Одеський національний університет імені І. І. Мечникова
Факультет математики, фізики та інформаційних технологій
Кафедра математичного та комп'ютерного моделювання

Дипломна робота

бакалавра

на тему: «Дослідження однієї динамічної системи на стійкість»

«Research of stability of one dynamical system»

Виконала: студентка денної форми навчання
спеціальності 113 Прикладна математика

Колчинська Яна Євгенівна

Керівник: к. ф.-м. н., доц. Таірова М. С.
Рецензент: к. ф.-м. н., доц. Васильєв О. Б.

Рекомендовано до захисту:
Протокол засідання кафедри
№ ___ від _____ р.

Завідувач кафедри

_____ _____
(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК №_____
протокол № ____ від _____ р.
Оцінка_____/___ ___/_____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

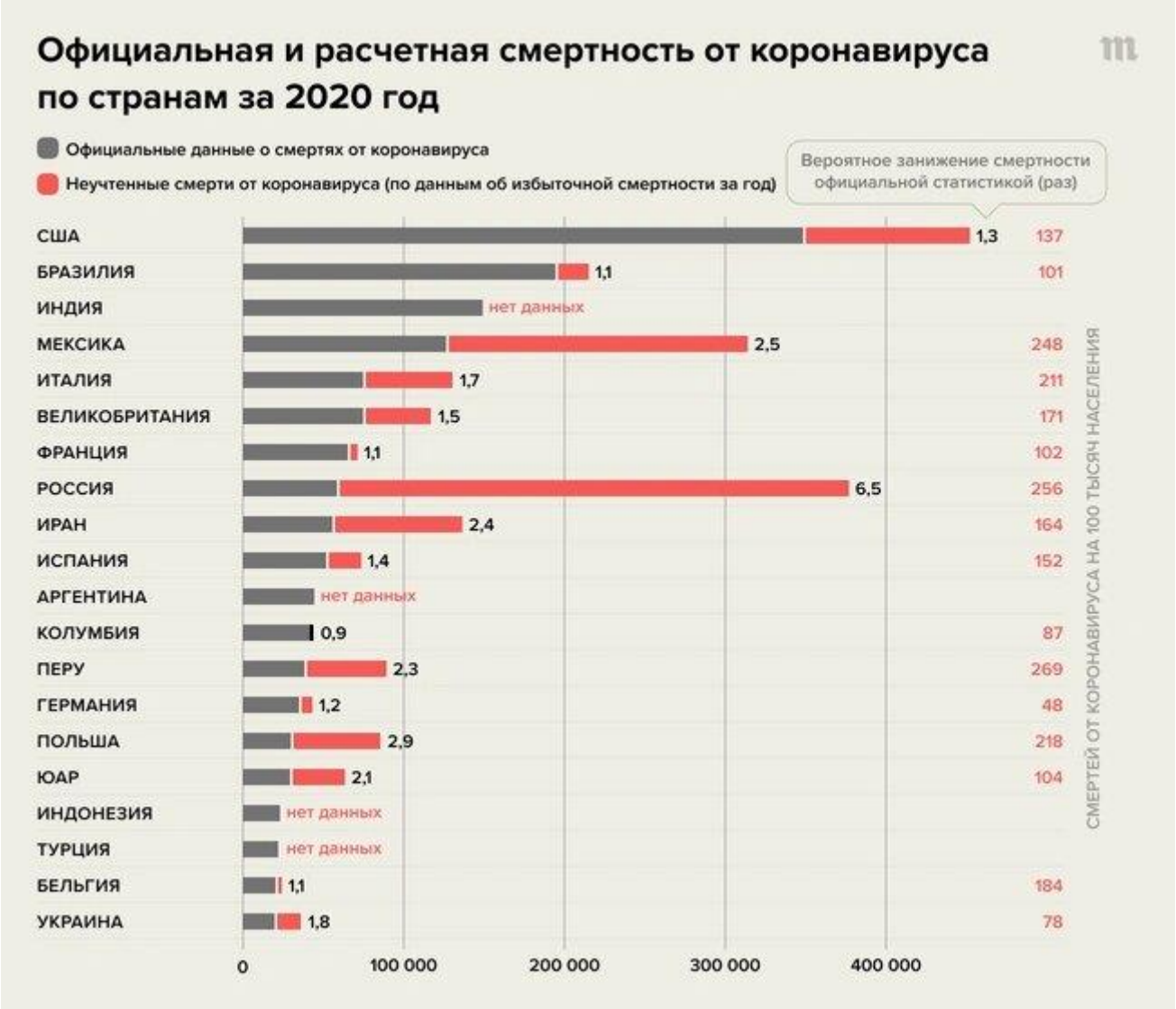
_____ _____
(підпис) (прізвище, ініціали)

Одеса – 2021

ЗМІСТ

ВСТУП

Наш час показав актуальність напрямку дослідження математичних моделей епідемії. Такі дослідження дозволяють нам промоделювати процес розповсюдження епідемії. Дослідити вплив різних факторів таких як карантин, вакцинація, вчасна терапія. В роботі були обрані для дослідження класичні моделі туберкульозу. Цей вибір об’ґрунтований схожістю та близькими симптомами із захворюванням COVID-19 – смертельні небезпечні інфекційні захворювання, які передаються аерогенним шляхом та уражає, переважно, легені. Смертність при нелікованому туберкульозі органів дихання складає близько 50%. У випадку COVID-19 статистика трішки краща, але, через відсутність вакцини у 2020 році, не менш лякаюча:



За оцінками Всесвітньої Організації Здравоохорони (ВОЗ) на 2017 рік збудниками туберкульозу було інфіковано близько 2,4 мільярдів людей, що становить близько треті населення Землі. У 2017 році туберкульозом захворіло 10 мільйонів людей та 1,8 мільйонів людей померло від цієї хвороби. Для порівняння, у 2005 році ці цифри були 8,8 мільйонів людей та 1,6 мільйонів смертей відповідно.

Проведений огляд літератури показав, що на даний час створена значна кількість математичних моделей, що описують як окремі процеси («природня» динаміка розповсюдження туберкульозу, виявлення хворих, лікування хворих), так й тих, що роблять спроби одночасно охопити усі значні аспекти розповсюдження та контролю туберкульозу.

Було вибрано одну з найновіших моделей – базова модель, що заснована на підході, що був запропонований С. Є. Борисовим та А. А. Романюхою, та порівняння її із класичною моделлю SLT.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Метод Рунге-Кутти

Метод Рунге — Кутти четвертого порядку розв’язування систем диференційних рівнянь.

Розглянемо наступну задачу:

$$\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \end{cases}$$

де $y, f, k_i \in \mathbb{R}$, $x, h \in \mathbb{R}^1$.

Тоді найближче значення в наступних точках обчислюється ітераційною формулою:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + k_2 + k_3 + k_4).$$

Обчислення нового значення проходить у чотири стадії:

$k_1 = f(x_n, y_n)$,

$k_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$,

$k_3 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$,

$k_4 = f(x_n + h, y_n + hk_3)$,

де h - величина кроку сітки по x. Похибка даного методу на одному кроці становить $O(h^5)$, а на кінцевому інтервалі - $O(h^4)$. Для систем з двох або більше рівнянь розв’язування методом Рунге-Кутти проводиться в аналогічний спосіб. Розглянемо наступну систему:

$$\begin{cases} X' = f(t, X, Y), \\ Y' = f(t, X, Y), \\ \dots \end{cases}$$

А її розв’язок має вигляд:

$$\begin{cases} X = X(t), \\ Y = Y(t), \\ \quad \dots \end{cases}$$

де t - незалежна змінна(наприклад, час); X, Y і т.д. - шукані функції (залежні від t змінні). Функції f, g і т.д. - задані. Також передбачаються заданими і початкові умови. Метод Рунге-Кутти для системи полягає у рекурентному застосуванні наступних формул:

$$X_{k+1} = X_k + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$Y_{k+1} = Y_k + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + 2m_4)$$

де k_i, m_i обчислюються за формулами:

$$k_1 = f(t_k, X_k, Y_k, \dots)\Delta t,$$
$$m_1 = g(t_k, X_k, Y_k, \dots)\Delta t, \dots,$$
$$k_2 = f\left(t_k + \frac{\Delta t}{2}, X_k + \frac{k_1}{2}, Y_k + \frac{m_1}{2}, \dots\right)\Delta t,$$
$$m_2 = g\left(t_k + \frac{\Delta t}{2}, X_k + \frac{k_1}{2}, Y_k + \frac{m_1}{2}, \dots\right)\Delta t, \dots,$$
$$k_3 = f\left(t_k + \frac{\Delta t}{2}, X_k + \frac{k_2}{2}, Y_k + \frac{m_2}{2}, \dots\right)\Delta t,$$
$$m_3 = g\left(t_k + \frac{\Delta t}{2}, X_k + \frac{k_2}{2}, Y_k + \frac{m_2}{2}, \dots\right)\Delta t, \dots,$$
$$k_4 = f(t_k + \Delta t, X_k + k_3, Y_k + m_3, \dots)\Delta t,$$
$$m_4 = g(t_k + \Delta t, X_k + k_3, Y_k + m_3, \dots)\Delta t, \dots,$$

1.2 Схема дослідження системи на стійкість

Розглянемо систему двох автономних звичайних диференційних рівнянь загального вигляду:

$$\begin{cases} \frac{dx}{dt} = P(x, y) \\ \frac{dy}{dt} = Q(x, y) \end{cases} \quad (1)$$

Розв'язком системи двох автономних звичайних диференційних рівнянь є будь-які дві функції $x(t), y(t)$, що задовільняють цій системі. Фазовою площиною називаєтеся площина з осями координат, на яких відкладено значення змінних x і y, кожна точка площини відповідає певному стану системи. Сукупність точок на фазовій площині, положення яких відповідає станам системі в процесі зміни в часі змінних $x(t), y(t)$ відповідно до рівнянь досліджуваної системі, називається фазовою траєкторією. Сукупність фазових траєкторій для різних початкових значень дає фазовий портрет системи. Побудова фазового портрета дозволяє зробити висновки про характер зміни змінних x і y, не знаючи аналітичних рішень вихідної системи рівнянь. Вираз для фазових траєкторій в аналітичному вигляді можна отримати, розділивши друге з рівнянь системи (1) на перше:

$$\frac{dy}{dx} = \frac{Q(x,y)}{P(x,y)} \quad (2)$$

Розв'язок цього рівняння $y = y(x, c)$, або в неявному вигляді $F(x, y) = c$, де c — стала інтегрування, дає сімейство інтегральних кривих рівняння (2) — фазових траєкторій системи (1) на площині OY.

1.2.1 Стаціонарний стан системи

Точка $(\bar{x}, \bar{y})$, в якій похідні за часом змінних x і y одночасно звертаються в нуль є особливою точкою (точкою покою).

$$\begin{cases} \left.\frac{dx}{dt}\right|_{x,y} = P(\bar{x}, \bar{y}) = 0 \\ \left.\frac{dy}{dt}\right|_{x,y} = Q(\bar{x}, \bar{y}) = 0 \end{cases} \Longrightarrow (\bar{x}, \bar{y})$$

Особлива точка диференційних рівнянь фазових траєкторій системи (2) відповідає стаціонарному стану системи (1).

1.2.2 Лінеаризація системи звичайних диференційних рівнянь

Розглянемо характер поведінки змінних при деякому невеликому відхиленні системи від стану рівноваги (x, y). Введемо нові змінні ξ, η, визначивши їх як зміщення, відносно рівноважних значень змінних:

$$x = x + \xi, y = y + \eta \tag{3}$$

Підставивши ці вирази в (1), отримаємо:

$$\begin{cases} \frac{d\bar{x}}{dt} + \frac{d\xi}{dt} = P(\bar{x} + \xi, \bar{y} + \eta) \\ \frac{d\bar{y}}{dt} + \frac{d\eta}{dt} = Q(\bar{x} + \xi, \bar{y} + \eta) \end{cases} \tag{4}$$

Так як $\bar{x}, \bar{y}$ — координати особливої точки, то $\frac{dx}{dt} = \frac{dy}{dt} = 0$, тоді система рівнянь відносно ξ и η буде мати вигляд:

$$\begin{cases} \frac{d\xi}{dt} = P(\bar{x} + \xi, \bar{y} + \eta) \\ \frac{d\eta}{dt} = Q(\bar{x} + \xi, \bar{y} + \eta) \end{cases} \tag{5}$$

Припустимо, що функції P і Q неперервні імають n-і похідні в точці $(\xi_{0,0}) = (0, 0)$. Тоді ми можемо розкласти праві частини рівнянь (5) в ряд Тейлора по змінних ξ, η.

$$\begin{cases} \frac{d\xi}{dt} = P(\bar{x}, \bar{y}) + a\xi + b\eta + (p_1 1\xi^2 + 2p_1 2\xi\eta + p_2 2\eta^2 + \cdots) \\ \frac{d\eta}{dt} = Q(\bar{x}, \bar{y}) + c\xi + d\eta + (q_1 1\xi^2 + 2p_1 2\xi\eta + p_2 2\eta^2 + \cdots) \end{cases} \tag{6}$$

де

$$a = P'_\xi(\bar{x}, \bar{y}), b = P'_\eta(\bar{x}, \bar{y}), c = Q'_\xi(\bar{x}, \bar{y}), d = Q'_\eta(\bar{x}, \bar{y}). \tag{7}$$

Врахуємо, що за визначенням особливої точки

$$P(\bar{x}, \bar{y}) = 0$$
$$Q(\bar{x}, \bar{y}) = 0$$

і відкинемо в рівняннях нелінійні члени (6). Отримаємо систему лінійних рівнянь з постійними коефіцієнтами, які називаються лінеаризованою системою або системою першого наближення:

$$\frac{d\xi}{dt} = a\xi + b\eta$$
$$\frac{d\eta}{dt} = c\xi + d\eta$$

1.2.3 Характеристичне рівняння

Лінійній системі двох диференційних рівнянь (8) можна поставити у відповідність детермінант:

$$\begin{vmatrix} (a-\lambda) & b \\ c & (d-\lambda) \end{vmatrix} \tag{9}$$

Розкриваючи визначник (9), отримаємо характеристичне рівняння:

$$\lambda^2 - (a+d)\lambda + (ad - bc) = 0 \tag{10}$$

Корені рівняння (10) називаються власними числами визначника (9) або показниками Ляпунова:

$$\lambda_{1,2} = \frac{(a+d) \pm \sqrt{(a+d)^2 - 4(ad-bc)}}{2} \tag{11}$$

Загальний розв’язок системи (8) можна представити у вигляді:

$$\xi(t) = C_{11}e^{\lambda_1 t} + C_{12}e^{\lambda_2 t}, \eta(t) = C_{21}e^{\lambda_1 t} + C_{22}e^{\lambda_2 t}$$

Якщо обидва корені (11) мають від’ємну дійсну частину, і, отже, усі рухи рівнянь першого наближення (8) загасають, то стан рівноваги стійкий. Якщо навіть один корінь має додатну дійсну частину, то система (8) має зростаючі розв’язки, тобто стан рівноваги нестійкий.

1.2.4 Біфуркаційна діаграма і типи фазових портретів

Отже, характеристичні числа можуть бути дійсними або комплексносполученими. Введемо позначення:

$$\sigma = (a + d);\ \Delta = \begin{vmatrix} a & b \\ c & d \end{vmatrix} \tag{12}$$

Тоді характеристичне рівняння запишеться у вигляді (10):

$$\lambda^2 - \sigma\lambda + \Delta = 0 \tag{13}$$

Розглянемо площину $\sigma(\Delta)$ і визначимо на ній області, відповідні томі чи іншому типу стану рівноваги, який визначається характером коренів характеристичного рівняння:

$$\lambda_{1,2} = \frac{\sigma \pm \sqrt{\sigma^2 - 4\Delta}}{2}$$

Виділимо наступні випадки (див. малюнок нижче):

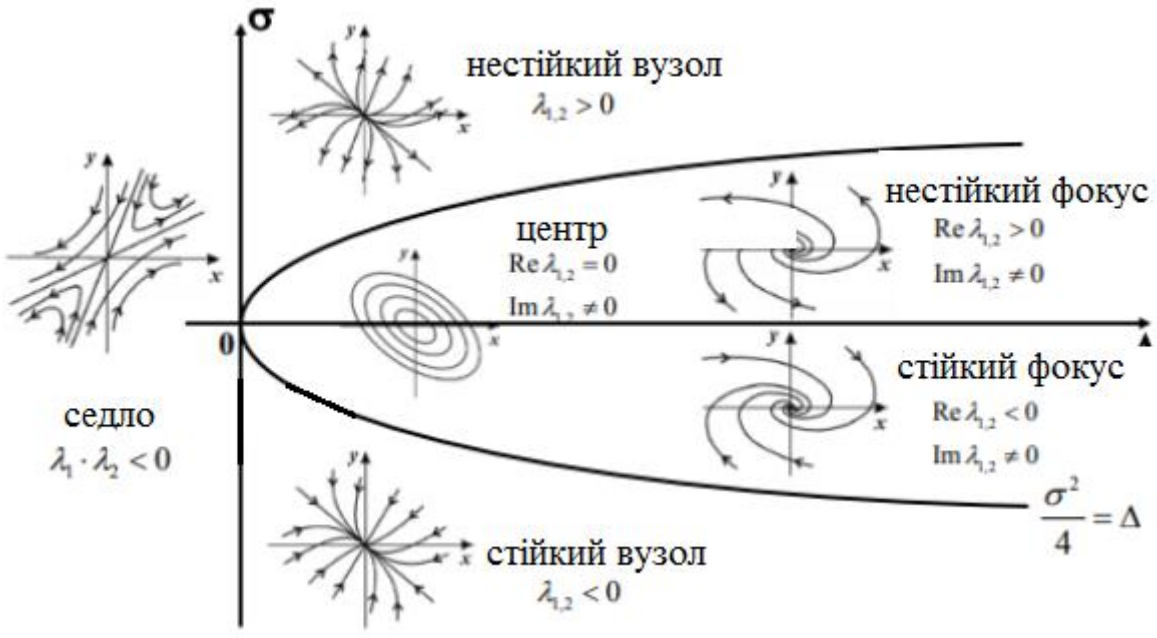


Рис. 1: Біфуркаційна діаграма для системи лінійних рівнянь (8)

1) λ_1 и λ_2 — дійсні і різних знаків, $\lambda_1 \cdot \lambda_2 < 0$, особлива точка — сідло.	Область на площині $\sigma(\triangle)$ визначається нерівністю $\triangle < 0$.
2) λ_1 и λ_2 — дійсні, $\lambda_{1,2} > 0$, особлива точка — нестійкий вузол.	Область на площині $\sigma(\triangle)$ визначається системою нерівностей $\begin{cases} \triangle > 0, \\ \sigma > 0, \\ \sigma^2 - 4\triangle > 0. \end{cases}$
3) λ_1 и λ_2 — дійсні, $\lambda_{1,2} < 0$, особлива точка — стійкий вузол.	Область на площині $\sigma(\triangle)$ визначається системою нерівностей $\begin{cases} \triangle > 0, \\ \sigma < 0, \\ \sigma^2 - 4\triangle > 0. \end{cases}$
4) λ_1 и λ_2 — комплексні, $Re\lambda_{1,2} > 0 >$, особлива точка — нестійкий фокус.	Область на площині $\sigma(\triangle)$ визначається системою нерівностей $\begin{cases} \triangle > 0, \\ \sigma > 0, \\ \sigma^2 - 4\triangle < 0. \end{cases}$

5) λ_1 і λ_2 — комплексні, $Re\ \lambda_{1,2} < 0$, особлива точка — стійкий фокус.	Область на площині $\sigma(\triangle)$ визначається системою нерівностей $\begin{cases} \triangle > 0, \\ \sigma < 0, \\ \sigma^2 - 4\triangle < 0. \end{cases}$
6) λ_1 и λ_2 — комплексні, $Re\ \lambda_{1,2} = 0$, $Im\ \lambda_{1,2} \neq 0$, особлива точка — центр.	Область на площині $\sigma(\triangle)$ визначається системою нерівностей $\begin{cases} \triangle > 0, \\ \sigma = 0. \end{cases}$

Таблиця 1.1: Опис станів рівноваги за допомогою систем нерівностей

Варіанти 1) – 5) описують так звані грубі стани рівноваги. Якщо дійсні частини одного або обох коренів характеристичного рівняння дорівнюють нулю, то для грубого стану рівноваги рівняння (8) не дають відповіді на питання про асимптотичну стійкість за Ляпуновим, і необхідно розглядати члени більш високого порядку малості в розкладенні в ряд Тейлора правих частин рівнянь (6). У підсумку ми отримаємо діаграму розбиття площини параметрів $\sigma\Delta$ на області, відповідні різним типам стану рівноваги (рис. 4.2). Нагадаємо, що коефіцієнти лінійної функції системи a, b, c, d обчислювалися як відповідні часткові похідні правих частин вихідних нелінійних рівнянь і є комбінаціями параметрів якоїсь похідної моделі. При зміні будь-якого параметра вихідної нелінійної моделі будуть змінюватися і коефіцієнти відповідної лінійної системі, отже будуть змінюватися величини $\sigma,\ \delta$. При переході через осі координат характер фазового портрету якісно змінюється. Тому такі межі називаються біфуркаційними — по різні боки від межі система має два якісно різні портрети і, відповідно, дві різних типів поведінки. Потрібно визначити, що біфуркаційними є тільки ті переходи через осі координат, під час яких нестійкий стан змінюється стійким або навпаки.

Оскільки неможливо поступовою неперервною зміною фазового портрету перейти, наприклад, від вузла до сідла або від нестійкого фокусу до стійкого. Переходи стійкий вузол – стійкий фокус (нестійкий вузол – нестійкий фокус) не є біфуркаційними, оскільки можна поступовими неперервними змінами фазового портрету перейти від вузла до фокусу.

1.2.5 Рівняння сепаратрис

Важливим елементом фазового портрету особливої точки типу «сідло» є дві сепаратриси. Ці прямі завжди спрямовані уздовж власних векторів матриці коефіцієнтів лінійних рівнянь системи:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Нагадаємо, що власним вектором матриці M, відповідним власному числу λ, називається будь-який відмінний від нуля вектор $\bar{x}$, який задовольняє рівнянню $M\bar{x} = \lambda\bar{x}$. Отже, прямі-сепаратриси задаються рівнянням

$$1)\ (a - \lambda_{1,2})x + by = 0$$

або

$$2)\ cx + (d - \lambda_{1,2})y = 0$$

де $\lambda_{1,2}$ — характеристичні числа матриці коефіцієнтів системи. Одному значенню λ відповідає одна пряма. Якщо у виразах 1) і 2) коефіцієнтів при x и y не рівні нулю, то вирази 1) і 2) співпадають и задають одну й ту саму пряму.

1.2.6 Метод ізоклін

Для побудови фазового портрету користуються методом ізоклін: на фазовій площині наносять лінії, які перетинають дотичні до інтегральних кривих під певним кутом. Рівняння ізокліни виводиться з співвідношення:

$$\frac{dy}{dx} = \frac{Q(x,y)}{P(x,y)} = A = const \qquad (15)$$

В рівнянні (15) константа A є тангенсом кута нахилу $A = tg\varphi$ дотичної до фазової траєкторії. Через головні ізокліни (нуль-ізокліни) дотичні до фазових траєкторій проходять під кутом $\varphi = 0°$ (ізокліна горизонтальних дотичних) и $\varphi = 90°$ (ізокліна вертикальних дотичних). Для ізокліни горизонтальних дотичних рівняння (15) набуває вигляду:

$$\frac{dy}{dx} = \frac{Q(x,y)}{P(x,y)} = tg0° = 0$$

Для ізокліни вертикальних дотичних:

$$\frac{dy}{dx} = \frac{Q(x,y)}{P(x,y)} = tg90° = \infty \ \textit{або}\ P(x,y) = 0$$

РОЗДІЛ II
ЯКІСНЕ ДОСЛІДЖЕННЯ МОДЕЛЕЙ

2.1 Модель SLT

Дана модель являє собою опис природньої динаміки епідемії, де уся популяція ділиться на чутливих (S), носіїв латентної інфекції (L) та інфекційних індивідів з активним туберкульозом (T). Динаміка цих груп задається системою диференційних рівнянь:

$$\frac{dS}{dt} = \Pi - \lambda S - \mu S,$$
$$\frac{dL}{dt} = (1-p)\lambda S - \delta L - \mu L,$$
$$\frac{dT}{dt} = \delta L + p\lambda S - (\mu + \mu_T)T,$$
$$\lambda(t) = \beta T(t).$$

Величина	Опис
$S(t)$	кількість чутливих у момент часу t
$L(t)$	кількість інфікованих у момент часу t
$T(t)$	кількість заразних хворих у момент часу t
$\lambda(t)$	сила інфекції на душу населення у момент часу t
Π	приток молодді у модельну популяцію
μ	середня смертність від причин, що не пов'язані з туберкульозом
p	ймовірність швидкого прогресування хвороби
δ	константа швидкості реактивації туберкульозної інфекції
μ_T	додаткова смертність, що була викликана активним туберкульозом
β	"коефіцієнт передачі" туберкульозної інфекції
f_F	ймовірність розвитку інфекційної форми хвороби при швидкому прогресуванні
f_S	ймовірність розвитку інфекційної форми хвороби при ендогенній активації
2ρ	коефіцієнт швидкості розвитку рецидиву туберкульозу
α	коефіцієнт швидкості спонтанного самозцілення

Таблиця 2.1: Опис змінних моделі SLT

Знайдемо початкове розв’язання данної моделі, для цього прирівняємо праві частини нулю:

$$\begin{cases} \Pi - \beta TS - \mu S = 0 \\ (1-p)\beta TS - \delta L - \mu L = 0 \\ \delta L + p\beta TS - (\mu + \mu_T)T = 0 \end{cases}$$

Отримуємо вектор рішень $X_0 = \{\frac{\Pi}{\mu}, 0,0\}$.

Для дослідження на стійкість знайдемо власні значення:

$$\begin{vmatrix} -(\beta T + \mu) & (1-p)\beta T & p\beta T \\ 0 & -(\delta + \mu) & \delta \\ -\beta S & (1-p)\beta S & -(\mu + \mu_T) + p\beta S \end{vmatrix}$$

З урахуванням X_0 отримуємо:

$$\begin{vmatrix} -\mu - \lambda & 0 & 0 \\ 0 & -(\delta + \mu) - \lambda & \delta \\ -\frac{\beta\Pi}{\mu} & \frac{(1-p)\beta\Pi}{\mu} & -(\mu + \mu_T) + \frac{p\beta\Pi}{\mu} - \lambda \end{vmatrix} =>, \begin{cases} \lambda_1 = -\mu \\ \lambda_2 = -(\delta + \mu) \\ \lambda_3 = -(\mu + \mu_T) + \frac{p\beta\Pi}{\mu} \end{cases}$$

Так як усі розв’язки невід’ємні, дане рішення та система є стійкими.

2.2 Базова модель запропонована Борисовим та Романюхою

Математична модель розповсюдження та контролю туберкульозу запропонована С. Є. Борисовим та А. А. Романюхой. Вперше модель юуда опублікова у 2004 році [2]. Оскільки вона орієнтована на моделювання епідеміології туберкульозу у СНГ, у її основу було покладено чотириключові особливості, що відображають схеми противотуберкульозних заходів, що застосовуються на території СНГ:

- розрізняють БК- та БК+ форми туберкульозу, при чому БК- вважається ранньою стадією хвороби, що передує розвитку тяжких БК+ стадій.
- Виліковані та самовилікові від туберкульозу прирівнюються до носіїв латентної інфекції.
- Хворі розділяються на виявлених та невиявлених.
- Переважна частина населення вважається вакцинованою БЦЖ.

Перевагами такого підходу є:

- Можливість розділенняу рамках моделі раннього та пізнього виявлення (виявлення БК- та БК+ хворих).
- Облік відмінностей виявлених та невиявлених (а, відповідно, лікуємих та нелікуємих) хворих.
- Розгляд епідемічних процесів лише серед дорослого населення, що знімає багато проблем із неоднорідністю властивостей хазяїв інфекції.

В основі розглядаємої математичної моделі полягає розділення популяції, що моделюється, на 6 груп, чисельність яких залежить від часу t та є змінними моделі:

- $S(t)$ – неінфіковані збудниками туберкульозу
- $L(t)$ – інфіковані індивіди (носії латентної інфекції)
- $T_n^U(t)$ – невиявлені хворі без бактеріовиділення (БК-)
- $T_i^U(t)$ – невиявлені хворі із бактеріовиділення (БК+)
- $T_n^T(t)$ – виявлені хворі без бактеріовиділення (БК-)
- $T_i^T(t)$ – виявлені хворі із бактеріовиділення (БК+)

Всередині кожної групи усі індивіди рахуються ідентичними за своїми властивостями. У силу масової вакцинації БЦЖ населення на території країн колишнього СРСР, що призводить до зниження дитячої захворюваності туберкульозом до пренебрижимо малого рівня, у модеолюючу популяцію були враховані індивіди старше 15 років. А молодь, що досягла 15 років може поповнити групи S та L, але не групу хворих T. У якості хворроби розглядається лише туберкульоз органів дихання, тому що внелегеневий туберкульоз значно більш рідкий та не привносить вагомого вкладу у подальше розповсюдження інфекції.

Неінфіковані індивіди S вважаються схильними до інфікування, що призводить або до швидкого прогресування хвороби, або до переводу у групу латентних носіїв L. У інфікованих індивідів активний туберкульоз може розвиватися як у результаті ендогенної активації (із постійною питомою швидкістю), так й через екзогенне суперінфікування (питома швидкість пропорційна силі інфекції). Усі захворівши індивіди включаються до групи T_n^U, тобто враховується, що будь-яка туберкульозна хвороба починає із форм без бактеріовиділення та поза «зоною видимості» противотуберкульозних установ. Формальним критерієм подальшого прогресування захворювання є розвиток бактеріовиділення (а, відповідно, й перехід до групи T_i^U). Передбачається, що це відбувається із постійною питомою швидкістю. Процеси спонтанного припинення бактеріовиділення (перехід із T_i^U до T_n^U) та спонтанного самовилікування

(перехід із T_n^U до L) також вважаються такими, що відбуваються із постійними питомими швидкостями. Індивіди з груп S та L схильні лише до середньопопуляційному ризику смерті від нетуберкульозних причин. Хворі з груп T_n^U и T_i^U крім среднепопуляціонного ризику смерті від нетуберкульозних причин схильні до додаткового ризику смерті, пов'язаного з активним туберкульозом (у групі T_i^U він вище, у T_n^U – нижче). Хворі з груп T_n^U та T_i^U можуть бути виявлені протитуберкульозними закладами та спрямовані на лікування, що моделюється за допомогою переказу виявлених хворих в групи T_n^T та T_i^T відповідно. Процеси розвитку і припинення бактеріовиділення, а також клінічного лікування (що переводить індивідів з T_n^T до L) визначаються постійними питомими швидкостями, але їх значення відрізняються від значень відповідних швидкостей для невиявлених (а значить, і не одержують лікування) хворих. Виявлені (отримують лікування) хворі схильні як среднепопуляціонному ризику смерті від нетуберкульозних причин, так і до додаткового ризику смерті від причин, пов'язаних з туберкульозом (нижчу, ніж відповідний ризик для невиявлених хворих). Вид залежності сили інфекції від змінних моделі (численностей груп) може варіювати, але важливим припущенням є те, що виявлені бактеріовиделітелі (T_i^T) являють собою набагато меншу епідеміологічну небезпеку, ніж невиявлені бактеріовиделітелі (T_i^U). Це пов'язано з тим, що отримує ефективному лікування бактеріовиділювач різко знижує або повністю припиняє бактеріовиділення через кілька тижнів або місяців терапії, хоча з точки зору формального обліку він продовжує вважатися бактеріовидільником порядку року. Крім того в моделі закладена можливість обліку міграційного відтоку і припливу індивідів з і в усі групи. Таким чином, чисельність груп моделі підпорядковується системі нелінійних звичайних диференціальних рівнянь (2.1) - (2.6), принцип побудови яких буде описаний далі. Параметри, які використовуються в рівняннях (2.1) - (2.6) перераховані в табл. 2.1, блок-схеми моделі наведені на рис. 2.1 і 2.2

$$\frac{dS}{dt} = f_S(t) - (\lambda(t, S, \ldots, T_i^T) + \mu + e)S, \tag{2.1}$$

$$\frac{dL}{dt} = f_L(t, S, \ldots, T_i^T) + (1-p)\lambda(t, S, \ldots, T_i^T)S + \gamma_L T_n^U + \gamma_{L0} T_n^T - \\ - (\delta + p\lambda(t, S, \ldots, T_i^T) + \mu + e)L, \tag{2.2}$$

$$\frac{dT_n^U}{dt} = f_{nU}(t) + p\lambda(t, S, \ldots, T_i^T)S + (\delta + p\lambda(t, S, \ldots, T_i^T))L + \gamma_n T_i^U - \\ - (\varphi_n + \gamma_L + \delta_i + \mu_{nU} + e_{nU})T_n^U, \tag{2.3}$$

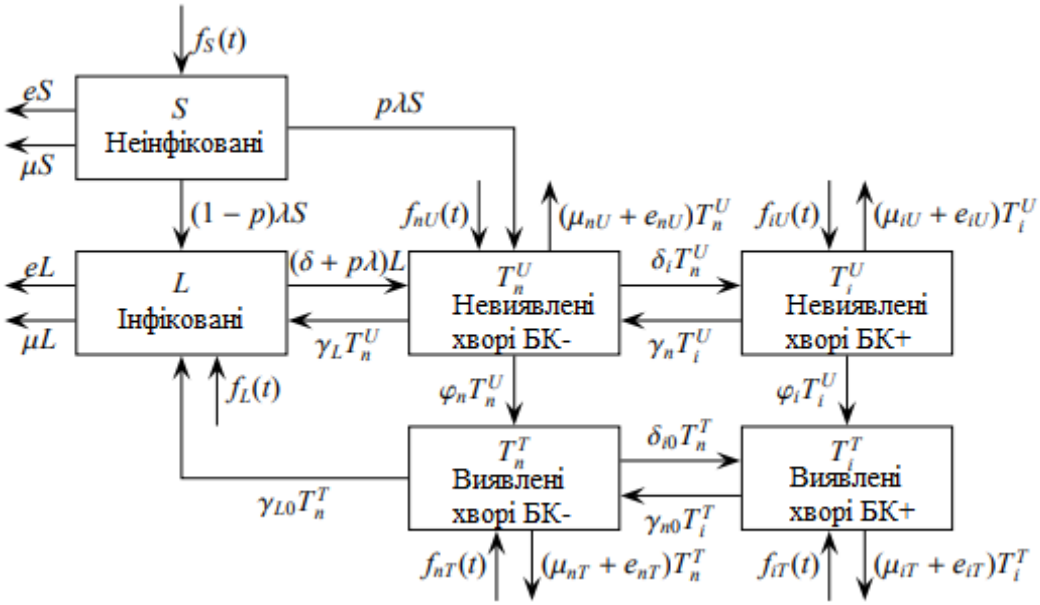
$$\frac{dT_i^U}{dt} = f_{iU}(t) + \delta_i T_n^U - (\varphi_i + \gamma_n + \mu_{iU} + e_{iU})T_i^U, \tag{2.4}$$

$$\frac{dT_n^T}{dt} = f_{nT}(t) + \gamma_{n0} T_i^T + \varphi_n T_n^U - (\gamma_{L0} + \delta_{i0} + \mu_{nT} + e_{nT})T_n^T, \tag{2.5}$$

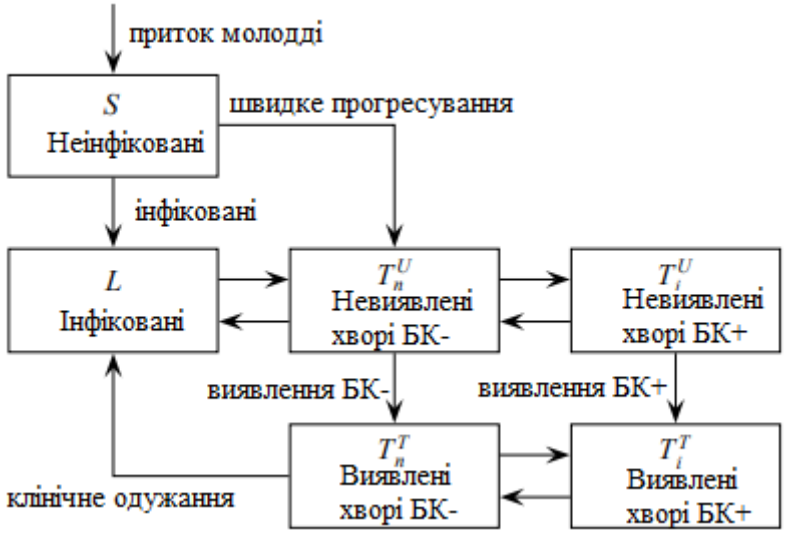
$$\frac{dT_i^T}{dt} = f_{iT}(t) + \delta_{i0} T_n^T + \varphi_i T_i^U - (\gamma_{n0} + \mu_{iT} + e_{iT})T_i^T. \tag{2.6}$$

Символ	Опис
$f_i(t, \ldots)$	швидкість міграційного притоку у кожну з груп fs(t) та fL(t) включає у себе приток молодді
λ	сила інфекції
μ	середньопопуляційна смертність для груп S та L
μ_{nU}	смертність у групі невиявлених БК-
μ_{iU}	смертність у групі невиявлених БК+
μ_{nT}	смертність у групі виявлених БК-
μ_{iT}	смертність у групі виявлених БК+
e	коефіцієнт швидкості міграційного відтоку з груп S та L
e_i	коефіцієнт швидкості міграційного відтоку з груп хворих
p	ймовірність швидкого прогресування хвороби
δ	коефіцієнт швидкості ендогенної активації
δ_i	коефіцієнт швидкості розвитку інфекційних форм хвороби
γ_L	коефіцієнт швидкості спонтанного самозцілення
γ_n	коефіцієнт швидкості спонтанного припинення бактеріовиділення
φ_i	коефіцієнт швидкості виявлення хворих із бактеріовиділенням
φ_n	коефіцієнт швидкості виявлення хворих без бактеріовиділення
δ_{i0}	коефіцієнт швидкості розвитку інфекцій. форм захвор-я. на фоні лікування
γ_{L0}	коефіцієнт швидкості клінічного одужання
γ_{n0}	коефіцієнт швидкості припинення бактеріовиділення на фоні лікування

Таблиця 2.2: Опис змінних базової моделі



Таблиця 2.3: Блок-схема моделі (2.1)-(2.6)



Таблиця 2.4: Спрощена схема моделі (2.1)-(2.6). Потоки смертності та міграції не показані

Рівняння базової моделі будуються на основі принципу збереження кількості індивідів. Повна чисельність моделюється популяції може змінюватися тільки за рахунок процесів, природним чином впливають на її чисельність (приплив молоді (народження), смерть, міграційні приплив і відтік). Всі інші процеси (по в'язані з інфікуванням, розвитком, прогресом або лікуванням від туберкульозу, а також виявлення хворих) лише переміщають індивідів з однієї групи в іншу, зберігаючи повну чисельність моделюється популяції.

Чисельність групи неінфікованих індивідів *S* підпорядковується рівнянню

$$\frac{dS}{dt} = f_S(t) - \left(\lambda\left(t, S, \ldots, T_i^T\right) + \mu + e\right)S,$$

у якому функція $f_S(t)$ задає приплив неінфікованих індивідів в популяцію (як в результаті припливу молоді, що досягає 15-річного віку, так і за рахунок міграції), а член $-(\lambda(t, S, \ldots, T_i^T) + \mu + e)S$ визначає зменшення чисельності цієї групи за рахунок інфікування $-(\lambda(t, S, \ldots, T_i^T) + \mu + e)S$, смерті від нетуберкульозних причин $(-\mu S)$ і міграційного відтоку $(-eS)$.

Чисельність групи носіїв латентної інфекції *L* підпорядковується рівнянню

$$\frac{dL}{dt} = f_L(t)\left(t, S, \ldots, T_i^T\right) + (1-p)\lambda\left(t, S, \ldots, T_i^T\right)S + \gamma_L T_n^U + \gamma_{L0} T_n^T -$$

$$- \left(\delta + p\lambda\left(t, S, \ldots, T_i^T\right) + \mu + e\right)L,$$

де $f_L(t)\left(t, S, \ldots, T_i^T\right)$ визначає приплив інфікованих індивідів (інфікованої молоді, що досягає 15-річного віку і інфікованих мігрантів) $(1 - p)\lambda(t, S, \ldots, T_i^T)S$ описує приплив зазнали первинного інфікування, але не розвинули активної хвороби незабаром після цього, $\gamma_L T_n^U$ задає приплив спонтанно самовилікуватися від туберкульозу, а $\gamma_{L0} T_n^T$– вилікувалися з числа що знаходяться на лікуванні, член $-(\delta + p\lambda\left(t, S, \ldots, T_i^T\right))L$ визначає зменшення кількості носіїв латентної інфекції в результаті ендогенної активації $-\delta L$ і екзогенного суперінфікування $-(\delta + p\lambda\left(t, S, \ldots, T_i^T\right))L$, а члени $-\mu L$ та -

eL – в результаті смерті від нетуберкульозних причин і міграційного відтоку. Відзначимо, що ймовірність розвитку хвороби в результаті суперінфікування вважається рівної ймовірності швидкого прогресування туберкульозу після первинного інфікування.

Чисельність групи невиявлених хворих без бактеріовиділення T_n^U підпорядковується рівнянню

$$\frac{dT_n^U}{dt} = f_{nU}(t) + p\lambda\left(t, S, \ldots, T_i^T\right)S + \left(\delta + p\lambda\left(t, S, \ldots, T_i^T\right)\right)L +$$

$$+\gamma_n T_i^U - (\varphi_n + \gamma_L + \delta_i + \mu_{nU} + e_{nU})T_n^U,$$

в якому за допомогою функції $f_{nU}(t)$ моделюється приплив невиявлених БК- хворих ззовні популяції, вираз $p\lambda\left(t, S, \ldots, T_i^T\right)S + p\lambda(t, S, \ldots, T_i^T)L$ задає приплив індивідів, у яких розвинувся активний туберкульоз (в результаті швидкого прогресування, ендогенної активації або екзогенного суперінфікування), $\gamma_n T_i^U$ визначає приплив спонтанно припинили бактеріовиділення невиявлених хворих і член $-(\varphi_n + \gamma_L + \delta_i + \mu_{nU} + e_{nU})T_n^U$ моделює спадання чисельності групи T_n^U в результаті виявлення $(-\varphi_n T_n^U)$, спонтанного самозцілення $(-\gamma_L T_n^U)$, прогресування хвороби і розвитку бактеріовиділення $(-\delta_i T_n^U)$, смерті $(-\mu_{nU} T_n^U)$ та міграційного відтоку $(-e_{nU} T_n^U)$.

Чисельність групи невиявленх хворих із бактеріовиділенням T_i^U підпорядковується рівнянню

$$\frac{dT_i^U}{dt} = f_{iU}(t) + \delta_i T_n^U - (\varphi_i + \gamma_n + \mu_{iU} + e_{iU})T_i^U,$$

де $f_{iU}(t)$ задає приплив невиявлених БК + хворих ззовні популяції, $\delta_i T_n^U$ визначає приплив за рахунок розвитку бактеріовиділення у невиявлених БК хворих (T_n^U), а за допомогою виразу $-(\varphi_i + \gamma_n + \mu_{iU} + e_{iU})T_i^U$ моделюється зменшення кількості невиявлених БК + хворих в результаті виявлення

$(-\varphi_i T_i^U)$, спонтанного припинення бактеріовиділення $(-\gamma_n T_i^U)$, смерті $(-\mu_{iU} T_i^U)$ і міграційного відтоку $(-e_{iU} T_i^U)$.

Чисельність групи виявлених хворих без бактеріовиділення $\boldsymbol{T_n^T}$ підпорядковується рівнянню

$$\frac{dT_n^T}{dt} = f_{nT}(t) + \gamma_{n0} T_i^T + \varphi_n T_n^U - (\gamma_{L0} + \delta_{i0} + \mu_{nT} + e_{nT}) T_n^T,$$

у якому $f_{nT}(t)$ визначає приплив виявлених БК-хворих ззовні популяції, $\gamma_{n0} T_i^T$ задає приплив виявлених хворих індивідів, які припинили бактеріовиділення (спонтанно або в результаті лікування), $\varphi_n T_n^U$ – потік виявлення БК-хворих, а член $-(\gamma_{L0} + \delta_{i0} + \mu_{nT} + e_{nT}) T_n^T$ визначає зменшення чисельності групи T_n^T за рахунок лікування $(-\gamma_{L0} T_n^T)$, прогресування хвороби і розвитку бактеріовиділення $(-\delta_{i0} T_n^T)$, смерті $(-\mu_{nT} T_n^T)$ і міграційного відтоку $(-e_{nT} T_n^T)$.

Чисельність групи виявлених хворих з бактеріовиділенням $\boldsymbol{T_i^T}$ підпорядковується рівнянню

$$\frac{dT_i^T}{dt} = f_{iT}(t) + \delta_{i0} T_n^T + \varphi_i T_i^U - (\gamma_{n0} + \mu_{iT} + e_{iT}) T_i^T,$$

де $f_{iT}(t)$ задає приплив виявлених БК + хворих ззовні популяції, $\delta_{i0} T_n^T$ визначає приплив за рахунок розвитку бактеріовиділення у виявлених БК хворих (T_n^T), $\varphi_i T_i^U$ – потік виявлення БК + хворих, а вираз $-(\gamma_{n0} + \mu_{iT} + e_{iT}) T_i^T$ задає зменшення кількості виявлених БК + хворих за рахунок припинення бактеріовиділення $(-\gamma_{n0} T_i^T)$, смерті $(-\mu_{iT} T_i^T)$ і міграційного відтоку $(-e_{iT} T_i^T)$. Початкові умови і всі коефіцієнти, використовувані в рівняннях (2.1)-(2.6), є невід'ємними величинами; функції $\lambda\left(t, S, \ldots, T_i^T\right)$ і $f_i(t), i = \{S, L, nU, iU, nT, iT\}$ також є невід'ємними і досить гладкими.

ВИСНОВКИ

В роботі були проведені дослідження математичних моделей – класичної SLT моделі та базової моделі розповсюдження туберкульозу, що була запропонована С. Є. Борисовим та А. А. Романюхой. При чисельних дослідженнях були построєні рішення відповідних математичних моделей, при якісних – були знайдені стаціонарні розв’язки моделей та досліджено їх на стійкість.

СПИСОК ЛІТЕРАТУРИ

1. Братусь А.С., Новожилов А.С., Платонов А.П. — Динамічні системи та моделі біології // М.: МГУ, 2011. — 436 с.

2. Ібрагимов Н. Х. — Практичний курс диференційних рівнянь та математичного моделювання // Нижній Новгород: Вид-во Нижегородського держуніверситета, 2007. — 421 с.

3. Колесін И.Д., Житкова Е.М. — Математичні моделі епідемій // СПб.: НИИФ СПбГУ, 2004. — 92 с.

4. Романюха А. А. — Математичні моделі в імунології та епідеміології інфекційних захворювань / ред.: Г.И. Марчук, А.А. Романюха .— 2-е вид. // М. : Лабораторія знань, 2015 .— 296 с.

5. Philip Munz, Ioan Hudea, Joe Imad, Robert J. Smith — When zombies attack!: mathematical modelling of an outbreak of zombie infection //Nova Science Publishers, Inc., 2009. — 150 с.

6. Аль-Азазі А. А. М. — Алгоритми та детермінована компартментна модель в управленні розвитком епідемічного процесу // Тверь: Тверський державний технічний університет, 2016. — 171 с.

7. Дисертація на здобуття наукового ступеня к. ф.-м. н. Авілова К. К. https://www.inm.ras.ru/wp-content/uploads/dis-sovet/disser/avilov_disser.pdf

8. http://www.apmath.spbu.ru/ru/staff/nogin/tu/part1-1.pdf

9. https://www.science-education.ru/ru/article/view?id=5196

Додаток А

Машинне дослідження на стійкість системи.

```
from scipy.integrate import odeint

import matplotlib.pyplot as plt
import numpy as np
import math


h1 = 2 #-b * S*
h2 = 0.8 #(lambda + mu + epsilon) * S*

h3 = 0.5 #f_L
h4 = 10 #(lambda - p*lambda) * S*
h5 = 8 #gama_L * T_n_U*
h6 = 2 #gama_L0 * T_n_T*
h7 = 0.12 #(delta + p*lambda + mu + epsilon) * L*

h8 = 0.8 #f_n_U
h9 = 0 #p*lambda * S*
h10 = 10000 #(delta + p*lambda) * L*
h11 = 10000 #gama_n * t_i_U*
h12 = 0.17 #(phi_n + gama_L + delta_i + mu_n_U + epsilon_n_U) * T_n_U*

h13 = 0.17 #f_i_U
h14 = 0.5 #delta_i * T_n_U*
h15 = 0.17 #(phi_i + gama_n + mu_i_U + epsilon_i_U) * T_i_U*


h16 = 10 #f_n_T
h17 = 0.12 #gama_n0 * T_i_T*
h18 = 8 #phi_n * T_n_U*
h19 = 0.17 #(gama_L0 + delta_i0 + mu_n_T + epsilon_n_T) * T_n_T*

h20 = 10 #f_i_T
h21 = 0.12 #delta_i0 * T_n_T*
h22 = 8 #phi_i * T_i_U*
h23 = 0.17 #(gama_n0 + mu_i_T + epsilon_i_T) * T_i_T*



def S(s):
    return h1 - h2*s

def L(s, t_n_U, t_n_T, l):
    return h3 + h4*s + h5*t_n_U + h6*t_n_T - h7*l

def T_n_U(s, l, t_i_U, t_n_U):
    return h8 + h9*s + h10*l + h11*t_i_U - h12*t_n_U
```

```
def T_i_U(t_n_U, t_i_U):
    return h13 + h14*t_n_U - h15*t_i_U

def T_n_T(t_i_T, t_n_U, t_n_T):
    return h16 + h17*t_i_T + h18*t_n_U - h19*t_n_T

def T_i_T(t_n_T, t_i_U, t_i_T):
    return h20 + h21*t_n_T + h22*t_i_U - h23*t_i_T

def rungeKutta(y0, time0, time, step):
    s, l, t_n_U, t_i_U, t_n_T, t_i_T = y0
    dotsS = []
    dotsL = []
    dotsT_n_U = []
    dotsT_i_U = []
    dotsT_n_T = []
    dotsT_i_T = []
    dotsS.append(s)
    dotsL.append(l)
    dotsT_n_U.append(t_n_U)
    dotsT_i_U.append(t_i_U)
    dotsT_n_T.append(t_n_T)
    dotsT_i_T.append(t_i_T)

    ready = 0
    print('Runge:')
    for t in np.arange(time0, time, step):
        readyTmp = int(t / time * 100)
        if(ready != readyTmp):
            ready = readyTmp
            print(ready, '%')
        koefS = []
        koefL = []
        koefT_n_U = []
        koefT_i_U = []
        koefT_n_T = []
        koefT_i_T = []


        #first koef[0]
        koefS.append(S(s) * step)
        koefL.append(L(s, t_n_U, t_n_T, l) * step)
        koefT_n_U.append(T_n_U(s, l, t_i_U, t_n_U) * step)
        koefT_i_U.append(T_i_U(t_n_U, t_i_U) * step)
        koefT_n_T.append(T_n_T(t_i_T, t_n_U, t_n_T) * step)
        koefT_i_T.append(T_i_T(t_n_T, t_i_U, t_i_T) * step)


        #second koef[1]
```

```
        koefS.append(S(s + (1/4) * koefS[0]) * step)
        koefL.append(L(s + (1/4) * koefS[0], t_n_U + (1/4) * koefT_n_U[0], t_n_T +
(1/4) * koefT_n_T[0], l + (1/4) * koefL[0]) * step)
        koefT_n_U.append(T_n_U(s + (1/4) * koefS[0], l + (1/4) * koefL[0], t_i_U +
(1/4) * koefT_i_U[0], t_n_U + (1/4) * koefT_n_U[0]) * step)
        koefT_i_U.append(T_i_U(t_n_U + (1/4) * koefT_n_U[0], t_i_U + (1/4) * koefT_
i_U[0]) * step)
        koefT_n_T.append(T_n_T(t_i_T + (1/4) * koefT_i_T[0], t_n_U + (1/4) * koefT_
n_U[0], t_n_T + (1/4) * koefT_n_T[0]) * step)
        koefT_i_T.append(T_i_T(t_n_T + (1/4) * koefT_n_T[0], t_i_U + (1/4) * koefT_
i_U[0], t_i_T + (1/4) * koefT_i_T[0]) * step)


        #third koef[2]
        koefS.append(S(s + (3/32) * koefS[0] + (9/32) * koefS[1]) * step)
        koefL.append(L(s + (3/32) * koefS[0] + (9/32) * koefS[1], t_n_U + (3/32) *
koefT_n_U[0] + (9/32) * koefT_n_U[1], t_n_T + (3/32) * koefT_n_T[0] + (9/32) * koef
T_n_T[1], l + (3/32) * koefL[0] + (9/32) * koefL[1]) * step)
        koefT_n_U.append(T_n_U(s + (3/32) * koefS[0] + (9/32) * koefS[1], l + (3/32
) * koefL[0] + (9/32) * koefL[1], t_i_U + (3/32) * koefT_i_U[0] + (9/32) * koefT_i_
U[1], t_n_U + (3/32) * koefT_n_U[0] + (9/32) * koefT_n_U[1]) * step)
        koefT_i_U.append(T_i_U(t_n_U + (3/32) * koefT_n_U[0] + (9/32) * koefT_n_U[1
], t_i_U + (3/32) * koefT_i_U[0] + (9/32) * koefT_i_U[1]) * step)
        koefT_n_T.append(T_n_T(t_i_T + (3/32) * koefT_i_T[0] + (9/32) * koefT_i_T[1
], t_n_U + (3/32) * koefT_n_U[0] + (9/32) * koefT_n_U[1], t_n_T + (3/32) * koefT_n_
T[0] + (9/32) * koefT_n_T[1]) * step)
        koefT_i_T.append(T_i_T(t_n_T + (3/32) * koefT_n_T[0] + (9/32) * koefT_n_T[1
], t_i_U + (3/32) * koefT_i_U[0] + (9/32) * koefT_i_U[1], t_i_T + (3/32) * koefT_i_
T[0] + (9/32) * koefT_i_T[1]) * step)


        #forth koef[3]

        koefS.append(S(s + (1932/2197) * koefS[0] - (7200/2197) * koefS[1] + (7296/
2197) * koefS[2]) * step)

        koefL.append(L(s + (1932/2197) * koefS[0] - (7200/2197) * koefS[1] + (7296/
2197) * koefS[2],
                       t_n_U + (1932/2197) * koefT_n_U[0] - (7200/2197) * koefT_n_U
[1] + (7296/2197) * koefT_n_U[2],
                       t_n_T + (1932/2197) * koefT_n_T[0] - (7200/2197) * koefT_n_T
[1] + (7296/2197) * koefT_n_T[2],
                       l + (1932/2197) * koefL[0] - (7200/2197) * koefL[1] + (7296/
2197) * koefL[2]) * step)

        koefT_n_U.append(T_n_U(s + (1932/2197) * koefS[0] - (7200/2197) * koefS[1]
+ (7296/2197) * koefS[2],
                               l + (1932/2197) * koefL[0] - (7200/2197) * koefL[1]
+ (7296/2197) * koefL[2],
```

```
                              t_i_U + (1932/2197) * koefT_i_U[0] - (7200/2197) * k
oefT_i_U[1] + (7296/2197) * koefT_i_U[2],
                              t_n_U + (1932/2197) * koefT_n_U[0] - (7200/2197) * k
oefT_n_U[1] + (7296/2197) * koefT_n_U[2]) * step)

        koefT_i_U.append(T_i_U(t_n_U + (1932/2197) * koefT_n_U[0] - (7200/2197) * k
oefT_n_U[1] + (7296/2197) * koefT_n_U[2],
                              t_i_U + (1932/2197) * koefT_i_U[0] - (7200/2197) * k
oefT_i_U[1] + (7296/2197) * koefT_i_U[2]) * step)

        koefT_n_T.append(T_n_T(t_i_T + (1932/2197) * koefT_i_T[0] - (7200/2197) * k
oefT_i_T[1] + (7296/2197) * koefT_i_T[2],
                              t_n_U + (1932/2197) * koefT_n_U[0] - (7200/2197) * k
oefT_n_U[1] + (7296/2197) * koefT_n_U[2],
                              t_n_T + (1932/2197) * koefT_n_T[0] - (7200/2197) * k
oefT_n_T[1] + (7296/2197) * koefT_n_T[2]) * step)

        koefT_i_T.append(T_i_T(t_n_T + (1932/2197) * koefT_n_T[0] - (7200/2197) * k
oefT_n_T[1] + (7296/2197) * koefT_n_T[2],
                              t_i_U + (1932/2197) * koefT_i_U[0] - (7200/2197) * k
oefT_i_U[1] + (7296/2197) * koefT_i_U[2],
                              t_i_T + (1932/2197) * koefT_i_T[0] - (7200/2197) * k
oefT_i_T[1] + (7296/2197) * koefT_i_T[2]) * step)


        #fifth koef[4]

        koefS.append(S(s + (439/216) * koefS[0] - 8 * koefS[1] + (3680/513) * koefS
[2] - (845/4104) * koefS[3]) * step)

        koefL.append(L(s + (439/216) * koefS[0] - 8 * koefS[1] + (3680/513) * koefS
[2] - (845/4104) * koefS[3],
                       t_n_U + (439/216) * koefT_n_U[0] - 8 * koefT_n_U[1] + (3680/
513) * koefT_n_U[2] - (845/4104) * koefT_n_U[3],
                       t_n_T + (439/216) * koefT_n_T[0] - 8 * koefT_n_T[1] + (3680/
513) * koefT_n_T[2] - (845/4104) * koefT_n_T[3],
                       l + (439/216) * koefL[0] - 8 * koefL[1] + (3680/513) * koefL
[2] - (845/4104) * koefL[3]) * step)

        koefT_n_U.append(T_n_U(s + (439/216) * koefS[0] - 8 * koefS[1] + (3680/513)
 * koefS[2] - (845/4104) * koefS[3],
                              l + (439/216) * koefL[0] - 8 * koefL[1] + (3680/513)
 * koefL[2] - (845/4104) * koefL[3],
                              t_i_U + (439/216) * koefT_i_U[0] - 8 * koefT_i_U[1]
+ (3680/513) * koefT_i_U[2] - (845/4104) * koefT_i_U[3],
                              t_n_U + (439/216) * koefT_n_U[0] - 8 * koefT_n_U[1]
+ (3680/513) * koefT_n_U[2] - (845/4104) * koefT_n_U[3]) * step)

        koefT_i_U.append(T_i_U(t_n_U + (439/216) * koefT_n_U[0] - 8 * koefT_n_U[1]
+ (3680/513) * koefT_n_U[2] - (845/4104) * koefT_n_U[3],
```

```
                              t_i_U + (439/216) * koefT_i_U[0] - 8 * koefT_i_U[1]
+ (3680/513) * koefT_i_U[2] - (845/4104) * koefT_i_U[3]) * step)

        koefT_n_T.append(T_n_T(t_i_T + (439/216) * koefT_i_T[0] - 8 * koefT_i_T[1]
+ (3680/513) * koefT_i_T[2] - (845/4104) * koefT_i_T[3],
                              t_n_U + (439/216) * koefT_n_U[0] - 8 * koefT_n_U[1]
+ (3680/513) * koefT_n_U[2] - (845/4104) * koefT_n_U[3],
                              t_n_T + (439/216) * koefT_n_T[0] - 8 * koefT_n_T[1]
+ (3680/513) * koefT_n_T[2] - (845/4104) * koefT_n_T[3]) * step)

        koefT_i_T.append(T_i_T(t_n_T + (439/216) * koefT_n_T[0] - 8 * koefT_n_T[1]
+ (3680/513) * koefT_n_T[2] - (845/4104) * koefT_n_T[3],
                              t_i_U + (439/216) * koefT_i_U[0] - 8 * koefT_i_U[1]
+ (3680/513) * koefT_i_U[2] - (845/4104) * koefT_i_U[3],
                              t_i_T + (439/216) * koefT_i_T[0] - 8 * koefT_i_T[1]
+ (3680/513) * koefT_i_T[2] - (845/4104) * koefT_i_T[3]) * step)

        #sixth koef[5]

        koefS.append(S(s - (8/27) * koefS[0] + 2 * koefS[1] - (3544/2565) * koefS[2
] + (1859/4104) * koefS[3] - (11/40) * koefS[4]) * step)

        koefL.append(L(s - (8/27) * koefS[0] + 2 * koefS[1] - (3544/2565) * koefS[2
] + (1859/4104) * koefS[3] - (11/40) * koefS[4],
                       t_n_U - (8/27) * koefT_n_U[0] + 2 * koefT_n_U[1] - (3544/256
5) * koefT_n_U[2] + (1859/4104) * koefT_n_U[3] - (11/40) * koefT_n_U[4],
                       t_n_T - (8/27) * koefT_n_T[0] + 2 * koefT_n_T[1] - (3544/256
5) * koefT_n_T[2] + (1859/4104) * koefT_n_T[3] - (11/40) * koefT_n_T[4],
                       l - (8/27) * koefL[0] + 2 * koefL[1] - (3544/2565) * koefL[2
] + (1859/4104) * koefL[3] - (11/40) * koefL[4]) * step)

        koefT_n_U.append(T_n_U(s - (8/27) * koefS[0] + 2 * koefS[1] - (3544/2565) *
 koefS[2] + (1859/4104) * koefS[3] - (11/40) * koefS[4],
                              l - (8/27) * koefL[0] + 2 * koefL[1] - (3544/2565) *
 koefL[2] + (1859/4104) * koefL[3] - (11/40) * koefL[4],
                              t_i_U - (8/27) * koefT_i_U[0] + 2 * koefT_i_U[1] - (
3544/2565) * koefT_i_U[2] + (1859/4104) * koefT_i_U[3] - (11/40) * koefT_i_U[4],
                              t_n_U - (8/27) * koefT_n_U[0] + 2 * koefT_n_U[1] - (
3544/2565) * koefT_n_U[2] + (1859/4104) * koefT_n_U[3] - (11/40) * koefT_n_U[4]) *
step)

        koefT_i_U.append(T_i_U(t_n_U - (8/27) * koefT_n_U[0] + 2 * koefT_n_U[1] - (
3544/2565) * koefT_n_U[2] + (1859/4104) * koefT_n_U[3] - (11/40) * koefT_n_U[4],
                              t_i_U - (8/27) * koefT_i_U[0] + 2 * koefT_i_U[1] - (
3544/2565) * koefT_i_U[2] + (1859/4104) * koefT_i_U[3] - (11/40) * koefT_i_U[4]) *
step)

        koefT_n_T.append(T_n_T(t_i_T - (8/27) * koefT_i_T[0] + 2 * koefT_i_T[1] - (
3544/2565) * koefT_i_T[2] + (1859/4104) * koefT_i_T[3] - (11/40) * koefT_i_T[4],
```

```
                                t_n_U - (8/27) * koefT_n_U[0] + 2 * koefT_n_U[1] - (
3544/2565) * koefT_n_U[2] + (1859/4104) * koefT_n_U[3] - (11/40) * koefT_n_U[4],
                                t_n_T - (8/27) * koefT_n_T[0] + 2 * koefT_n_T[1] - (
3544/2565) * koefT_n_T[2] + (1859/4104) * koefT_n_T[3] - (11/40) * koefT_n_T[4]) *
step)

        koefT_i_T.append(T_i_T(t_n_T - (8/27) * koefT_n_T[0] + 2 * koefT_n_T[1] - (
3544/2565) * koefT_n_T[2] + (1859/4104) * koefT_n_T[3] - (11/40) * koefT_n_T[4],
                                t_i_U - (8/27) * koefT_i_U[0] + 2 * koefT_i_U[1] - (
3544/2565) * koefT_i_U[2] + (1859/4104) * koefT_i_U[3] - (11/40) * koefT_i_U[4],
                                t_i_T - (8/27) * koefT_i_T[0] + 2 * koefT_i_T[1] - (
3544/2565) * koefT_i_T[2] + (1859/4104) * koefT_i_T[3] - (11/40) * koefT_i_T[4]) *
step)


        s += (16/135) * koefS[0] + (6656/12825) * koefS[2] + (28561/56430) * koefS[
3] - (9/50) * koefS[4] + (2/55) * koefS[5]
        l += (16/135) * koefL[0] + (6656/12825) * koefL[2] + (28561/56430) * koefL[
3] - (9/50) * koefL[4] + (2/55) * koefL[5]
        t_n_U += (16/135) * koefT_n_U[0] + (6656/12825) * koefT_n_U[2] + (28561/564
30) * koefT_n_U[3] - (9/50) * koefT_n_U[4] + (2/55) * koefT_n_U[5]
        t_i_U += (16/135) * koefT_i_U[0] + (6656/12825) * koefT_i_U[2] + (28561/564
30) * koefT_i_U[3] - (9/50) * koefT_i_U[4] + (2/55) * koefT_i_U[5]
        t_n_T += (16/135) * koefT_n_T[0] + (6656/12825) * koefT_n_T[2] + (28561/564
30) * koefT_n_T[3] - (9/50) * koefT_n_T[4] + (2/55) * koefT_n_T[5]
        t_i_T += (16/135) * koefT_i_T[0] + (6656/12825) * koefT_i_T[2] + (28561/564
30) * koefT_i_T[3] - (9/50) * koefT_i_T[4] + (2/55) * koefT_i_T[5]

        if (math.isnan(s)):
            dotsS.append(0)
        else:
            dotsS.append(s)
        if (math.isnan(l)):
            dotsL.append(0)
        else:
            dotsL.append(l)
        if (math.isnan(t_n_U)):
            dotsT_n_U.append(0)
        else:
            dotsT_n_U.append(t_n_U)
        if (math.isnan(t_i_U)):
            dotsT_i_U.append(0)
        else:
            dotsT_i_U.append(t_i_U)
        if (math.isnan(t_n_T)):
            dotsT_n_T.append(0)
        else:
            dotsT_n_T.append(t_n_T)
        if (math.isnan(t_i_T)):
            dotsT_i_T.append(0)
```

```
        else:
            dotsT_i_T.append(t_i_T)

        koefS.clear()
        koefL.clear()
        koefT_n_U.clear()
        koefT_i_U.clear()
        koefT_n_T.clear()
        koefT_i_T.clear()

    return [dotsS, dotsL, dotsT_n_U, dotsT_i_U, dotsT_n_T, dotsT_i_T]

def countAccuracy(y1, y2, t1, t2): #suppose y1 is shorter
    err = []
    time = []
    for i in range(len(t2)):
        j = np.where(t1 == t2[i])[0]
        if (j.size != 0):
            err.append(abs(y2[i] - y1[j[0]]) / 15)
            time.append(t1[j])
    plt.plot(time, err, label = 'Runge error estimate 0')

def modified_runge(y0, time0, time, step, eps, mu, control, mult):
    s, l, t_n_U, t_i_U, t_n_T, t_i_T = y0
    #v2, f2 = v, f
    dotsS = []
    dotsL = []
    dotsT_n_U = []
    dotsT_i_U = []
    dotsT_n_T = []
    dotsT_i_T = []
    dotsS.append(s)
    dotsL.append(l)
    dotsT_n_U.append(t_n_U)
    dotsT_i_U.append(t_i_U)
    dotsT_n_T.append(t_n_T)
    dotsT_i_T.append(t_i_T)

    ES = []
    EL = []
    ET_n_U = []
    ET_i_U = []
    ET_n_T = []
    ET_i_T = []
    ES.append(0.0)
    EL.append(0.0)
    ET_n_U.append(0.0)
    ET_i_U.append(0.0)
    ET_n_T.append(0.0)
    ET_i_T.append(0.0)
```

```
    t = time0
    timeVec = []
    timeVec.append(t)
    ready = 0;
    print('Modified Runge:')
    while(t < time):
        readyTmp = int(t/time*100);
        if (ready != readyTmp):
            ready = readyTmp
            print(ready, '%')
        koefS = []
        koefL = []
        koefT_n_U = []
        koefT_i_U = []
        koefT_n_T = []
        koefT_i_T = []

        #first koef[0]
        koefS.append(S(s) * step)
        koefL.append(L(s, t_n_U, t_n_T, l) * step)
        koefT_n_U.append(T_n_U(s, l, t_i_U, t_n_U) * step)
        koefT_i_U.append(T_i_U(t_n_U, t_i_U) * step)
        koefT_n_T.append(T_n_T(t_i_T, t_n_U, t_n_T) * step)
        koefT_i_T.append(T_i_T(t_n_T, t_i_U, t_i_T) * step)

        #print('first koef pased')

        #second koef[1]
        koefS.append(S(s + (1/4) * koefS[0]) * step)
        koefL.append(L(s + (1/4) * koefS[0], t_n_U + (1/4) * koefT_n_U[0], t_n_T +
(1/4) * koefT_n_T[0], l + (1/4) * koefL[0]) * step)
        koefT_n_U.append(T_n_U(s + (1/4) * koefS[0], l + (1/4) * koefL[0], t_i_U +
(1/4) * koefT_i_U[0], t_n_U + (1/4) * koefT_n_U[0]) * step)
        koefT_i_U.append(T_i_U(t_n_U + (1/4) * koefT_n_U[0], t_i_U + (1/4) * koefT_
i_U[0]) * step)
        koefT_n_T.append(T_n_T(t_i_T + (1/4) * koefT_i_T[0], t_n_U + (1/4) * koefT_
n_U[0], t_n_T + (1/4) * koefT_n_T[0]) * step)
        koefT_i_T.append(T_i_T(t_n_T + (1/4) * koefT_n_T[0], t_i_U + (1/4) * koefT_
i_U[0], t_i_T + (1/4) * koefT_i_T[0]) * step)

        #print('second koef pased')

        #third koef[2]
        koefS.append(S(s + (3/32) * koefS[0] + (9/32) * koefS[1]) * step)
        koefL.append(L(s + (3/32) * koefS[0] + (9/32) * koefS[1], t_n_U + (3/32) *
koefT_n_U[0] + (9/32) * koefT_n_U[1], t_n_T + (3/32) * koefT_n_T[0] + (9/32) * koef
T_n_T[1], l + (3/32) * koefL[0] + (9/32) * koefL[1]) * step)
```

```
        koefT_n_U.append(T_n_U(s + (3/32) * koefS[0] + (9/32) * koefS[1], l + (3/32
) * koefL[0] + (9/32) * koefL[1], t_i_U + (3/32) * koefT_i_U[0] + (9/32) * koefT_i_
U[1], t_n_U + (3/32) * koefT_n_U[0] + (9/32) * koefT_n_U[1]) * step)
        koefT_i_U.append(T_i_U(t_n_U + (3/32) * koefT_n_U[0] + (9/32) * koefT_n_U[1
], t_i_U + (3/32) * koefT_i_U[0] + (9/32) * koefT_i_U[1]) * step)
        koefT_n_T.append(T_n_T(t_i_T + (3/32) * koefT_i_T[0] + (9/32) * koefT_i_T[1
], t_n_U + (3/32) * koefT_n_U[0] + (9/32) * koefT_n_U[1], t_n_T + (3/32) * koefT_n_
T[0] + (9/32) * koefT_n_T[1]) * step)
        koefT_i_T.append(T_i_T(t_n_T + (3/32) * koefT_n_T[0] + (9/32) * koefT_n_T[1
], t_i_U + (3/32) * koefT_i_U[0] + (9/32) * koefT_i_U[1], t_i_T + (3/32) * koefT_i_
T[0] + (9/32) * koefT_i_T[1]) * step)

        #print('third koef pased')

        #forth koef[3]

        koefS.append(S(s + (1932/2197) * koefS[0] - (7200/2197) * koefS[1] + (7296/
2197) * koefS[2]) * step)

        koefL.append(L(s + (1932/2197) * koefS[0] - (7200/2197) * koefS[1] + (7296/
2197) * koefS[2],
                       t_n_U + (1932/2197) * koefT_n_U[0] - (7200/2197) * koefT_n_U
[1] + (7296/2197) * koefT_n_U[2],
                       t_n_T + (1932/2197) * koefT_n_T[0] - (7200/2197) * koefT_n_T
[1] + (7296/2197) * koefT_n_T[2],
                       l + (1932/2197) * koefL[0] - (7200/2197) * koefL[1] + (7296/
2197) * koefL[2]) * step)

        koefT_n_U.append(T_n_U(s + (1932/2197) * koefS[0] - (7200/2197) * koefS[1]
+ (7296/2197) * koefS[2],
                               l + (1932/2197) * koefL[0] - (7200/2197) * koefL[1]
+ (7296/2197) * koefL[2],
                               t_i_U + (1932/2197) * koefT_i_U[0] - (7200/2197) * k
oefT_i_U[1] + (7296/2197) * koefT_i_U[2],
                               t_n_U + (1932/2197) * koefT_n_U[0] - (7200/2197) * k
oefT_n_U[1] + (7296/2197) * koefT_n_U[2]) * step)

        koefT_i_U.append(T_i_U(t_n_U + (1932/2197) * koefT_n_U[0] - (7200/2197) * k
oefT_n_U[1] + (7296/2197) * koefT_n_U[2],
                               t_i_U + (1932/2197) * koefT_i_U[0] - (7200/2197) * k
oefT_i_U[1] + (7296/2197) * koefT_i_U[2]) * step)

        koefT_n_T.append(T_n_T(t_i_T + (1932/2197) * koefT_i_T[0] - (7200/2197) * k
oefT_i_T[1] + (7296/2197) * koefT_i_T[2],
                               t_n_U + (1932/2197) * koefT_n_U[0] - (7200/2197) * k
oefT_n_U[1] + (7296/2197) * koefT_n_U[2],
                               t_n_T + (1932/2197) * koefT_n_T[0] - (7200/2197) * k
oefT_n_T[1] + (7296/2197) * koefT_n_T[2]) * step)
```

```
        koefT_i_T.append(T_i_T(t_n_T + (1932/2197) * koefT_n_T[0] - (7200/2197) * k
oefT_n_T[1] + (7296/2197) * koefT_n_T[2],
                               t_i_U + (1932/2197) * koefT_i_U[0] - (7200/2197) * k
oefT_i_U[1] + (7296/2197) * koefT_i_U[2],
                               t_i_T + (1932/2197) * koefT_i_T[0] - (7200/2197) * k
oefT_i_T[1] + (7296/2197) * koefT_i_T[2]) * step)

        #print('forth koef pased')

        #fifth koef[4]

        koefS.append(S(s + (439/216) * koefS[0] - 8 * koefS[1] + (3680/513) * koefS
[2] - (845/4104) * koefS[3]) * step)

        koefL.append(L(s + (439/216) * koefS[0] - 8 * koefS[1] + (3680/513) * koefS
[2] - (845/4104) * koefS[3],
                       t_n_U + (439/216) * koefT_n_U[0] - 8 * koefT_n_U[1] + (3680/
513) * koefT_n_U[2] - (845/4104) * koefT_n_U[3],
                       t_n_T + (439/216) * koefT_n_T[0] - 8 * koefT_n_T[1] + (3680/
513) * koefT_n_T[2] - (845/4104) * koefT_n_T[3],
                       l + (439/216) * koefL[0] - 8 * koefL[1] + (3680/513) * koefL
[2] - (845/4104) * koefL[3]) * step)

        koefT_n_U.append(T_n_U(s + (439/216) * koefS[0] - 8 * koefS[1] + (3680/513)
 * koefS[2] - (845/4104) * koefS[3],
                               l + (439/216) * koefL[0] - 8 * koefL[1] + (3680/513)
 * koefL[2] - (845/4104) * koefL[3],
                               t_i_U + (439/216) * koefT_i_U[0] - 8 * koefT_i_U[1]
+ (3680/513) * koefT_i_U[2] - (845/4104) * koefT_i_U[3],
                               t_n_U + (439/216) * koefT_n_U[0] - 8 * koefT_n_U[1]
+ (3680/513) * koefT_n_U[2] - (845/4104) * koefT_n_U[3]) * step)

        koefT_i_U.append(T_i_U(t_n_U + (439/216) * koefT_n_U[0] - 8 * koefT_n_U[1]
+ (3680/513) * koefT_n_U[2] - (845/4104) * koefT_n_U[3],
                               t_i_U + (439/216) * koefT_i_U[0] - 8 * koefT_i_U[1]
+ (3680/513) * koefT_i_U[2] - (845/4104) * koefT_i_U[3]) * step)

        koefT_n_T.append(T_n_T(t_i_T + (439/216) * koefT_i_T[0] - 8 * koefT_i_T[1]
+ (3680/513) * koefT_i_T[2] - (845/4104) * koefT_i_T[3],
                               t_n_U + (439/216) * koefT_n_U[0] - 8 * koefT_n_U[1]
+ (3680/513) * koefT_n_U[2] - (845/4104) * koefT_n_U[3],
                               t_n_T + (439/216) * koefT_n_T[0] - 8 * koefT_n_T[1]
+ (3680/513) * koefT_n_T[2] - (845/4104) * koefT_n_T[3]) * step)

        koefT_i_T.append(T_i_T(t_n_T + (439/216) * koefT_n_T[0] - 8 * koefT_n_T[1]
+ (3680/513) * koefT_n_T[2] - (845/4104) * koefT_n_T[3],
                               t_i_U + (439/216) * koefT_i_U[0] - 8 * koefT_i_U[1]
+ (3680/513) * koefT_i_U[2] - (845/4104) * koefT_i_U[3],
                               t_i_T + (439/216) * koefT_i_T[0] - 8 * koefT_i_T[1]
+ (3680/513) * koefT_i_T[2] - (845/4104) * koefT_i_T[3]) * step)
```

```
        #print('fifth koef pased')

        #sixth koef[5]

        koefS.append(S(s - (8/27) * koefS[0] + 2 * koefS[1] - (3544/2565) * koefS[2
] + (1859/4104) * koefS[3] - (11/40) * koefS[4]) * step)

        koefL.append(L(s - (8/27) * koefS[0] + 2 * koefS[1] - (3544/2565) * koefS[2
] + (1859/4104) * koefS[3] - (11/40) * koefS[4],
                       t_n_U - (8/27) * koefT_n_U[0] + 2 * koefT_n_U[1] - (3544/256
5) * koefT_n_U[2] + (1859/4104) * koefT_n_U[3] - (11/40) * koefT_n_U[4],
                       t_n_T - (8/27) * koefT_n_T[0] + 2 * koefT_n_T[1] - (3544/256
5) * koefT_n_T[2] + (1859/4104) * koefT_n_T[3] - (11/40) * koefT_n_T[4],
                       l - (8/27) * koefL[0] + 2 * koefL[1] - (3544/2565) * koefL[2
] + (1859/4104) * koefL[3] - (11/40) * koefL[4]) * step)

        koefT_n_U.append(T_n_U(s - (8/27) * koefS[0] + 2 * koefS[1] - (3544/2565) *
 koefS[2] + (1859/4104) * koefS[3] - (11/40) * koefS[4],
                               l - (8/27) * koefL[0] + 2 * koefL[1] - (3544/2565) *
 koefL[2] + (1859/4104) * koefL[3] - (11/40) * koefL[4],
                               t_i_U - (8/27) * koefT_i_U[0] + 2 * koefT_i_U[1] - (
3544/2565) * koefT_i_U[2] + (1859/4104) * koefT_i_U[3] - (11/40) * koefT_i_U[4],
                               t_n_U - (8/27) * koefT_n_U[0] + 2 * koefT_n_U[1] - (
3544/2565) * koefT_n_U[2] + (1859/4104) * koefT_n_U[3] - (11/40) * koefT_n_U[4]) *
step)

        koefT_i_U.append(T_i_U(t_n_U - (8/27) * koefT_n_U[0] + 2 * koefT_n_U[1] - (
3544/2565) * koefT_n_U[2] + (1859/4104) * koefT_n_U[3] - (11/40) * koefT_n_U[4],
                               t_i_U - (8/27) * koefT_i_U[0] + 2 * koefT_i_U[1] - (
3544/2565) * koefT_i_U[2] + (1859/4104) * koefT_i_U[3] - (11/40) * koefT_i_U[4]) *
step)

        koefT_n_T.append(T_n_T(t_i_T - (8/27) * koefT_i_T[0] + 2 * koefT_i_T[1] - (
3544/2565) * koefT_i_T[2] + (1859/4104) * koefT_i_T[3] - (11/40) * koefT_i_T[4],
                               t_n_U - (8/27) * koefT_n_U[0] + 2 * koefT_n_U[1] - (
3544/2565) * koefT_n_U[2] + (1859/4104) * koefT_n_U[3] - (11/40) * koefT_n_U[4],
                               t_n_T - (8/27) * koefT_n_T[0] + 2 * koefT_n_T[1] - (
3544/2565) * koefT_n_T[2] + (1859/4104) * koefT_n_T[3] - (11/40) * koefT_n_T[4]) *
step)

        koefT_i_T.append(T_i_T(t_n_T - (8/27) * koefT_n_T[0] + 2 * koefT_n_T[1] - (
3544/2565) * koefT_n_T[2] + (1859/4104) * koefT_n_T[3] - (11/40) * koefT_n_T[4],
                               t_i_U - (8/27) * koefT_i_U[0] + 2 * koefT_i_U[1] - (
3544/2565) * koefT_i_U[2] + (1859/4104) * koefT_i_U[3] - (11/40) * koefT_i_U[4],
                               t_i_T - (8/27) * koefT_i_T[0] + 2 * koefT_i_T[1] - (
3544/2565) * koefT_i_T[2] + (1859/4104) * koefT_i_T[3] - (11/40) * koefT_i_T[4]) *
step)
```

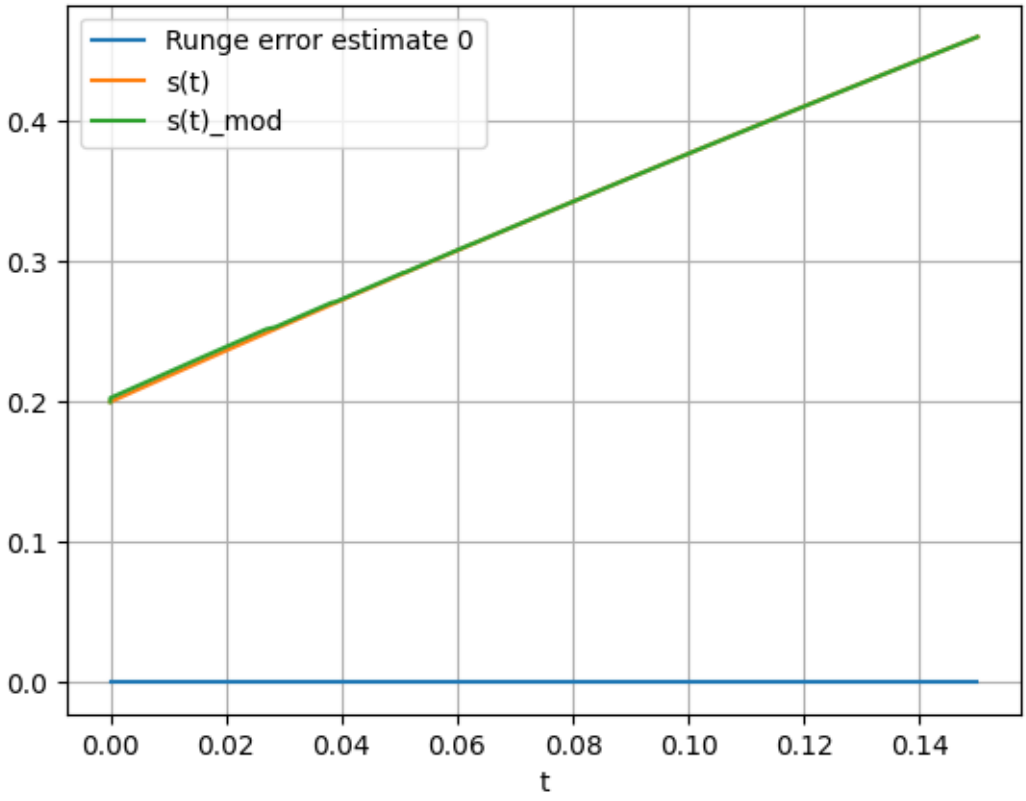
```
        #print('sixth koef pased')

        # index = 0
        # for index in range(len(timeVec)):
        #     v2 = dotsV[index]
        #     f2 = dotsF[index]

        s1 = s + (16/135) * koefS[0] + (6656/12825) * koefS[2] + (28561/56430) * ko
efS[3] - (9/50) * koefS[4] + (2/55) * koefS[5]
        l1 = l + (16/135) * koefL[0] + (6656/12825) * koefL[2] + (28561/56430) * ko
efL[3] - (9/50) * koefL[4] + (2/55) * koefL[5]
        t_n_U1 = t_n_U + (16/135) * koefT_n_U[0] + (6656/12825) * koefT_n_U[2] + (2
8561/56430) * koefT_n_U[3] - (9/50) * koefT_n_U[4] + (2/55) * koefT_n_U[5]
        t_i_U1 = t_i_U + (16/135) * koefT_i_U[0] + (6656/12825) * koefT_i_U[2] + (2
8561/56430) * koefT_i_U[3] - (9/50) * koefT_i_U[4] + (2/55) * koefT_i_U[5]
        t_n_T1 = t_n_T + (16/135) * koefT_n_T[0] + (6656/12825) * koefT_n_T[2] + (2
8561/56430) * koefT_n_T[3] - (9/50) * koefT_n_T[4] + (2/55) * koefT_n_T[5]
        t_i_T1 = t_i_T + (16/135) * koefT_i_T[0] + (6656/12825) * koefT_i_T[2] + (2
8561/56430) * koefT_i_T[3] - (9/50) * koefT_i_T[4] + (2/55) * koefT_i_T[5]

        #print('function pased')

        es = abs((1/360) * koefS[0] - (128/4275) * koefS[2] - (2197/75240) * koefS[
3] + (1/50) * koefS[4] + (2/55) * koefS[5])
        el = abs((1/360) * koefL[0] - (128/4275) * koefL[2] - (2197/75240) * koefL[
3] + (1/50) * koefL[4] + (2/55) * koefL[5])
        et_n_U = abs((1/360) * koefT_n_U[0] - (128/4275) * koefT_n_U[2] - (2197/752
40) * koefT_n_U[3] + (1/50) * koefT_n_U[4] + (2/55) * koefT_n_U[5])
        et_i_U = abs((1/360) * koefT_i_U[0] - (128/4275) * koefT_i_U[2] - (2197/752
40) * koefT_i_U[3] + (1/50) * koefT_i_U[4] + (2/55) * koefT_i_U[5])
        et_n_T = abs((1/360) * koefT_n_T[0] - (128/4275) * koefT_n_T[2] - (2197/752
40) * koefT_n_T[3] + (1/50) * koefT_n_T[4] + (2/55) * koefT_n_T[5])
        et_i_T = abs((1/360) * koefT_i_T[0] - (128/4275) * koefT_i_T[2] - (2197/752
40) * koefT_i_T[3] + (1/50) * koefT_i_T[4] + (2/55) * koefT_i_T[5])

        #print('error pased')

        if (control == 0):
            if (es >= eps):
                step /= mult
                continue
            elif (es < mu):
                step *= mult
                continue
        elif (control == 1):
            if (el >= eps):
                step /= mult
                continue
            elif (el < mu):
                step *= mult
```

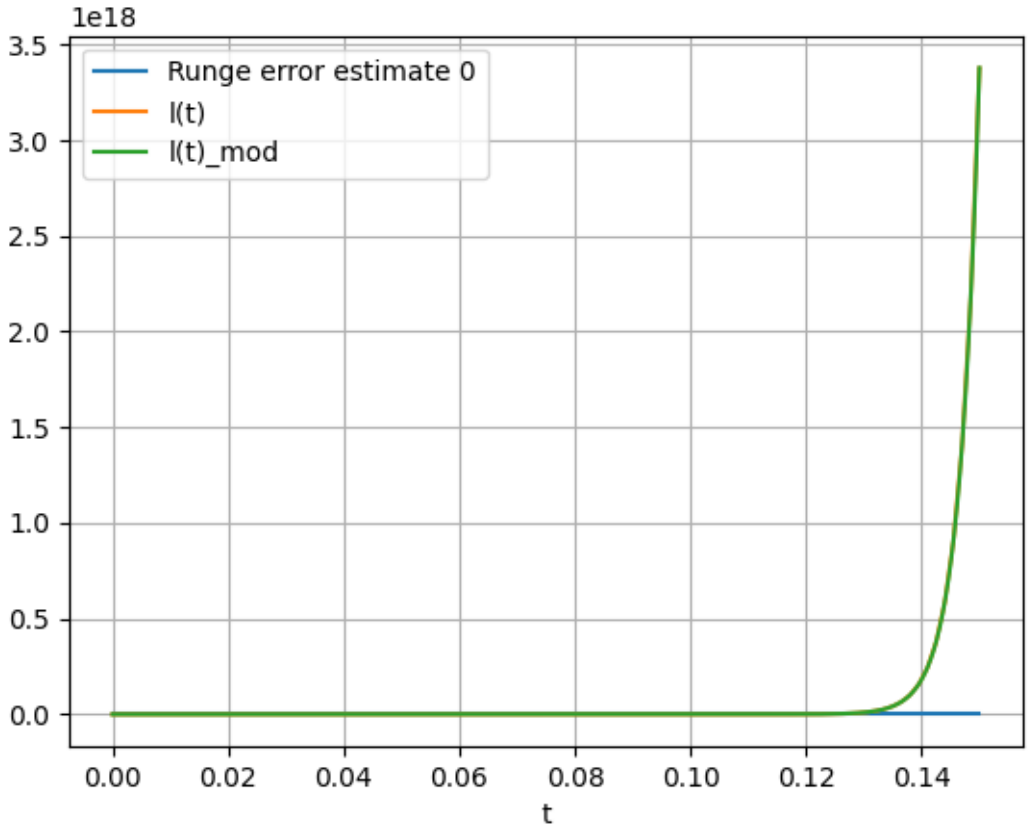
```
            continue
        elif (control == 2):
            if (et_n_U >= eps):
                step /= mult
                continue
            elif (et_n_U < mu):
                step *= mult
                continue
        elif (control == 3):
            if (et_i_U >= eps):
                step /= mult
                continue
            elif (et_i_U < mu):
                step *= mult
                continue
        elif (control == 4):
            if (et_n_T >= eps):
                step /= mult
                continue
            elif (et_n_T < mu):
                step *= mult
                continue
        else:
            if (et_i_T >= eps):
                step /= mult
                continue
            elif (et_i_T < mu):
                step *= mult
                continue

        #print(mult)

        #print('Control pased')

        if (math.isnan(s1)):
            dotsS.append(0)
            ES.append(0.0)
        else:
            dotsS.append(s1)
            ES.append(es)
        if (math.isnan(l1)):
            dotsL.append(0)
            EL.append(0.0)
        else:
            dotsL.append(l1)
            EL.append(el)
        if (math.isnan(t_n_U1)):
            dotsT_n_U.append(0)
            ET_n_U.append(0.0)
        else:
```

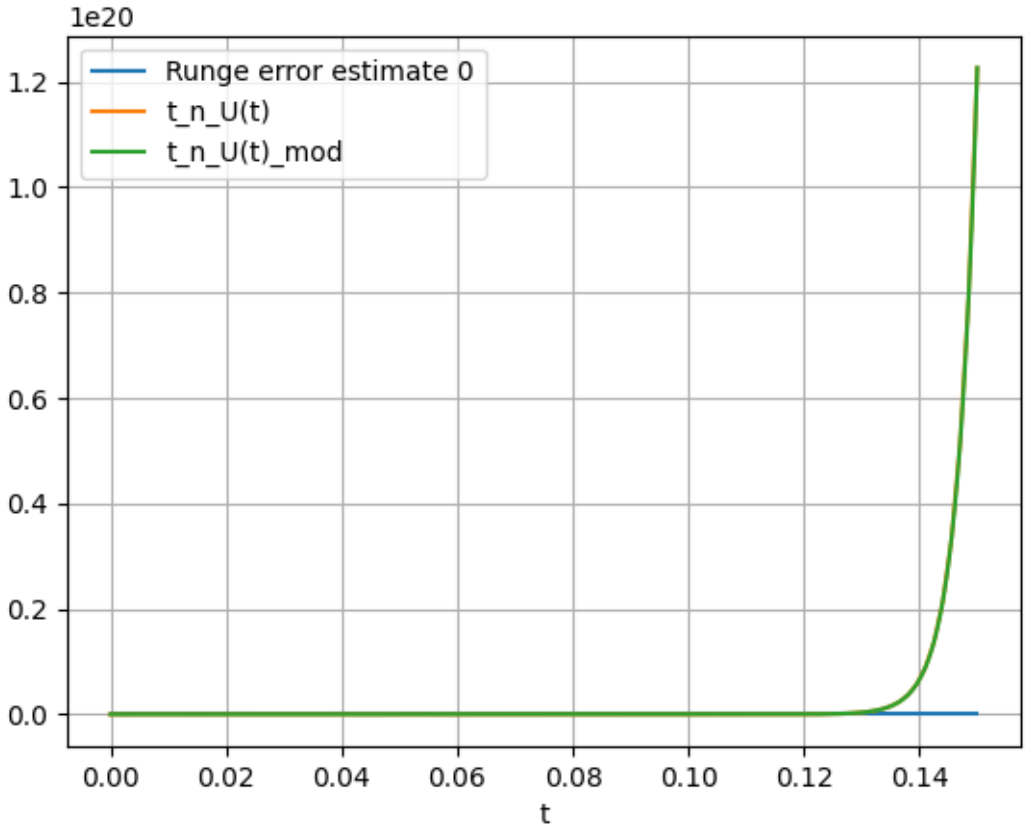
```
            dotsT_n_U.append(t_n_U1)
            ET_n_U.append(et_n_U)
        if (math.isnan(t_i_U1)):
            dotsT_i_U.append(0)
            ET_i_U.append(0.0)
        else:
            dotsT_i_U.append(t_i_U1)
            ET_i_U.append(et_i_U)
        if (math.isnan(t_n_T1)):
            dotsT_n_T.append(0)
            ET_n_T.append(0.0)
        else:
            dotsT_n_T.append(t_n_T1)
            ET_n_T.append(et_n_T)
        if (math.isnan(t_i_T1)):
            dotsT_i_T.append(0)
            ET_i_T.append(0.0)
        else:
            dotsT_i_T.append(t_i_T1)
            ET_i_T.append(et_i_T)

        #print('dots pased')

        koefS.clear()
        koefL.clear()
        koefT_n_U.clear()
        koefT_i_U.clear()
        koefT_n_T.clear()
        koefT_i_T.clear()

        #print('clear pased')

        timeVec.append(t)
        t += step
        s = s1
        l = l1
        t_n_U = t_n_U1
        t_i_U = t_i_U1
        t_n_T = t_n_T1
        t_i_T = t_i_T1

        #print (t)
    return [dotsS, dotsL, dotsT_n_U, dotsT_i_U, dotsT_n_T, dotsT_i_T, timeVec]


eps = 1e-3
mu = 1e-350
y0 = [0.20, 0.60, 0.02, 0.04, 0.01, 0.13]
start = 0
end = 20 # 0.15 - MODIFIED RUNGE
```

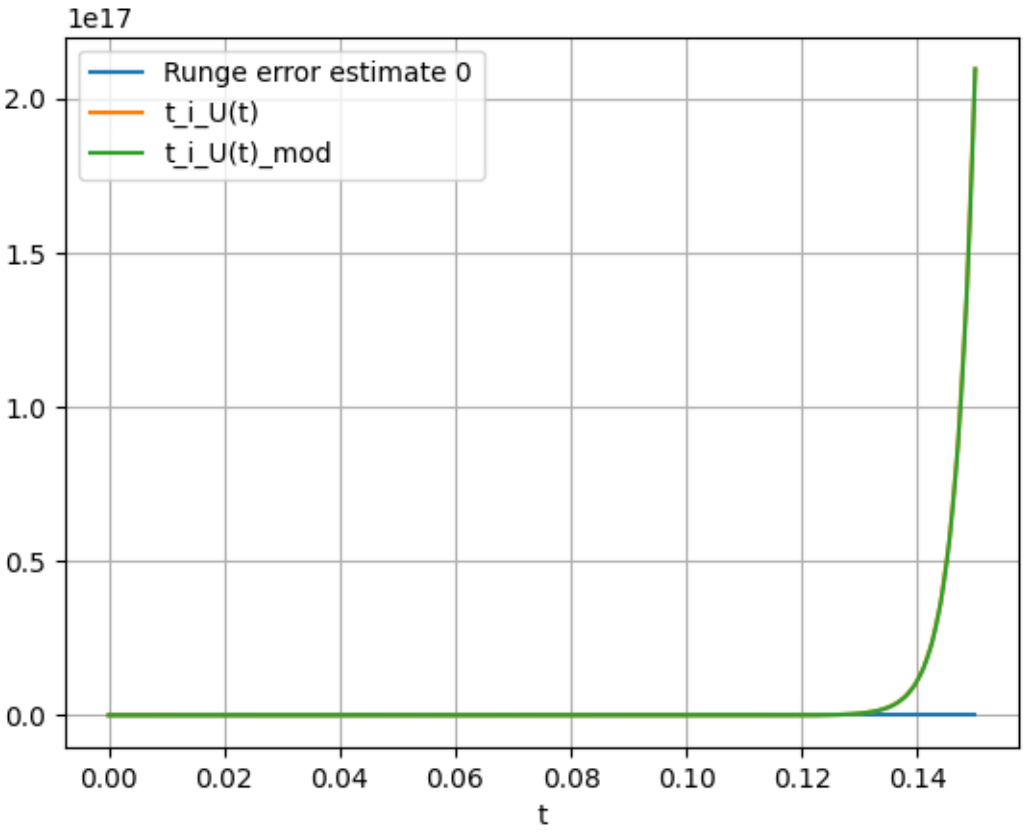
```
n = 100

step = (end - start) / n
t = np.linspace(start, end, n + 1)
solS, solL, solT_n_U, solT_i_U, solT_n_T, solT_i_T = rungeKutta(y0, start, end, ste
p)
#solS1, solL1, solT_n_U1, solT_i_U1, solT_n_T1, solT_i_T1, t1 = modified_runge(y0,
start, end, step, eps, mu, 3, 2)

#accuracy
n2 = n / 2
step2 = (end - start) / n2
t2 = np.linspace(start, end, int(n2) + 1)
solS2, solL2, solT_n_U2, solT_i_U2, solT_n_T2, solT_i_T2 = rungeKutta(y0, start, en
d, step2)
countAccuracy(solS2, solS, t2, t)
#plt.plot(t, solS, label = 's(t)')
#plt.plot(t1, solS1, label = 's(t)_mod')
plt.plot(t, solL, label = 'l(t)')
#plt.plot(t1, solL1, label = 'l(t)_mod')
#plt.plot(t, solT_n_U, label = 't_n_U(t)')
#plt.plot(t1, solT_n_U1, label = 't_n_U(t)_mod')
#plt.plot(t, solT_i_U, label = 't_i_U(t)')
#plt.plot(t1, solT_i_U1, label = 't_i_U(t)_mod')
#plt.plot(t, solT_n_T, label = 't_n_T(t)')
#plt.plot(t1, solT_n_T1, label = 't_n_T(t)_mod')
#plt.plot(t, solT_i_T, label = 't_i_T(t)')
#plt.plot(t1, solT_i_T1, label = 't_i_T(t)_mod')
plt.legend(loc = 'best')
plt.xlabel('t')
plt.yscale('linear')
plt.grid()
plt.show()
```

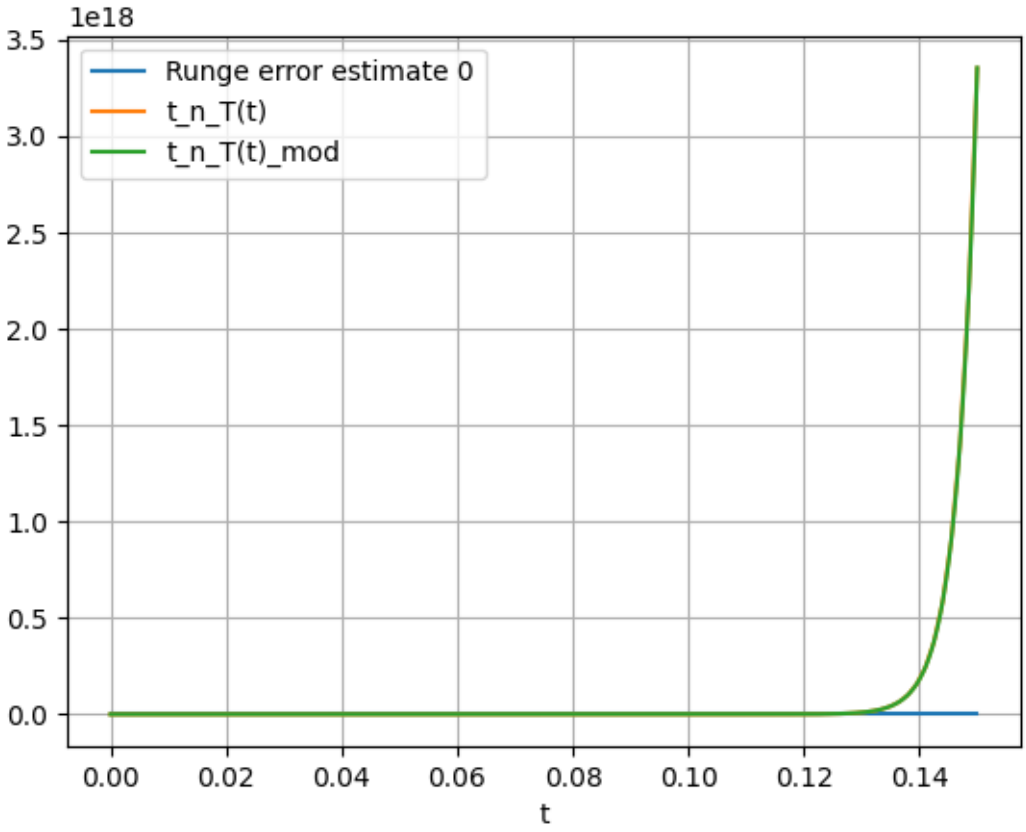
Вывод:

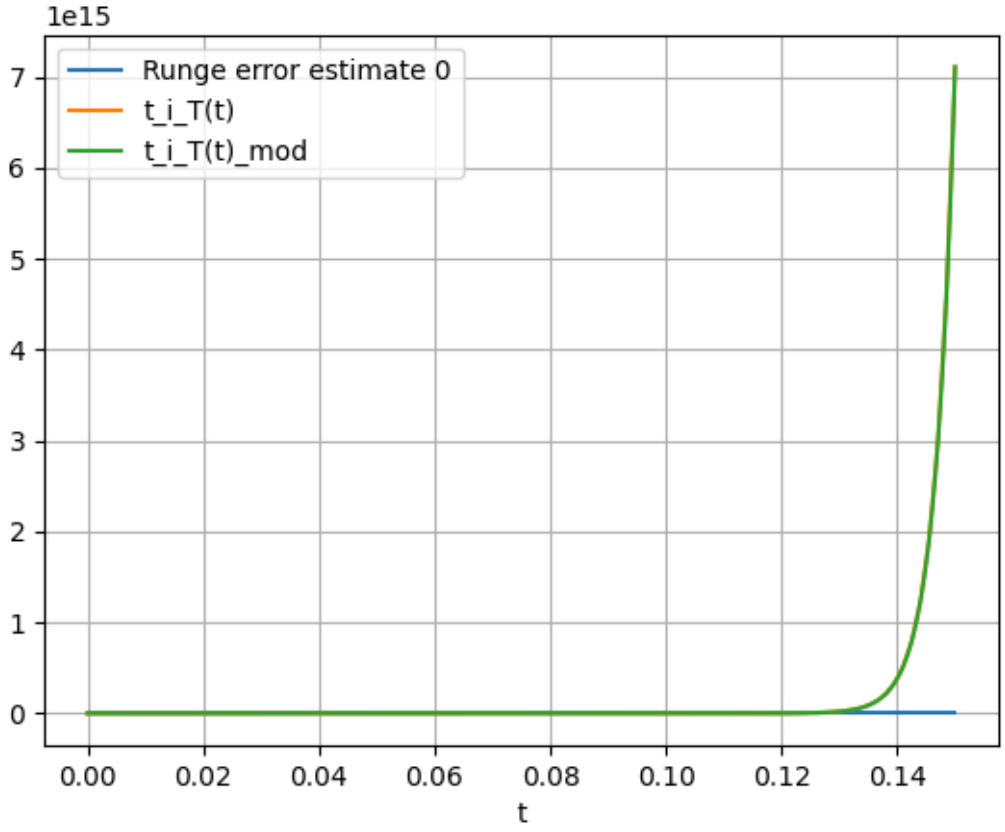

Runge error estimate 0
s(t)
s(t)_mod
0.0
0.1
0.2
0.3
0.4
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
t


1e18
Runge error estimate 0
I(t)
I(t)_mod
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
t


1e20
Runge error estimate 0
t_n_U(t)
t_n_U(t)_mod
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
t


1e17
Runge error estimate 0
t_i_U(t)
t_i_U(t)_mod
2.0
1.5
1.0
0.5
0.0
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
t


1e18
Runge error estimate 0
t_n_T(t)
t_n_T(t)_mod
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
t


1e15
Runge error estimate 0
t_i_T(t)
t_i_T(t)_mod
7
6
5
4
3
2
1
0
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
t