1^i & 0 & 0 \\
\delta_2 x_3^i & 0 & \delta_2 x_1^i - \sigma_2 - u_2^i & 0 \\
0 & \sigma_1 + u_1^i & \sigma_2 + u_2^i & 0
\end{pmatrix},$$

$$B^i(t) = \begin{pmatrix} 0 & 0 \\ -x_2^i & 0 \\ 0 & -x_3^i \\ x_2^i & x_3^i \end{pmatrix}.$$

Пару $(\overline{x}, \overline{u})$ будем называть приближенным решением, если оно является решением задачи с линейной динамической системой. Пусть пара (x^*, u^*) – точное решение задачи (2,7), тогда обозначим за ошибку в решении величину ε_i, которая определяется формулой:

$$\varepsilon_i = |J_i(\overline{x}, \overline{u}) - J_i(x^*, u^*)|.$$

При $k \to \infty$ получаем:

$$\lim_{k\to\infty} \dot{x}^i = A^i x + B^i u => \lim_{k\to\infty} (\overline{x}, \overline{u}) = (x^*, u^*) => \lim_{k\to\infty} \varepsilon_i = 0.$$

Это означает, что чем больше количество интервалов, на которое разбивается исходный отрезок времени, тем точнее получаемое решение. Линеаризация моделей SIIRS и SWIIRS проводится аналогичным образом.

1.5 Управляемый параметр скорости заражения

Рассмотрим модель (1) с одним вирусом I_i и пусть p_i – вероятность передачи вируса V_i от заболевшего индивида к восприимчивому при единичном контакте, а l – среднее число контактов индивида (с другими представителями популяции) за единицу времени. Рассмотрим случайный контакт восприимчивого индивида. Заражение вирусом произойдёт только в том случае, если одновременно произошли два события: контакт был с зараженным индивидом и вирус передался при контакте.

Определим два несовместных события:

- событие A – при l контактах между восприимчивым и случайными представителями популяции произойдёт заражение вирусом V_i;
- событие B – при аналогичном количестве контактов заражение не случится.

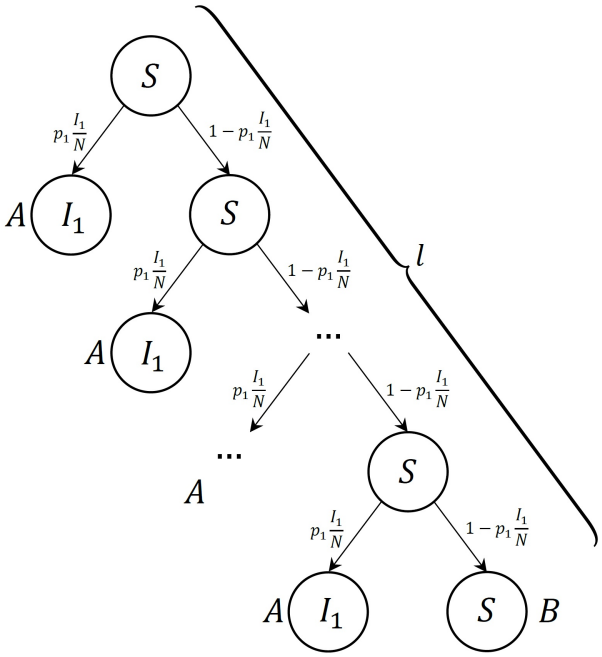

Рис. 4: Граф исходов в модели SIR

Очевидно, что события A и B образуют полную группу событий (см. рис. 4), тогда

$$p(A) + p(B) = 1. \tag{10}$$

При этом событие B произойдёт только в том случае, если при каждом из l контактов заражения не произойдёт. Иными словами

$$p(B) = \left(1 - p_i \frac{I_i(t)}{N}\right)^l. \tag{11}$$

Из соотношений (10) и (11) получаем, что вероятность передачи вируса восприимчивому индивиду при l контактах с инфицированными может быть определена как:

$$p(A) = 1 - \left(1 - p_i \frac{I_i(t)}{N}\right)^l.$$

Эту же формулу можно получить и другим способом. Вероятность события

A представима как:

$$p(A) = p_i\frac{I_i(t)}{N} + \left(1 - p_i\frac{I_i(t)}{N}\right)p_i\frac{I_i(t)}{N} + \left(1 - p_i\frac{I_i(t)}{N}\right)^2 p_i\frac{I_i(t)}{N} +$$

$$... + \left(1 - p_i\frac{I_i(t)}{N}\right)^l p_i\frac{I_i(t)}{N} = \frac{p_i\frac{I_i(t)}{N}\left[\left(1-p_i\frac{I_i(t)}{N}\right)^l - 1\right]}{-p_i\frac{I_i(t)}{N}} = \tag{12}$$

$$= 1 - \left(1 - p_i\frac{I_i(t)}{N}\right)^l.$$

Полученную вероятность $p(A)$ будем называть модифицированным параметром заражения и обозначать $\widetilde{\delta}_i(I_i(t), l)$.

Переобозначим u_i (интенсивное лечение вируса V_i) как u_{i_1}. Введём два вида управления в полученную систему: изоляция инфицированных индивидов и снижение числа контактов во всей системе. Пусть u_{i_2} – доля изолированных инфицированных, а u_l – доля контактов, которые удалось предотвратить, с помощью снижения мобильности индивидов во всей популяции. В таком случае получаем, что

$$\widetilde{\delta}_i(I_i(t), l, u_2, u_3) = \left(1 - p_i\frac{I_i(t)}{N}(1 - u_{i_2})\right)^{l(1-u_l)},$$

где $0 \leq u_{i_2}, u_l \leq 1$. Тогда система дифференциальных уравнений соответствующей одновирусной задачи SIR примет вид:

$$\begin{cases} \dot{S}(t) = -\delta_i S(t) I_i(t), \\ \dot{I}_i(t) = \left(1 - p_i\frac{I_i(t)}{N}(1 - u_{i_2})\right)^{l(1-u_l)} S(t) - (\sigma_1 + u_{i_1})I_i(t), \\ \dot{R}(t) = (\sigma_1 + u_{i_1})I_1(t). \end{cases}$$

Если провести преобразования, аналогичные (12), для двухвирусной модели SIIR то расчётные формулы управляемых параметров заражения бу-

дут иметь вид:

$$\widetilde{\delta}_i(I_1(t), I_2(t), l, u_{1_2}, u_{2_2}, u_l) =$$

$$= \frac{p_i \frac{I_i(t)}{N}(1-u_{i_2})\left[\left(1-p_1\frac{I_1(t)}{N}(1-u_{1_2})-p_2\frac{I_2(t)}{N}(1-u_{2_2})\right)^{l(1-u_l)}-1\right]}{-p_1\frac{I_1(t)}{N}(1-u_{1_2})-p_2\frac{I_2(t)}{N}(1-u_{2_2})},$$

где $i = \overline{1,2}$. Если подставить полученные параметры в систему (2), то дифференциальные уравнения примут следующий вид:

$$\begin{cases} \dot{S}(t) = -\delta_1 S(t) I_1(t) - \delta_2 S(t) I_2(t), \\ \dot{I}_1(t) = \widetilde{\delta}_1(I_1(t), I_2(t), l, u_{1_2}, u_{2_2}, u_l) S(t) - (\sigma_1 + u_1) I_1(t), \\ \dot{I}_2(t) = \widetilde{\delta}_2(I_1(t), I_2(t), l, u_{1_2}, u_{2_2}, u_l) S(t) - (\sigma_2 + u_2) I_2(t), \\ \dot{R}(t) = (\sigma_1 + u_1) I_1(t) + (\sigma_2 + u_2) I_2(t). \end{cases}$$

Для введенных управлений необходимо также определить функции затрат. Пусть $h_{1_2}(u_{1_2}, I_1)$, $h_{2_2}(u_{2_2}, I_2)$, $h_l(u_l)$ – некоторые неубывающие функции соответствующие затратам на управления u_{1_2}, u_{2_2} и u_l. Тогда функционал (3) принимает вид:

$$\widetilde{J} = \int_0^T f_1(I_1(t)) + f_2(I_2(t)) - g(R(t)) + h_l(u_l(t)) + h_{1_1}(u_{1_1}(t)) + \\ + h_{1_2}(u_{1_2}(t) + h_{2_1}(u_{2_1}(t)) + h_{2_2}(u_{2_2}(t) dt.$$

Глава 2. Управление с прогнозированием модели

Управление с прогнозированием модели (Model Predictive Control, MPC) – это комплекс методов поиска эффективных управлений, основанных на прогнозировании модели. Основная идея УПМ состоит в том, чтобы оценить будущее поведение управляемой системы на конечном интервале времени и вычислить эффективный управляющий сигнал, который с учётом ограничений на систему, минимизирует заданный целевой функционал. Алгоритм метода управление с прогнозированием модели включает в себя следующие шаги:

1. Построение модели объекта или процесса с использований технологий математического и компьютерного моделирования;
2. Определение основных целей, которых необходимо достичь посредством управления, выбор горизонта прогнозирования;
3. Построение прогноза траектории процесса под воздействием управляющего сигнала;
4. Поиск эффективного управляющего сигнала с учётом всего комплекса ограничений, наложенных на управляющие и фазовые переменные;
5. Выбор горизонта управления, на котором применяется найденное эффективное управление;
6. Измерение фактических состояний процесса в конечный момент горизонта управления, которые принимаются за новые начальные условия;
7. Сдвиг горизонта прогнозирования на величину горизонта управления и повторение пунктов 3-6.

2.1 Построение УМП для линейных систем

Рассмотрим линейную управляемую и наблюдаемую конечно-разностную систему:

$$\begin{gathered} x(t+1) = Ax(t) + Bu(t), u \in U, \\ y(t) = Cx(t). \end{gathered} \tag{13}$$

Необходимо найти такое управление $u(t)$, которое доставляет минимум следующему функционалу:

$$J(u) = \sum_{\tau=t+1}^{t+N} \left[(y(\tau))^2 + r(u(\tau) - u(\tau-1)^2) \right] \to min, \tag{14}$$

где $r > 0$ – весовой коэффициент, отражающий стоимость смены управления.

Прогнозные значения для задачи (13-14) определяются формулой:

$$\overline{y} = Gx(t) + Hv + Fu(t), \text{где}$$

$x(t), u(t)$ – известные текущие состояния и управления системы,

$$\overline{y} = \begin{pmatrix} \overline{y}(t+1) \\ \overline{y}(t+2) \\ ... \\ \overline{y}(t+N) \end{pmatrix} \text{ – прогноз модели,}$$

$$v = \begin{pmatrix} v(t+1) \\ v(t+2) \\ ... \\ v(t+N) \end{pmatrix} \text{ – планируемые управляющие сигналы,}$$

$$H = \begin{pmatrix} 0 & 0 & ... & 0 \\ CB & 0 & ... & 0 \\ ... & ... & ... & ... \\ CA^{N-2}B & CA^{N-3}B & ... & 0 \end{pmatrix} = \begin{pmatrix} h(1) & 0 & ... & 0 \\ h(2) & h(1) & ... & 0 \\ ... & ... & ... & ... \\ h(N) & h(N-1) & ... & h(1) \end{pmatrix},$$

$$G = \begin{pmatrix} CA \\ CA^2 \\ ... \\ CA^N \end{pmatrix}, F = \begin{pmatrix} h(2) \\ h(3) \\ ... \\ h(N+1) \end{pmatrix}.$$

Тогда получаем задачу квадратичного программирования:

$$\begin{gathered} J(u) = \overline{y}^T \overline{y} + r\left(v^T D^T D v\right) \to min \\ \overline{y}^T = Gx(t) + Hv + Fu(t), v \in U, \end{gathered} \tag{15}$$

где

$$D = \begin{pmatrix} 1 & -1 & 0 & ... & 0 \\ 0 & 1 & -1 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & 0 \end{pmatrix}.$$

Решение полученной задачи производится с помощью стандартных методов решения задач квадратичного программирования. Пусть

$$v^* = \begin{pmatrix} v^*(t+1) \\ v^*(t+2) \\ ... \\ v^*(t+N) \end{pmatrix}$$

решение задачи (15), тогда $v^*(t+1)$ будем называть эффективным управлением для исходной системы (13) на интервале $[t, t+1]$, полученным с помощью метода УПМ. После применения управления получим начальные данные $x(t+1)$ и $u(t+1)$ для составления следующего прогноза и поиска эффективного управления на интервале $[t+1, t+2]$.

Глава 3. Реализация УПМ в Matlab

3.1 Моделирование эпидемических задач

Для применения УПМ к моделям, описанным в главе 1, их необходимо представить в виде функций языка Matlab. Программный код для каждой модели состоит из 4 блоков. Описание блоков программного кода представлено в таблице 1..

Таблица 1: Структура компьютерной эпидемической модели

Название	**Описание**
Блок 1. Параметры	в этой части программного кода задаются характеристики вирусов и системы, такие как среднее число контактов, вероятность заразиться при единичном контакте и скорость выздоровления.
Блок 2. Инициализация состояний и управлений	предназначен для объявления всех переменных состояния и управления.
Блок 3. Вспомогательные вычисления	в данном блоке производится подсчёт промежуточных значений, которые могут неоднократно использоваться в других программных блоках
Блок 4. Уравнения состояний	здесь описывается система дифференциальных уравнений, которая и определяет тип компартментной модели

Также в виде функций Matlab были программно описаны целевые функционалы (см. лист. 2). Такое представление моделей и функционалов значительно упрощает настройку MPC-контроллера.

```
function J = SIIR_Cost(X,U,e,data)
T = data.PredictionHorizon;
U1 = U(1:T,1);
U2 = U(1:T,2);
U3 = U(1:T,3);
```

```
U4 = U(1:T,4);
U5 = U(1:T,5);
X2 = X(2:T+1,2);
X3 = X(2:T+1,3);
X4 = X(2:T+1,4);
J = 5*sum(U1.^2) + 10*sum(U2.^2) + 8*(sum(dot(U3,X2))) + 8*(sum(dot(U4,X3)
    )) + 8*(sum(U5.^2)) + 10*(sum(X2)) + 20*(sum(X3));
end
```

Листинг 1: Целевой функционал в виде функции Matlab

3.2 Подключение MPC-контроллера

После того, как элементы исследуемой задачи были описаны средствами языка программирования Matlab необходимо выполнить настройку MPC-контроллера, который и будет выполнять поиск эффективных управлений, решая вспомогательные задачи описанные в главах 1 и 2. Ознакомиться с программным кодом можно в приложении 1. Описание разделов программного кода представлено в таблице 2.

Таблица 2: Структура компьютерной эпидемической модели

Название	**Описание**
Блок 1. Начальные данные	предназначен для ввода исходных данных задачи таких как: интервал времени, на котором решается задача и начальное состояние системы. Также в этом блоке определяется необходимое количество последовательно подключённых MPC-контроллеров, которые будут задействованы в процессе решения.
Блок 2. SIIR без управления	здесь вычисляются траектории состояний при отсутствии управляющих воздействий. Эти траектории позволяют оценить насколько сильно удалось изменить процесс.
Блок 3.1 Создание MPC-контроллера	в блоке создаётся подходящий для данной модели MPC-контроллер

Блок 3.2. Временные характеристики	вычисление значений для горизонта прогнозирования и управления, а также шага MPC-контроллера
Блок 3.3. Настройка MPC-контроллера	в этой части программного кода производится подключение эпидемической модели и функционала к MPC-контроллеру
Блок 3.4. Ограничения на переменные	здесь объявляются основные ограничения на переменные состояний и управлений
Блок 3.5. Тест на ошибки	в данном блок проводится проверка MPC-контроллера с помощью функции validatFcns на наличие ошибок в параметрах
Блок 3.6. Поиск решения	если ошибки не выявлены в предыдущем блоке, то здесь определяются эффективные управления и состояния. Результаты вычислений записываются в массив
Блок 4. Графический вывод	используется для настройки графического вывода, а также подсчёта значений целевого функционала и выгоды, которой удалось достичь при использовании эффективных управлений

Глава 4. Вычислительные эксперименты

В данном разделе приведены результаты численных экспериментов, которые были использованы для оценки получаемых решений.

4.1 Эксперимент 1

В ходе эксперимента 1 рассматривалась модель SIIR с фиксированной интенсивностью заражения. Данные для этого эксперимента были взяты из статьи Optimal Control of Heterogeneous Mutating Viruses [10]. В таблице 3 представлены значения параметров, которые использовались в процессе симуляции.

Таблица 3: Параметры эксперимента 1

Параметр	**Значение**	**Описание**
Характеристики вируса		
δ_1	0.4	интенсивность заражения вирусом V_1
δ_2	0.5	интенсивность заражения вирусом V_2
σ_1	0.001	интенсивность выздоровления от вируса V_1
σ_2	0.002	интенсивность выздоровления от вируса V_2
Параметры функционала затрат		
$f_1(I_1)$	$5I_1$	ущерб от количества заболевших вирусом V_1
$f_2(I_2)$	$6I_2$	ущерб от количества заболевших вирусом V_2
$g(R)$	$0.1R$	доход от выздоровления
$h_1(u_1)$	$15u_1^2$	стоимость управления u_1
$h_2(u_2))$	$10u_2^2$	стоимость управления u_2
Начальные данные системы		
$S(0)$	0.50	доля восприимчивых
$I_1(0)$	0.32	доля инфицированных вирусом V_1
$I_2(0)$	0.32	доля инфицированных вирусом V_1
$R(0)$	0.00	доля иммунных к обоим вирусам
Параметры контроллера		
k	1	количество контроллеров

Ts	0.5	шаг контроллера
PH	90	горизонт прогнозирования
CH	90	горизонт управления

После проведения компьютерной симуляции было получено решение сформулированной задачи, а также найдены значения функционала (см. рис. 5).

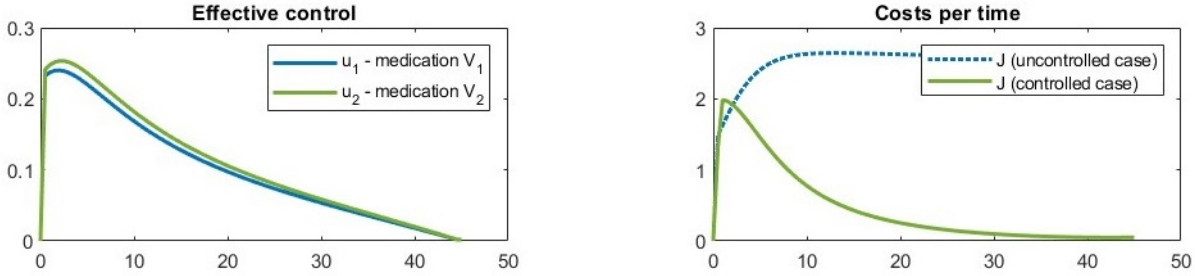

Рис. 5: Управление и затраты для эксперимента 1

Эффективными оказались стратегии которые за первые три дня эпидемии возрастают до значений $u_1 = 0.26$ и $u_2 = 0.24$, а после убывают к нулю. Полученные управляющие сигналы по виду напоминают непрерывные управления, представленные в экспериментах статьи. Значение функционала в управляемом случае составило ≈ 41.6, а в неуправляемом ≈ 227.51. Таким образом, выгода от использования управления составляет ≈ 185.91 условных единиц, что чуть меньше, чем от использования оптимального управления ≈ 189.17.

На рисунке 6 представлены состояния популяции в управляемом и не управляемом случаях. Можно заметить, что использование управления, найденного с помощью MPC, существенно снижает число заражений. Так в неуправляемом случае максимальная доля числа заболевших $I_{1_{max}} = 0.594$, $I_{2_{max}} = 0.386$, а в случае применения эффективного управления $I_{1_{max}} = 0.32$, $I_{2_{max}} = 0.18$, что соответствует начальной доле инфицированных в популяции.

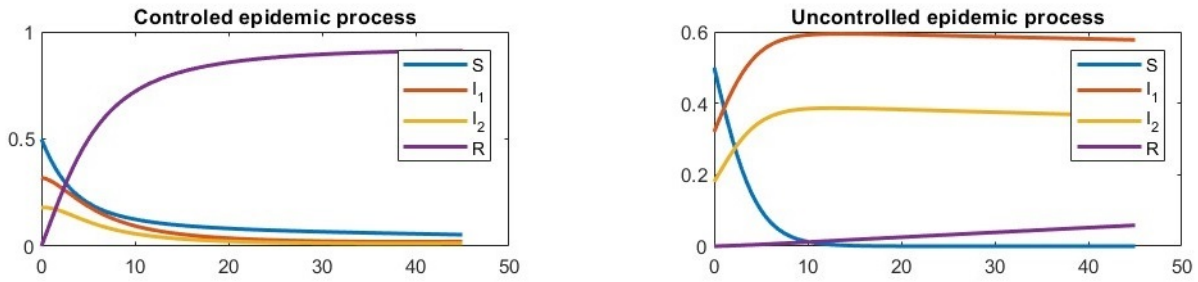

Рис. 6: Управление и затраты для эксперимента 1

Таким образом, эксперимент 1 показал, что разработанный пакет программ позволяет находит близкие к оптимальным решения классических задач SIIR и может быть использован как альтернатива классическим методам оптимального управления.

4.2 Эксперимент 2

Так как программная реализация метода УПМ показала неплохую точность для стандартных задач, то эксперимент 2 был проведён с использованием модифицированной модели SIIRS (с управляемой интенсивностью заражения). Данные для эксперимента были взяты с таких ресурсов как World Health Organization, Our World in Data и Worldometer. Параметры эксперимента представлены в таблице 4.

Таблица 4: Параметры эксперимента 2

Параметр	**Значение**	**Описание**
Характеристики вируса и системы		
δ_1	0.256	вероятность заражения вирусом V_1 при единичном контакте
δ_2	0.517	вероятность заражения вирусом V_2 при единичном контакте
σ_1	0.025	интенсивность выздоровления от вируса V_1
σ_2	0.007	интенсивность выздоровления от вируса V_2
γ	$1/120 \approx 0.0083$	интенсивность снижения иммунитета

l	30,	среднее число контактов в популяции
Параметры функционала затрат		
$f_1(I_1)$	$135I_1$	ущерб от количества заболевших вирусом V_1
$f_2(I_2)$	$135I_2$	ущерб от количества заболевших вирусом V_2
$g(R)$	0	доход от выздоровления
$h_{1_1}(u_{1_1})$	$192(u_{1_1})^2$	стоимость лечения u_{1_1}
$h_{1_2}(u_{1_2}, I_1)$	$94u_{1_2}I_1$	стоимость изоляции u_{1_2} инфицированных I_1
$h_{2_1}(u_{2_1})$	$192(u_{2_1})^2$	стоимость лечения u_{2_1}
$h_{2_2}(u_{2_2}I_2)$	$94u_{2_2}I_2$	стоимость изоляции u_{2_2} инфицированных I_2
$h_l(u_l$	$94(u_l)^2$	стоимость ограничения мобильности популяции u_l
Начальные данные системы		
$S(0)$	0.50	доля восприимчивых
$I_1(0)$	0.32	доля инфицированных вирусом V_1
$I_2(0)$	0.32	доля инфицированных вирусом V_1
$R(0)$	0.00	доля иммунных к обоим вирусам
Параметры контроллера		
k	1	количество контроллеров
Ts	0.5	шаг контроллера
PH	45	горизонт прогнозирования
CH	30	горизонт управления

В ходе запуска программы компьютерной симуляции были получены эффективные управления для сформулированной задачи (см. рис. 7, 8). Можно заметить, что лечение особей для обоих вирусов принимает максимальное значение начиная с $t = 1$ и снижается для вируса V_1 после $t = 21$, а для вируса V_1 после $t = 18$. При этом для V_1 производится изолирование небольшой доли (($u_{1_2} = 0.1$) инфицированных в период $[5, 8]$, и после $t = 15$ управление u_{1_2} снова включается достигая максимального значения и отключается после $t = 24$. Для V_2 максимальное изолирование инфицированных проводится в периоды $[0, 4]$, $[5, 8]$, $[12, 21]$, а также частичное изолирование в $[9, 11]$,

$[29, 30]$. Снижение общего числа контактов оказалось наиболее эффективным в период $[15, 25]$.

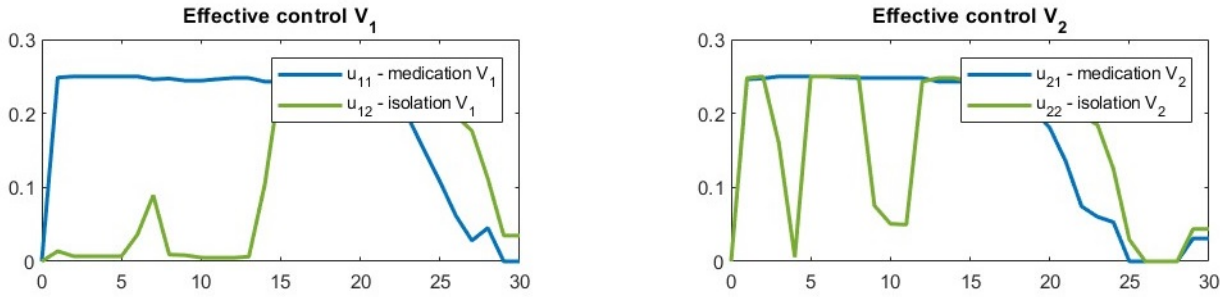

Рис. 7: Лечение и изоляция в эксперименте 2

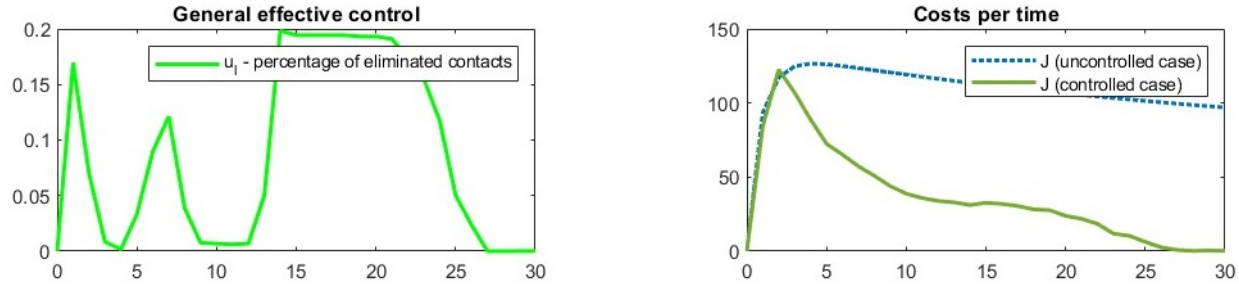

Рис. 8: Общее управление и затраты в эксперименте 2

Используя эффективные стратегии (см. рис. 9) удалось снизить максимальную долю инфицированных вирусом V_2 с 0.6 до 0.3, однако доля заболевших вирусом V_1 возросла с 0.34 до 0.38, это связано с тем, что из-за часть индивидов (30% популяции), которые не заразились вирусом V_2 (из-за введенных мер), смогли заразиться вирусом V_1. Также, можно заметить, что начиная с $t = 15$ доля инфицированных обоими вирусами не превышает 1%.

Для задач с модифицированным параметром заражения на данный момент не найдены условия оптимальности, поэтому единственным способом оценки качества решения является экономия затрат, которую обеспечивает построенное эффективное управление. По этой причине для данного эксперимента были посчитаны значения целевого функционала и общее управление для двух вирусов (см. рис. 9). С помощью совокупности всех управляю-

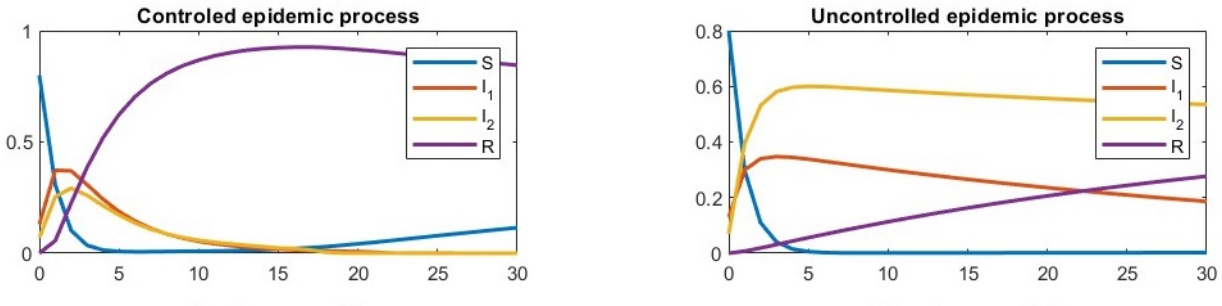

Рис. 9: Эпидемические процессы эксперимента 2

щих воздействий удалось значительно снизить значение целевого функционала 33286 до 11180. Таким образом, выгода от использования эффективного управления составила 22106 единиц.

4.3 Эксперимент 3

Для эксперимента 3 была взята модель SWIIRS, а основной задачей являлось построение долгосрочных стратегий ($T \geq 90$), с помощью нескольких MPC-контроллеров. Значения параметров эксперимента доступны в таблице 5.

Таблица 5: Параметры эксперимента 3

Параметр	**Значение**	**Описание**
Характеристики вируса		
δ_1^S	0.25	интенсивность заражения вирусом V_1 восприимчивых
δ_2^S	0.3	интенсивность заражения вирусом V_2 восприимчивых
δ_1^W	0.2	интенсивность заражения вирусом V_1 предупрежденных
δ_2^W	0.25	интенсивность заражения вирусом V_2 предупрежденных
σ_1	0.3	интенсивность выздоровления от вируса V_1
σ_2	0.4	интенсивность выздоровления от вируса V_2

σ_3	0.3	вероятность вакцинации предупрежденных
γ	0.3	интенсивность снижения иммунитета
η	0.15	интенсивность передачи информации о вирусе
Параметры функционала затрат		
$f_1(I_1)$	$30I_1$	ущерб от количества заболевших вирусом V_1
$f_2(I_2)$	$40I_2$	ущерб от количества заболевших вирусом V_2
$g(R)$	0	доход от выздоровления
$h_1(u_1)$	$20u_1^2$	стоимость управления u_1
$h_2(u_2))$	$25u_2^2$	стоимость управления u_2
$h_3(u_3))$	$10u_3^2$	стоимость управления u_3
Начальные данные системы		
$S(0)$	0.9	доля восприимчивых
$W(0)$	0.00	доля предупрежденных
$I_1(0)$	0.08	доля инфицированных вирусом V_1
$I_2(0)$	0.02	доля инфицированных вирусом V_2
$R(0)$	0.00	доля иммунных к обоим вирусам
Параметры контроллера		
k	3	количество контроллеров
Ts	0.5	шаг контроллера
PH	45	горизонт прогнозирования
CH	30	горизонт управления

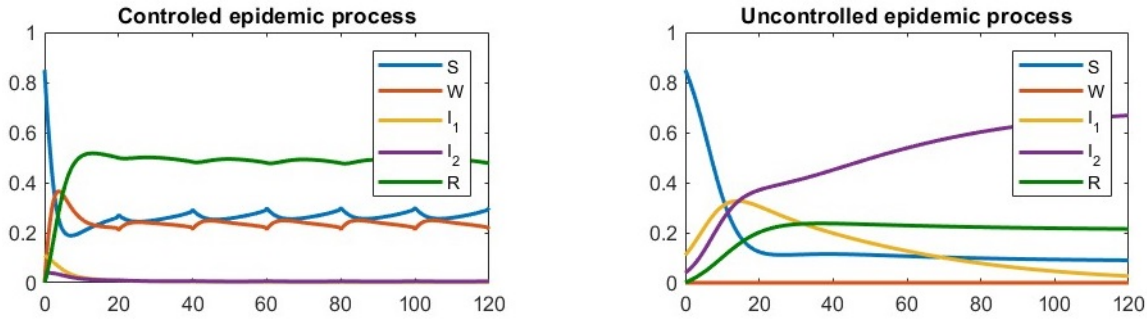

Рис. 10: Эпидемические процессы эксперимента 3

На рисунках 10, 11 представлены результаты эксперимента 3. На графиках эффективных управлений видно, что в случае подключения нескольких MPC-контроллеров, существует импульсы (точки переключения) в управлении. Они связаны с тем, что происходит пересчёт модели прогноза на новый интервал прогнозирования. Можно заметить, что начиная с третьего контроллера ($t = 60$) построенные управления имеют схожую структуру. Это отражается как на поведении процесса, так и на поведение функции суммарных затрат (они становится периодичными).

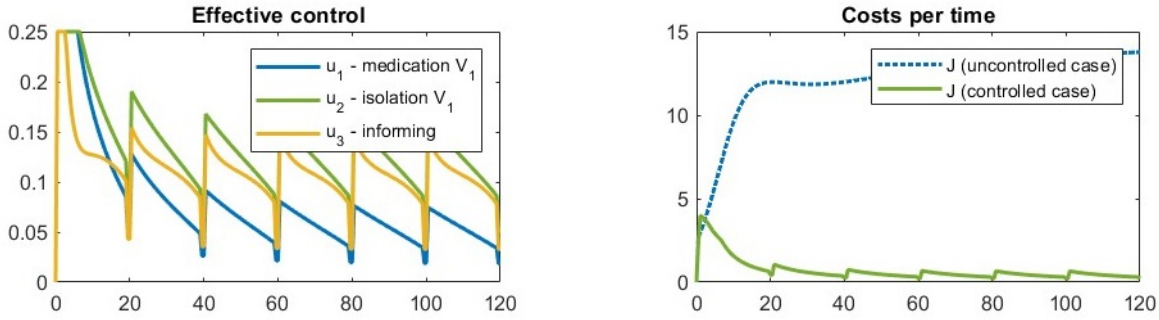

Рис. 11: Эффективные управления и затраты в эксперименте 3

Благодаря применению эффективного управления удалось значительно снизить значение целевого функционала: с 2918.8 до 177.8. Таким образом, выгода от применения управления составила 2748 единиц.

Заключение

В данной выпускной квалификационной работе было проведено исследование о применимости метода управления с прогнозированием модели для решения задачи SIIR с двумя вирусами и её модификаций. Помимо известных вариаций моделей SIIR, описанных в других научных исследованиях, была предложена и описана собственная модель с контролируемой интенсивностью заражения. Также в работе были представлены алгоритмы решения вспомогательных задач, такие как линеаризация модели, построение прогнозных значений, поиск эффективных управлений. После этого был подробно описан процесс компьютерного моделирования задач SIIR и применение к ним УПМ. Для подтверждения теоретических результатов и оценки эффективности метода была проведена серия экспериментов, которая показала, что решения, полученные с использованием метода УПМ близки к оптимальным (получаемым с помощью принципа максимума) и значительно сокращают затраты в изучаемых задачах. Метод показал высокую эффективность независимо от типа задачи и интервала времени. Таким образом, УПМ может быть использован как альтернатива имеющимся методам управления в эпидемических задачах. Однако стоит помнить, что в процессе вычисления, метод использует численные алгоритмы, что не позволяет использовать метод для поиска точных оптимальных решений. Таким образом, поставленная цель исследования была успешно выполнена.

Список литературы

[1] Alcalá E. LPV-MPC Control for Autonomous Vehicles / E. Alcalá, V. Puig, J. Quevedo // LPV-MPC Control for Autonomous Vehicles. – 2019. – Vol.52, №28 – P. 106-113.

[2] Berger A. An SEIR infectious disease model with testing and conditional quarantine / A. Berger, K. Herkenhoff, S. Mongey // NBER Working Paper Series. – 2020. – P. 1-30.

[3] Biegler L. A perspective on nonlinear model predictive control / L. Biegler // Korean Journal of Chemical Engineering. – 2021. – №7. – P. 1317–1332.

[4] Clarke D. Generalized predictive control—Part I. The basic algorithm / D. Clarke, C. Mohtadi, P. Tuffs // Automatica. – 1987. – Vol.23, №2 – P. 137-148.

[5] Clarke D. Generalized Predictive Control—Part II Extensions and interpretations / D. Clarke, C. Mohtadi, P. Tuffs // Automatica. – 1987. – Vol.23, №2 – P. 149-160.

[6] Cutler C. Constrained multivariable control of a hydrocracker reactor / C. Cutler, R. Hawkins // American Control Conference. – 1987. – P. 1014-1020.

[7] García C. Model predictive control: Theory and practice—A survey / C. García, D. Prett, M. Morari // Automatica. – 1989. – Vol.25, №3 – P. 335-348.

[8] Gold T. Model Predictive Interaction Control for Industrial Robots / T. Gold, A. Völz, K. Graichen // IFAC-PapersOnLine. – 2020. – Vol.53, №2 – P. 1-16.

[9] Grimm V. Extensions of the SEIR model for the analysis of tailored social distancing and tracing approaches to cope with COVID-19 / V. Grimm, F. Mengel, M. Schmidt // Sci Rep. – 2021. – Vol.11, №1 – P. 1-16.

[10] Gubar E. Optimal control of heterogeneous mutating viruses / E. Gubar, V. Taynitskiy, Q. Zhu // Games. – 2018. – №4. – P. 1-18.

[11] Gubar E. Optimal Control of Influenza Epidemic Model with Virus Mutations / E. Gubar, Z. Quanyan // European Control Conference (ECC). – 2013. – P. 3125-3130.

[12] Gubar E. Optimal Control of Joint Multi-Virus Infection and Information Spreading / E. Gubar et al. // IFAC-PapersOnLine. – 2020. – Vol.53, №2. – P. 6650-6655.

[13] Jiang Q. Data-Driven Model Predictive Monitoring for Dynamic Processes / Q. Jiang et al // IFAC-PapersOnLine. – 2020. – Vol.53, №2 P. 98-103

[14] Kermack W. A contribution to the mathematical theory of epidemics / W. Kermack, A. McKendrick // Proc. of the Royal Society. – 1927. – P. 700-721.

[15] Khouzani M. Optimal control of epidemic evolution / M. Khouzani, S. Sarkar, E. Altman // 2011 Proceedings IEEE INFOCOM. – 2011. – P. 1683-1691.

[16] Richalet J. Model predictive heuristic control: Applications to industrial processes / J. Richalet et al. // Automatica. – 1978. – №5. – P. 413-428.

[17] Ricker N. Case studies of model-predictive control in pulp and paper production / N. Ricker, T. Subrahmanian, T. Sim // IFAC Proceedings Volumes. – 1988. – №4. – P. 13-22.

[18] Schimperna I. On offset-free Model Predictive Control with Long Short-Term Memory Networks / I. Schimperna, C. Toffanin, L. Magni, // IFAC-PapersOnLine. – 2023. – Vol.56, №1 P. 156-161.

[19] Skjong E. Distributed control architecture for real-time model predictive control for system-level harmonic mitigation in power systems / E. Skjong, T. Johansen, M. Molinas, // ISA Transactions. – 2019. – Vol.93, №1 P. 231-243.

[20] Wei L. A Data-Driven Predictive Control Structure in the Behavioral Framework / L. Wei, Yan Y., Bao J. // IFAC-PapersOnLine. – 2020. – Vol.53, №2 P. 152-157.

[21] Xiao Y. Deep Neural Networks With Koopman Operators for Modeling and Control of Autonomous Vehicles / Y. Xiao et al. // IEEE Transactions on Intelligent Vehicles. – 2023. – Vol.8, №2 P. 135-146.

[22] Automated driving using model predictive control [Electronic resource] // Сайт MathWorks. – Режим доступа : https://www.mathworks.com/help/mpc/ug/automated-driving-using-model-predictive-control.html (12.10.2022).

[23] Coronavirus (COVID-19) Vaccinations [Electronic resource] // Сайт Our World in Data. – Режим доступа : https://ourworldindata.org/covid-vaccinations (30.01.2023).

[24] COVID - Coronavirus Statisctics [Electronic resource] // Сайт Worldometer. – Режим доступа : https://www.worldometers.info/coronavirus/ (27.01.2023).

[25] WHO Coronavirus (COVID-19) Dashboard [Electronic resource] // Сайт World Health Organization. – Режим доступа : https://covid19.who.int/ (27.01.2023).

Приложение 1

```
clc
clear all

% Model Predictive Control

%--------------------------------------------------------------------------
% Case 1: Initial data
%--------------------------------------------------------------------------

T = 45;
n = 90;
k = 1;
x_i = [0.5; 0.32; 0.18; 0.00];
u_i = [0; 0];
seq_x = x_i;
seq_u = u_i;


%--------------------------------------------------------------------------
% Case 2: SIIR model without control
%--------------------------------------------------------------------------
time = linspace(0,T,n+1);
sol = ode45(@(t, x) stn_SIIR_process(x, [0.0, 0.0]), ...
                [0 T], ...
                x_i, ...
                []);
uncon_x = deval(sol,time);

%--------------------------------------------------------------------------
% Case 3: Controllers sequence
%--------------------------------------------------------------------------
for i = 1:k
    %----------------------------------------------------------------------
    % Case 3.1: Create new MPC Controller
    %----------------------------------------------------------------------
    nx = length(x_i);
    ny = length(x_i);
    nu = length(u_i);
    nlobj = nlmpc(nx,ny,nu);

    %----------------------------------------------------------------------
    % Case 3.2: Define time parameters
    %----------------------------------------------------------------------
    Ts = T/n;
```

```
45      CH = n/k;
46      PH = CH;
47
48      %-------------------------------------------------------------------
49      % Case 3.3: Setup MPC controller
50      %-------------------------------------------------------------------
51      nlobj.Ts = Ts;
52      nlobj.PredictionHorizon = PH;
53      nlobj.ControlHorizon = CH;
54
55      nlobj.Model.StateFcn = 'stn_SIIR_process';
56      nlobj.Model.OutputFcn = @(x,u) [x(1); x(2); x(3); x(4)];
57
58      nlobj.Optimization.CustomCostFcn =  'stn_SIIR_Cost';
59      nlobj.Optimization.ReplaceStandardCost = true;
60      % nlobj.Optimization.CustomEqConFcn = @(X,U,data) sum(sum(X)) = 1;
61
62      %-------------------------------------------------------------------
63      % Case 3.4: Constraints for variables
64      %-------------------------------------------------------------------
65      for ct = 1:nx
66          nlobj.States(ct).Min = 0;
67          nlobj.States(ct).Max = 1;
68      end
69
70      for ct = 1:nu
71          nlobj.MV(ct).Min = 0;
72          nlobj.MV(ct).Max = 1;
73      end
74
75      %-------------------------------------------------------------------
76      % Case 3.5: Error test
77      %-------------------------------------------------------------------
78      x_0 = x_i;
79      u_0 = u_i;
80      validateFcns(nlobj,x_0,u_0);
81
82      %-------------------------------------------------------------------
83      % Case 3.6: Find the solution
84      %-------------------------------------------------------------------
85      [~,~,info] = nlmpcmove(nlobj,x_0,u_0);
86
87      seq_x = cat(2, seq_x, transpose(info.Xopt(2:CH+1,:)));
88      seq_u = cat(2, seq_u, transpose(info.MVopt(2:CH+1,:)));
89      x_i = [info.Xopt(CH+1,1); info.Xopt(CH+1,2); info.Xopt(CH+1,3); info.
    Xopt(CH+1,4)];
```

```
    u_i = [info.MVopt(CH,1); info.MVopt(CH,2)];
end

%-------------------------------------------------------------------------
% Case 4: Graphical output
%-------------------------------------------------------------------------
colors = validatecolor({'#0072BD', '#D95319', '#EDB120', '#7E2F8E', '
    #008000'},'multiple');
tiledlayout(2,2)

%-------------------------------------------------------------------------
% Case 4.1: State Graph
%-------------------------------------------------------------------------
nexttile
p1 = plot(time, seq_x);

for ct = 1:nx
    p1(ct).LineWidth = 1.8;
    p1(ct).Color = colors(ct,:);
end
legend('S','I_1','I_2','R')
title('Controled epidemic process')

hold on

nexttile
p2 = plot(time, uncon_x);
for ct = 1:nx
    p2(ct).LineWidth = 1.8;
    p2(ct).Color = colors(ct,:);
end

legend('S','I_1','I_2','R')
title('Uncontrolled epidemic process')

%-------------------------------------------------------------------------
% Case 4.2: Control Graph
%-------------------------------------------------------------------------
nexttile
p3 = plot(time, seq_u(:,1:CH*k+1));
p3(2).Color = '#77AC30';
for ct = 1:2
    p3(ct).LineWidth = 1.8;
end
legend('u_1 - medication V_1','u_2 - medication V_2')
title('Effective control')
```

```
135
136 %-------------------------------------------------------------------------
137 % Case 4.3: Costs Graph
138 %-------------------------------------------------------------------------
139 costs = zeros(2,n+1);
140 for i = 2:n+1
141     tdata.PredictionHorizon = 1;
142     costs(1,i) = stn_SIIR_Cost(transpose(uncon_x(:,i-1:i)), transpose(
    zeros(5,i)), 0, tdata) * Ts;
143     costs(2,i) = stn_SIIR_Cost(transpose(seq_x(:,i-1:i)), transpose(seq_u
    (:,i-1:i)), 0, tdata) * Ts;
144 end
145
146 nexttile
147 p4 = plot(time, costs);
148 p4(1).LineStyle = ':';
149 p4(2).Color =   '#77AC30';
150 for ct = 1:2
151     p4(ct).LineWidth = 1.8;
152 end
153 legend('J (uncontrolled case)','J (controlled case)')
154 title('Costs per time')
155
156 sim_data.PredictionHorizon = n;
157 stand_cost = stn_SIIR_Cost(transpose(uncon_x), transpose(zeros(5,n)), 0,
    sim_data)* Ts
158 eff_cost = stn_SIIR_Cost(transpose(seq_x), transpose(seq_u(:,2:n+1)), 0,
    sim_data)* Ts
159 profit = stand_cost - eff_cost
```

Листинг 2: Программный код эксперимента 1

```
1 clc
2 clear all
3
4 % Model Predictive Control
5
6 %-------------------------------------------------------------------------
7 % Case 1: Initial data
8 %-------------------------------------------------------------------------
9
10 T = 30;
11 n = 60;
12 k = 1;
13 x_i = [0.8; 0.13; 0.07; 0.00];
14 u_i = [0.0; 0.0; 0.0; 0.0; 0.0];
15 seq_x = x_i;
```

```
seq_u = u_i;


%-------------------------------------------------------------------------
% Case 2: SIIR model without control
%-------------------------------------------------------------------------
time = linspace(0,T,n+1);
sol = ode45(@(t, x) SIIRS_process(x, u_i), ...
                [0 T], ...
                x_i, ...
                []);
uncon_x = deval(sol,time);

%-------------------------------------------------------------------------
% Case 3: Controllers sequence
%-------------------------------------------------------------------------
for i = 1:k
    %-------------------------------------------------------------------------
    % Case 3.1: Create new MPC Controller
    %-------------------------------------------------------------------------
    nx = length(x_i);
    ny = length(x_i);
    nu = length(u_i);
    nlobj = nlmpc(nx,ny,nu);

    %-------------------------------------------------------------------------
    % Case 3.2: Define time parameters
    %-------------------------------------------------------------------------
    Ts = T/n;
    CH = n/k;
    PH = 1.5*CH;

    %-------------------------------------------------------------------------
    % Case 3.3: Setup MPC controller
    %-------------------------------------------------------------------------
    nlobj.Ts = Ts;
    nlobj.PredictionHorizon = PH;
    nlobj.ControlHorizon = CH;

    nlobj.Model.StateFcn = 'SIIRS_process';
    nlobj.Model.OutputFcn = @(x,u) [x(1); x(2); x(3); x(4)];

    nlobj.Optimization.CustomCostFcn =  'SIIRS_Cost';
    nlobj.Optimization.ReplaceStandardCost = true;
    % nlobj.Optimization.CustomEqConFcn = @(X,U,data) sum(sum(X)) = 1;

```

```
62      %---------------------------------------------------------------------
63      % Case 3.4: Constraints for variables
64      %---------------------------------------------------------------------
65      for ct = 1:nx
66          nlobj.States(ct).Min = 0;
67          nlobj.States(ct).Max = 1;
68      end
69
70      for ct = 1:nu
71          nlobj.MV(ct).Min = 0;
72          nlobj.MV(ct).Max = 0.25;
73      end
74
75      nlobj.MV(3).Max = 0.25;
76      nlobj.MV(4).Max = 0.25;
77      nlobj.MV(5).Max = 0.2;
78
79      %---------------------------------------------------------------------
80      % Case 3.5: Error test
81      %---------------------------------------------------------------------
82      x_0 = x_i;
83      u_0 = u_i;
84      validateFcns(nlobj,x_0,u_0);
85
86      %---------------------------------------------------------------------
87      % Case 3.6: Find the solution
88      %---------------------------------------------------------------------
89      [~,~,info] = nlmpcmove(nlobj,x_0,u_0);
90
91      seq_x = cat(2, seq_x, transpose(info.Xopt(2:CH+1,:)));
92      seq_u = cat(2, seq_u, transpose(info.MVopt(2:CH+1,:)));
93      x_i = [info.Xopt(CH+1,1); info.Xopt(CH+1,2); info.Xopt(CH+1,3); info.
        Xopt(CH+1,4)];
94      u_i = [info.MVopt(CH,1); info.MVopt(CH,2); info.MVopt(CH,3); info.
        MVopt(CH,4); info.MVopt(CH,5)];
95  end
96
97  %-------------------------------------------------------------------------
98  % Case 4: Graphical output
99  %-------------------------------------------------------------------------
100 colors = validatecolor({'#0072BD', '#D95319', '#EDB120', '#7E2F8E', '
        #008000'},'multiple');
101 tiledlayout(3,2)
102
103 %-------------------------------------------------------------------------
104 % Case 4.1: State Graph
```

```
%-------------------------------------------------------------------------
nexttile
p1 = plot(time, seq_x);

for ct = 1:nx
    p1(ct).LineWidth = 1.8;
    p1(ct).Color = colors(ct,:);
end
legend('S','I_1','I_2','R')
title('Controled epidemic process')

hold on

nexttile
p2 = plot(time, uncon_x);
for ct = 1:nx
    p2(ct).LineWidth = 1.8;
    p2(ct).Color = colors(ct,:);
end

legend('S','I_1','I_2','R')
title('Uncontrolled epidemic process')

%-------------------------------------------------------------------------
% Case 4.2: Control Graph
%-------------------------------------------------------------------------
nexttile
p3 = plot(time, seq_u([1,3],1:CH*k+1));
p3(2).Color = '#77AC30';
for ct = 1:2
    p3(ct).LineWidth = 1.8;
end
legend('u_1_1 - medication V_1','u_1_2 - isolation V_1')
title('Effective control V_1')

nexttile
p4 = plot(time, seq_u([2,4],1:CH*k+1));
p4(2).Color = '#77AC30';
for ct = 1:2
    p4(ct).LineWidth = 1.8;
end
legend('u_2_1 - medication V_2','u_2_2 - isolation V_2')
title('Effective control V_2')

nexttile
p5 = plot(time, seq_u(5,1:CH*k+1));
```

```
p5(1).Color = 'green';
p5(1).LineWidth = 1.8;
legend('u_l - percentage of eliminated contacts')
title('General effective control')

%-------------------------------------------------------------------------
% Case 4.3: Costs Graph
%-------------------------------------------------------------------------
costs = zeros(2,n+1);
for i = 2:n+1
    tdata.PredictionHorizon = 1;
    costs(1,i) = SIIRS_Cost(transpose(uncon_x(:,i-1:i)), transpose(zeros
    (5,i)), 0, tdata) * Ts;
    costs(2,i) = SIIRS_Cost(transpose(seq_x(:,i-1:i)), transpose(seq_u(:,i
    -1:i)), 0, tdata) * Ts;
end

nexttile
p4 = plot(time, costs);
p4(1).LineStyle = ':';
p4(2).Color = '#77AC30';
for ct = 1:2
    p4(ct).LineWidth = 1.8;
end
legend('J (uncontrolled case)','J (controlled case)')
title('Costs per time')

sim_data.PredictionHorizon = n;
stand_cost = SIIRS_Cost(transpose(uncon_x), transpose(zeros(5,n)), 0,
    sim_data)* Ts
eff_cost = SIIRS_Cost(transpose(seq_x), transpose(seq_u(:,2:n+1)), 0,
    sim_data)* Ts
profit = stand_cost - eff_cost
```

Листинг 3: Программный код эксперимента 2

```
clc
clear all

% Model Predictive Control

%-------------------------------------------------------------------------
% Case 1: Initial data
%-------------------------------------------------------------------------

T = 120;
n = 240;
```

```
k = 6;
x_i = [0.85; 0.0; 0.11; 0.04; 0.00];
u_i = [0.0; 0.0; 0.0];
seq_x = x_i;
seq_u = u_i;


%--------------------------------------------------------------------------
% Case 2: SIIR model without control
%--------------------------------------------------------------------------
time = linspace(0,T,n+1);
sol = ode45(@(t, x) SWIIRS_process(x, u_i), ...
                [0 T], ...
                x_i, ...
                []);
uncon_x = deval(sol,time);

%--------------------------------------------------------------------------
% Case 3: Controllers sequence
%--------------------------------------------------------------------------
for i = 1:k
    %----------------------------------------------------------------------
    % Case 3.1: Create new MPC Controller
    %----------------------------------------------------------------------
    nx = length(x_i);
    ny = length(x_i);
    nu = length(u_i);
    nlobj = nlmpc(nx,ny,nu);

    %----------------------------------------------------------------------
    % Case 3.2: Define time parameters
    %----------------------------------------------------------------------
    Ts = T/n;
    CH = n/k;
    PH = 1.5*CH;

    %----------------------------------------------------------------------
    % Case 3.3: Setup MPC controller
    %----------------------------------------------------------------------
    nlobj.Ts = Ts;
    nlobj.PredictionHorizon = PH;
    nlobj.ControlHorizon = CH;

    nlobj.Model.StateFcn = 'SWIIRS_process';
    nlobj.Model.OutputFcn = @(x,u) [x(1); x(2); x(3); x(4); x(5)];

```

```
58      nlobj.Optimization.CustomCostFcn =  'SWIIRS_Cost';
59      nlobj.Optimization.ReplaceStandardCost = true;
60      % nlobj.Optimization.CustomEqConFcn = @(X,U,data) sum(sum(X)) = 1;
61
62      %-----------------------------------------------------------------------
63      % Case 3.4: Constraints for variables
64      %-----------------------------------------------------------------------
65      for ct = 1:nx
66          nlobj.States(ct).Min = 0;
67          nlobj.States(ct).Max = 1;
68      end
69
70      for ct = 1:nu
71          nlobj.MV(ct).Min = 0;
72          nlobj.MV(ct).Max = 0.25;
73      end
74
75      %-----------------------------------------------------------------------
76      % Case 3.5: Error test
77      %-----------------------------------------------------------------------
78      x_0 = x_i;
79      u_0 = u_i;
80      validateFcns(nlobj,x_0,u_0);
81
82      %-----------------------------------------------------------------------
83      % Case 3.6: Find the solution
84      %-----------------------------------------------------------------------
85      [~,~,info] = nlmpcmove(nlobj,x_0,u_0);
86
87      seq_x = cat(2, seq_x, transpose(info.Xopt(2:CH+1,:)));
88      seq_u = cat(2, seq_u, transpose(info.MVopt(2:CH+1,:)));
89      x_i = [info.Xopt(CH+1,1); info.Xopt(CH+1,2); info.Xopt(CH+1,3); info.
    Xopt(CH+1,4); info.Xopt(CH+1,5)];
90      u_i = [info.MVopt(CH,1); info.MVopt(CH,2); info.MVopt(CH,3)];
91  end
92
93  %-----------------------------------------------------------------------
94  % Case 4: Graphical output
95  %-----------------------------------------------------------------------
96  colors = validatecolor({'#0072BD', '#D95319', '#EDB120', '#7E2F8E', '
    #008000'},'multiple');
97  tiledlayout(2,2)
98
99  %-----------------------------------------------------------------------
100 % Case 4.1: State Graph
101 %-----------------------------------------------------------------------
```

```
nexttile
p1 = plot(time, seq_x);

for ct = 1:nx
    p1(ct).LineWidth = 1.8;
    p1(ct).Color = colors(ct,:);
end
legend('S','W','I_1','I_2','R')
title('Controled epidemic process')

hold on

nexttile
p2 = plot(time, uncon_x);
for ct = 1:nx
    p2(ct).LineWidth = 1.8;
    p2(ct).Color = colors(ct,:);
end

legend('S','W','I_1','I_2','R')
title('Uncontrolled epidemic process')

%-------------------------------------------------------------------------
% Case 4.2: Control Graph
%-------------------------------------------------------------------------
nexttile
p3 = plot(time, seq_u(:,1:CH*k+1));
p3(2).Color = '#77AC30';
for ct = 1:3
    p3(ct).LineWidth = 1.8;
end
legend('u_1 - medication V_1','u_2 - isolation V_1', 'u_3 - informing')
title('Effective control')

%-------------------------------------------------------------------------
% Case 4.3: Costs Graph
%-------------------------------------------------------------------------
costs = zeros(2,n+1);
for i = 2:n+1
    tdata.PredictionHorizon = 1;
    costs(1,i) = SWIIRS_Cost(transpose(uncon_x(:,i-1:i)), transpose(zeros
    (5,i)), 0, tdata) * Ts;
    costs(2,i) = SWIIRS_Cost(transpose(seq_x(:,i-1:i)), transpose(seq_u(:,
    i-1:i)), 0, tdata) * Ts;
end

```

```
nexttile
p4 = plot(time, costs);
p4(1).LineStyle = ':';
p4(2).Color = '#77AC30';
for ct = 1:2
    p4(ct).LineWidth = 1.8;
end
legend('J (uncontrolled case)','J (controlled case)')
title('Costs per time')

sim_data.PredictionHorizon = n;
stand_cost = SWIIRS_Cost(transpose(uncon_x), transpose(zeros(5,n)), 0,
    sim_data)* Ts
eff_cost = SWIIRS_Cost(transpose(seq_x), transpose(seq_u(:,2:n+1)), 0,
    sim_data)* Ts
profit = stand_cost - eff_cost
```

Листинг 4: Программный код эксперимента 3