

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА**

**Факультет математики та інформатики
кафедра математичного моделювання**

**ЧИСЛОВЕ МОДЕЛЮВАННЯ РОЗВ'ЯЗКІВ ПОЧАТКОВИХ ЗАДАЧ
ДЛЯ ДИФЕРЕНЦІАЛЬНО-РІЗНИЦЕВИХ РІВНЯНЬ**

Кваліфікаційна робота

Рівень вищої освіти – другий (магістерський)

Виконав:

студент 2 курсу, 601 групи

Диренко Віктор Вікторович

Керівник:

доктор фізико-математичних наук,

професор Черевко І.М.

До захисту допущено

на засіданні кафедри

протокол № 5 від 26 листопада 2024 р.

Зав. кафедрою _____ проф. Черевко І.М.

АНОТАЦІЯ

У кваліфікаційній роботі досліджено алгоритми наближеного розв'язання початкових задач для диференціально-різницевих рівнянь. Запропоновано узагальнення θ методу на випадок диференціальних рівнянь нейтрального типу і побудовано схему автоматизації методу кроків з використанням лінійної інтерполяції та застосування кубічних інтерполяційних сплайнів. Розроблено прикладний додаток для реалізації запропонованих в роботі алгоритмів, проведено числові експерименти для модельних тестових прикладів.

Ключові слова: диференціально різницеві рівняння, початкова задача, апроксимація, метод кроків, θ різницеві схеми, кубічний сплайн, Python, Django, прикладний додаток.

ABSTRACT

In the qualification work, algorithms for approximate solution of initial problems for differential-difference equations are investigated. A generalization of the θ method for the case of neutral-type differential equations is proposed and a scheme for automating the step method using linear interpolation and cubic interpolation splines is constructed. An application is developed to implement the algorithms proposed in the work, numerical experiments are conducted for model test cases.

Keywords: differential-difference equations, initial problem, approximation, step method, θ difference schemes, cubic spline, Python, Django, application.

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів наукових досліджень інших авторів мають посилання на відповідне джерело.

_____ В. В. Диренко

Зміст

Вступ.....	5
1. Основні відомості про диференціально-різницеві рівняння.....	7
1.1. Вступ.....	7
1.2. Класифікація ДРР.....	8
1.3. Постановка початкової задачі.....	9
1.4. Метод кроків.....	10
1.5. Гладкість розв'язків початкової задачі для диференціальних рівнянь запізнюючого і нейтрального типів.....	12
2. Наближені алгоритми розв'язування початкових задач для диференціально-різницевих рівнянь.....	14
2.1. Вступ.....	14
2.2. Сім'я θ різницевих схем.....	14
2.3. Узагальнення θ різницевої схеми на рівняння нейтрального типу.....	16
3. Алгоритм автоматизації методу кроків.....	18
3.1. Схема алгоритму	18
3.2. Реалізація алгоритму.....	19
3.2.1. Лінійна інтерполяція.....	19
3.2.2. Кубічна інтерполяція.....	20
3.3. Кубічні сплайни.....	21
3.3.1. Вступ.....	21
3.3.2. Алгоритм побудови.....	22
3.4. Метод прогонки.....	26
4. Моделювання початкових задач на ЕОМ.....	29
4.1. Технології для створення прикладного додатку.....	29
4.1.1. Python.....	29
4.1.2. Фреймворк Django.....	30
4.1.3. Використані бібліотеки Python.....	31
4.2. Опис розробленого прикладного додатку.....	33
4.3. Аналіз числових експериментів.....	38

Висновки.....	45
Список літератури.....	46
Додаток 1 (метод Рунге-Кутти 2 порядку).....	49
Додаток 2 (метод Ейлера).....	50
Додаток 3 (обчислення коефіцієнтів кубічного сплайну).....	51
Додаток 4 (розв'язання різних видів ДРР).....	52
Додаток 5 (клієнтська частина додатку).....	56

Вступ

Диференціально-функціональні рівняння і особливо диференціальні рівняння з аргументом, що відхиляється, знаходять багато застосувань у теорії автоматичного регулювання, автоматичі і телемеханіці, теоретичній кібернетиці.

В останні роки сфери застосувань диференціальних рівнянь з аргументом, що відхиляється, значно розширилась, захопивши не тільки багато питань фізики, техніки, а також деякі питання економічних наук (розподіл продукції у відповідності з потребами покупців), біологічних наук (поширення епідемій із врахуванням латентного періоду) та інших.

Широке застосування диференціально-функціональних рівнянь стимулювало ріст досліджень таких рівнянь. Головний напрямок в теорії диференціально-функціональних рівнянь полягає в поширенні на них найважливіших результатів, відомих для звичайних диференціальних рівнянь.

Для знаходження точних розв'язків диференціально-функціональних рівнянь відсутні універсальні алгоритми, тому задача побудови алгоритмів знаходження наближених розв'язків таких задач є актуальною.

У даній кваліфікаційній роботі здійснено аналіз θ алгоритму для наближеного знаходження розв'язків початкових задач для рівнянь із сталим запізненням та здійснено його узагальнення на випадок диференціальних рівнянь нейтрального типу. В роботі також запропоновано схему автоматизації методу кроків з використанням лінійної інтерполяції та застосування кубічних інтерполяційних сплайнів.

Мета і завдання досліджень. Метою кваліфікаційної роботи є дослідження алгоритмів знаходження наближених розв'язків початкових задач для диференціально-різницевих рівнянь.

Для реалізації мети кваліфікаційної роботи поставлено такі завдання:

- узагальнити сім'ю θ різницевих схем на випадок диференціально-різницевих рівнянь нейтрального типу;

- автоматизувати реалізацію методу кроків для диференціально-різницевого рівняння;
- розробити прикладний програмний додаток та здійснити числові експерименти для модельних тестових прикладів.

Об'єкт досліджень. Початкові задачі для диференціально-різницевого рівняння із запізненням та нейтрального типів.

Предмет досліджень. Розробка прикладного програмного додатку для наближеного розв'язання початкових задач для різних класів диференціально-різницевого рівняння.

Практичне значення. Розроблений програмний додаток буде адаптований для розв'язання типових початкових задач. Він надає корисні інструменти для аналізу та візуалізації результатів числових експериментів, може використовуватись у навчальних цілях для студентів комп'ютерних наук та системного аналізу.

Структура роботи. Кваліфікаційна робота містить вступ, 4 розділи, висновки, список використаних джерел (28 найменування) та п'ять додатків.

Результати досліджень кваліфікаційної роботи доповідалися на VII Міжнародній науково-практичній конференції «Моделювання, керування та інформаційні технології (МСІТ–2024)» 7–9 листопада 2024 року, м. Рівне [1].

РОЗДІЛ 1. ОСНОВНІ ВІДОМОСТІ ПРО ДИФЕРЕНЦІАЛЬНО-РІЗНИЦЕВІ РІВНЯННЯ

1.1. Вступ

Диференціальні рівняння з аргументом, що відхиляється являються математичними моделями різних прикладних процесів в нелінійних коливаннях, об'єктах автоматизованого керування, хіміко-біологічних та економічних явищах та інших процесах майбутнє яких залежить від їхньої передісторії [2-4]. За допомогою таких рівнянь вдалося описати більш точно протікання процесів і явищ у фізиці, біології, екології, імунології та інших науках. Зокрема, в біологічних процесах, розвиток пов'язаний з такими довготривалими явищами, як розмноження, ріст і помирання, які відбуваються не миттєво, а з певними затримками. Численні системи технічного керування включають фізичні затримки, яку необхідно враховувати при прогнозуванні та налаштуванні алгоритмів управління зі зворотними зв'язками [5-6].

Багаточисленні застосування систем із запізненням стимулювали високий ріст наукових досліджень багатьох напрямків теорії диференціально-функціональних рівнянь та виявлення нових цікавих теоретичних та практичних задач, що потребують розв'язання.

Ще з середини минулого століття дослідження диференціально-функціональних рівнянь активно розвивалось такими визначними математиками як Р. Беллман, Дж. Хейл, А. Халанай, В. П. Рубаник, В. Ю. Слюсарчук, В. І. Фодчук та багатьма іншими. Наукові результати, що вони дістали, становлять зараз сучасну теорію диференціальних рівнянь із запізненням [7-11].

Не дивлячись на важливі здобутки при вивченні диференціально-функціональних рівнянь, на теперішній час ще залишається достатня кількість нерозв'язаних проблем, які стосуються як теорії, так і застосувань таких рівнянь.

Як правило на такі рівняння узагальнюються асимптотичні методи М. М. Боголюбова, Ю. А. Митропольського, метод усереднення та чисельно-аналітичні методи А. М. Самойленка [12-13].

При вивченні задач практичної стійкості, коливання та біфуркації, керування та стабілізації розв'язків лінійних диференціально-різницевих рівнянь важливе значення має розташування коренів їх характеристичних рівнянь, які у випадку диференціально-різницевих рівнянь називаються квазіполіномами.

На даний час відсутні поширені алгоритми знаходження нулів квазіполіномів. Важливі дослідження, що тут розвиваються, направлені на побудову спеціальних многочленів, нулі яких наближають нулі квазіполіномів. При дослідженні схем апроксимації лінійних диференціально-різницевих рівнянь встановлено, що для наближення неасимптотичних коренів відповідних квазіполіномів можна використовувати характеристичні многочлени апроксимуючих лінійних систем звичайних диференціальних рівнянь.

Головна увага в кваліфікаційній роботі приділена побудові та обґрунтуванню схеми апроксимації розв'язків деяких класів диференціально-функціональних рівнянь із запізненням і нейтрального типу.

1.2. Класифікація ДРР

Означення 1. [7-8] *Диференціальні рівняння, в яких шукана функція та її похідні входять при різних значеннях своїх аргументів, називаються диференціальними рівняннями з відхиленням аргументу.*

Наприклад:

$$\dot{x}(t) = f(t, x(t), x(t - \tau)), \tau > 0. \quad (1.1)$$

Такі рівняння відомі в науковій літературі вже більше двох століть і є предметом активних досліджень протягом останніх 50 років у зв'язку з багатьма застосуваннями у різних прикладних задачах. Якщо відхилення аргументу сталі то такі рівняння називаються диференціально-різницевиими рівняннями (ДРР).

У науковій літературі виділяють такі класи ДРР: рівняння з запізненням, рівняння нейтрального типу та рівняння з випередженням.

Означення 2. [7-8] Якщо в ДРР старша похідна входить тільки один раз, і при тому з найбільшим значенням аргументу, то його відносять до запізнюючого типу. Якщо ж старша похідна не входить з найбільшим значенням аргументу, то його відносять до випереджуючого типу. Якщо в ДРР старша похідна входить кілька разів з різними значеннями аргументу то його відносять до нейтрального типу.

Диференціально-різницеve рівняння (1.1) є рівнянням із запізненням аргументу, тому що величина старшої похідної в довільний момент часу t визначається через значення функції і молодших похідних в попередні моменти часу. Такі рівняння мають багату історію свого дослідження і знайшли численні застосування в різних прикладних науках та реальних сферах.

1.3. Постановка початкової задачі

Розглянемо диференціальне рівняння із запізненням

$$\dot{x}(t) = f(t, x(t), x(t - \tau)), \quad (1.2)$$

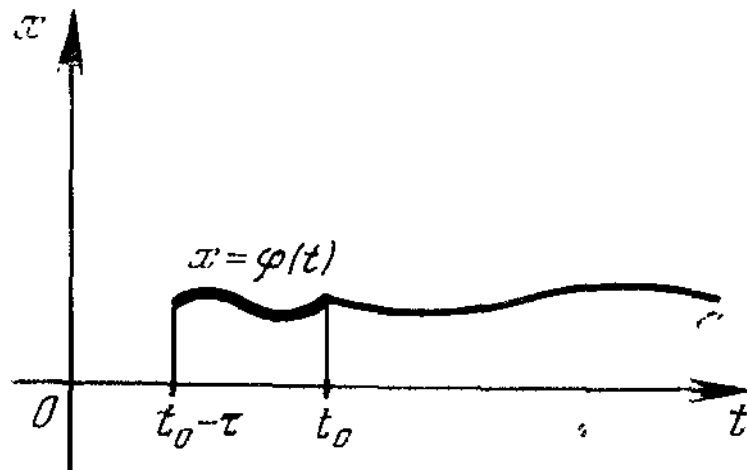


Рисунок 1.

де $\tau \geq 0$. Основна початкова задача при розв'язанні рівняння (1.2) полягає у знаходженні неперервного розв'язку $x(t)$ рівняння (1.2) при $t > t_0$ (t_0 - фіксований час) при умові, що $x(t) = \varphi(t)$ для $t_0 - \tau \leq t \leq t_0$, де $\varphi(t)$ -

задана неперервна функція, що називається початковою. Відрізок $[t_0 - \tau, t_0]$, на якому визначена початкова функція, називають початковою множиною і позначають E_{t_0} , а точка t_0 називається початковою.

Якщо запізнення τ в рівнянні (1.2) залежить від часу t ($\tau = \tau(t)$), то початкова задача задається так: знайти розв'язок рівняння (1.2) при $t > t_0$, якщо відома початкова функція $x(t) = \varphi(t)$ при $t \in E_{t_0}$

$$E_{t_0} = \{t_0\} \cup \{t - \tau(t) : t - \tau(t) < t_0 \text{ при } t \geq t_0\}. \quad (1.3)$$

Якщо у рівнянні (1.2) і в початкових умовах $x(t), f, \varphi(t)$ будуть векторними функціями, то ми матимемо задану початкову задачу для системи рівнянь.

Теорема 1. [7-8] *Якщо $\tau(t)$ неперервна і невід'ємна для $0 \leq t \leq t_0 + \mu$ ($\mu > 0$), f неперервна в деякому околі точки $(t_0, \varphi(t_0), \varphi(t_0 - \tau(t_0)))$ і задовольняє умову Ліпшиця по другому і третьому аргументу, функція $\varphi(t)$ неперервна на E_{t_0} , тоді існує єдиний розв'язок задачі (1.2-1.3) для $t_0 \leq t \leq t_0 + h$, де h – достатньо мале.*

Відзначимо, знаходження розв'язку початкової задачі в аналітичному вигляді можливо тільки для найпростіших ДРР, тому важливими є методи їх наближеного розв'язання.

1.4. Метод кроків

Розглянемо диференціальне рівняння із запізненням

$$\dot{x}(t) = f(t, x(t), x(t - \tau)), \quad (1.4)$$

де $\tau \geq 0, t - \tau \geq 0$, із заданою початковою функцією $x(t) = \varphi(t)$ при $t_0 - \tau \leq t \leq t_0$.

Потрібно знайти розв'язок рівняння (1.4) при $t \geq t_0$.

Для знаходження розв'язку (1.4) при $t \geq t_0$ застосовується метод послідовного інтегрування (метод кроків).

Суть методу в тому, що спочатку знаходимо розв'язок рівняння (1.4) для $t_0 \leq t \leq t_0 + \tau$, далі для $t_0 + \tau \leq t \leq t_0 + 2\tau$, і т. д. При цьому відзначимо, наприклад, що на відрізку $[t_0, t_0 + \tau]$ аргумент $t - \tau$ змінюється в межах

$t_0 - \tau \leq t - \tau \leq t_0$, тоді в рівнянні (1.4) на цьому відрізку замість $x(t - \tau)$ можна взяти початкову функцію $\varphi(t - \tau)$. Тоді отримаємо, що для знаходження розв'язку рівняння (1.4) на відрізку $t_0 \leq t \leq t_0 + \tau$ потрібно розв'язати звичайне диференціальне рівняння

$$\dot{x}(t) = f(t, x(t), \varphi(t - \tau)), \quad (1.5)$$

при $t_0 \leq t \leq t_0 + \tau$ з початковою умовою $x(t_0) = \varphi(t_0)$ (рис. 1.1).

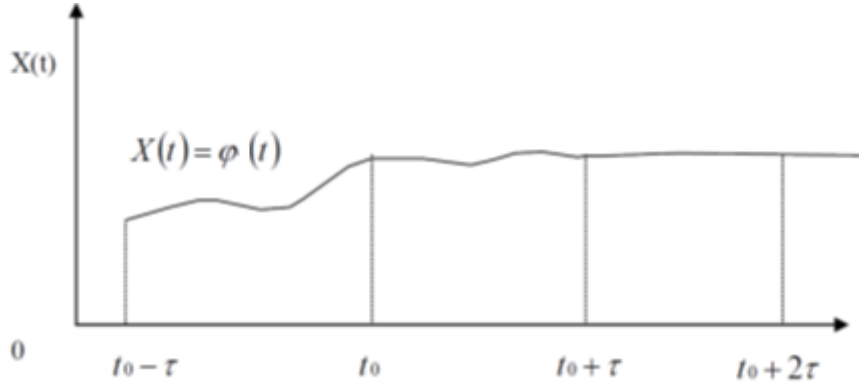


Рисунок 2.

Тепер, маючи розв'язок цієї початкової задачі у вигляді $x(t) = \varphi_1(t)$, можемо поставити задачу знаходження розв'язку на відрізку $t_0 + \tau \leq t \leq t_0 + 2\tau$ і т. д.

Отже, маємо:

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), \varphi(t - \tau)), \\ \text{при } t_0 \leq t \leq t_0 + \tau, x(t_0) &= \varphi(t_0), \end{aligned} \quad (1.6)$$

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), \varphi_1(t - \tau)), \\ \text{при } t_0 + \tau \leq t \leq t_0 + 2\tau, x(t_0 + \tau) &= \varphi_1(t_0 + \tau), \end{aligned} \quad (1.7)$$

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), \varphi_2(t - \tau)), \\ \text{при } t_0 + 2\tau \leq t \leq t_0 + 3\tau, x(t_0 + 2\tau) &= \varphi_2(t_0 + 2\tau), \end{aligned} \quad (1.8)$$

...

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), \varphi_n(t - \tau)), \\ \text{при } t_0 + n\tau \leq t \leq t_0 + (n + 1)\tau, x(t_0 + n\tau) &= \varphi_n(t_0 + n\tau), \end{aligned} \quad (1.9)$$

де $\varphi_i(t)$ є розв'язком заданої початкової задачі на відрізку $t_0 + (i - 1)\tau \leq t \leq t_0 + i\tau$, ($i = 1, 2, 3, \dots, n, \dots$).

Такий метод кроків розв'язування диференціального рівняння із запізненням (1.2) дозволяє визначити розв'язок $x(t)$ на деякому скінченному відрізку зміни аргументу t .

1.5. Гладкість розв'язків початкової задачі для диференціальних рівнянь запізнюючого і нейтрального типів

Відзначимо, що навіть у випадку диференційованості функцій φ, f як завгодно високого порядку, розв'язок початкової задачі для рівняння (1.2), взагалі кажучи, може мати розриви першого роду похідної порядку k в точці $t_0 + (k - 1)\tau$, але похідні нижчих порядків в цій точці вже будуть неперервними. Дійсно, в точці t_0 перша похідна $\dot{x}(t)$ може мати розрив першого роду, оскільки інтегруючи рівняння

$$\dot{x}(t) = f(t, x(t), \varphi(t - \tau)), t_0 \leq t \leq t_0 + \tau,$$

можемо задовольнити умову $x(t_0) = \varphi(t_0)$, але, не завжди можна задовольнити умову

$$\dot{x}(t_0 + 0) = \dot{\varphi}(t_0 - 0).$$

Тільки спеціально підбираючи початкову функцію $\varphi(t)$ можна забезпечити неперервність похідної розв'язку в точці t_0 . Для цього початкова функція $\varphi(t)$ повинна задовольняти умові

$$\dot{\varphi}(t_0 - 0) = f(t_0, \varphi(t_0), \varphi(t_0 - \tau)),$$

яка називається умовою склейки.

В точці $t_0 + \tau$ перша похідна розв'язку початкової задачі є неперервною. Дійсно, у рівнянні

$$\dot{x}(t) = f(t, x(t), x(t - \tau)),$$

права частина є неперервною функцією в точці $t_0 + \tau$, оскільки $x(t)$ неперервна в точці t_0 .

Однак друга похідна розв'язку початкової задачі

$$\ddot{x}(t) \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x(t)} \dot{x}(t) + \frac{\partial f}{\partial x(t-\tau)} \dot{x}(t-\tau),$$

в точці $t_0 + \tau$, взагалі кажучи, розривна, тому що похідна $\dot{x}(t-\tau)$ при $t = t_0 + \tau$, в загальному випадку, розривна. Але в точці $t = t_0 + 2\tau$ похідна розв'язку початкової задачі $\ddot{x}(t)$ неперервна, бо $\dot{x}(t-\tau)$ і $x(t-\tau)$ неперервні в точці $t = t_0 + 2\tau$.

Продовжуючи ці міркування дістаємо, що в точці $t_0 + (k-1)\tau$ похідна $x^{(k)}(t)$, взагалі кажучи є розривною, однак похідні нижчих порядків неперервні, припускаючи звичайно, що функція f є достатньо разів диференційованою.

Отже, розв'язок $x(t)$ початкової задачі для рівняння із запізненням при зростанні t згладжується.

Розглянемо тепер основну початкову задачу для диференціального рівняння нейтрального типу

$$\dot{x}(t) = f(t, x(t), x(t-\tau), \dot{x}(t-\tau)), \quad (1.10)$$

$$x(t) = \varphi(t) \text{ при } t_0 - \tau \leq t \leq t_0. \quad (1.11)$$

Виявляється, що розв'язки початкової задачі для рівнянь нейтрального типу не згладжуються. Дійсно, не тільки в точці t_0 ліва похідна розв'язку $\dot{\varphi}(t_0 - 0)$, взагалі кажучи, не дорівнює $\dot{x}(t_0 + 0)$, а тому розривна, але й в точці $t_0 + \tau$ $\dot{x}(t)$ буде розривна через розривність аргументу $\dot{x}(t-\tau)$ у правій частині (1.8) при $t = t_0 + \tau$. Аналогічні міркування демонструють, що розв'язок $x(t)$ початкової задачі, в загальному випадку є розривним при $t = t_0 + k\tau$ ($k = 0, 1, 2, \dots$).

РОЗДІЛ 2. НАБЛИЖЕНІ АЛГОРИТМИ РОЗВ'ЯЗУВАННЯ ПОЧАТКОВИХ ЗАДАЧ ДЛЯ ДИФЕРЕНЦІАЛЬНО-РІЗНИЦЕВИХ РІВНЯНЬ

2.1. Вступ

У багатьох прикладних областях застосування диференціальних рівнянь із запізненням є поширеною практикою. Однак відсутність ефективних алгоритмів та поширених пакетів прикладних програм для їх наближеного розв'язання робить таке застосування дещо обмеженим.

Відзначимо, що побудова наближених методів розв'язання початкових і крайових задач із запізненням в основному базується на адаптації відомих результатів для звичайних диференціальних рівнянь.

У роботах [14, 15] здійснено детальний огляд застосування методів Рунге-Кутти для розв'язування диференціальних рівнянь із запізненням. При цьому використовуються, як правило, методи кусково-сталої чи лінійної інтерполяції.

У роботах [16, 17] досліджуються схеми наближеної заміни як початкових так і крайових задач для диференціально-різницевих рівнянь послідовністю спеціальних систем диференціальних рівнянь без запізнення для розв'язання яких можна використовувати відомі різницеві схеми. Однак такий підхід виявився дещо складним через високу розмірність апроксимуючої системи звичайних диференціальних рівнянь.

Одновимірні методи предикатор-корелятор [18, 19] дозволяють одержати наближений розв'язок диференціально-різницевих рівнянь за допомогою деяких апостеріорних поправок.

Розвиток ефективних θ різницевих схем для диференціально-різницевих рівнянь досліджено в роботах [20-21], аналіз яких наведено в наступному пункті.

2.2. Сім'я θ різницевих схем

Будемо розглядати початкову задачу для диференціально-різницевого рівняння запізнюючого типу

$$\dot{x}(t) = f(t, x(t), x(t - \tau)), \quad (2.1)$$

де $t \in R, x(t) \in R^n, \tau > 0$.

На початковій множині $E_{t_0} = [t_0; t_0 - \tau]$ визначимо неперервну початкову функцію $\varphi(t): E_{t_0} \rightarrow R^n$.

Під розв'язком диференціально-різницевого рівняння (2.1) будемо розуміти функцію $x(t)$, яка при $t \geq t_0$ задовольняє рівняння (2.1) і співпадає з початковою функцією на множині

$$x(t) = \varphi(t), t \in E_{t_0}. \quad (2.2)$$

Розглянемо рівномірну сітку

$$\omega = \left\{ t_n = nh, n = -m, \dots, -1, 0, 1, \dots, k, kh = T, m = \frac{\tau}{h} \right\}.$$

Інтегруючи рівняння (2.1) від t_n до t_{n+1} маємо

$$x(t_{n+1}) = x(t_n) + \int_{t_n}^{t_{n+1}} f(t, x(t), x(t - \tau)) dt. \quad (2.3)$$

Замінюючи інтеграл у рівності (2.3) квадратурною формулою, дістанемо:

$$\int_{t_n}^{t_{n+1}} f(t, x(t), x(t - \tau)) dt = h[(1 - \theta)f(t_n, y_n, v_n) + \theta f(t_{n+1}, y_{n+1}, v_{n+1})], \text{ де } \theta \in$$

$[0; 1], v_n$ – апроксимація $x(t - \tau)$ при $t = t_n$:

$$v_n = \begin{cases} \varphi(t_0 - \tau), & \text{при } n < m, \\ y_{n-m}, & \text{при } n \geq m. \end{cases}$$

Тоді одержуємо [21] вигляд θ різницевої схеми для початкової задачі (2.1)-(2.2)

$$y_{n+1} = y_n + h[(1 - \theta)f(t_n, y_n, v_n) + \theta f(t_{n+1}, y_{n+1}, v_{n+1})]. \quad (2.4)$$

Якщо $\theta = 0$, то дістаємо

$$y_{n+1} = y_n + hf(t_n, y_n, v_n), \quad (2.5)$$

узагальнення явної різницевої схеми Ейлера. У випадку $\theta = 1$ дістали узагальнення неявної різницевої схеми Ейлера

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}, v_{n+1}), \quad (2.6)$$

а при $\theta = \frac{1}{2}$ одержали узагальнення різницевої схеми трапецій (неявна різницева схема Адамса другого порядку) для інтегрування диференціальних рівнянь із запізненням.

Зауваження 1. [21] Якщо запізнення τ та крок h не є раціонально залежними, то апроксимація $x(t - \tau)$ в точці $t = t_n$ здійснюється за правилом

$$x(t_n - \tau) \approx v_n = \begin{cases} \varphi(t_n - \tau), & \text{якщо } t_n - \tau < 0, \\ v_n^i, & \text{якщо } t_n - \tau \geq 0. \end{cases}$$

Значення v_n^i обчислюється за таким алгоритмом:

- 1) знаходимо i такий, що $t_k \leq t_i - \tau < t_{k+1}$;
- 2) значення v_n^i знаходимо використовуючи лінійну інтерполяцію за точками $(t_i, x_i), (t_{i+1}, x_{i+1})$:

$$v_n^i = \frac{t_n - \tau - t_i}{h} y_{i+1} + \frac{t_{i+1} - t_n + \tau}{h} y_i.$$

Теорема 2. [21] Якщо розв'язок початкової задачі (2.1)-(2.2) двічі неперервно диференційована функція, то метод (2.4) збіжний з першим порядком малості по h .

2.3. Узагальнення θ різницевої схеми на рівняння нейтрального типу

Розглянемо узагальнення θ різницевої схеми на випадок початкової задачі для диференціального різницевого рівняння нейтрального типу:

$$\dot{x}(t) = f(t, x(t), x(t - \tau), \dot{x}(t - \tau)), \quad (2.7)$$

$$x(t) = \varphi(t), \dot{x}(t) = \dot{\varphi}(t), t \in [t_0 - \tau; t_0]. \quad (2.8)$$

Розглянемо рівномірну сітку

$$\omega = \left\{ t_n = nh, n = -m, \dots, -1, 0, 1, \dots, k, kh = T, m = \frac{\tau}{h} \right\}.$$

Інтегруючи рівняння (2.7) від t_n до t_{n+1} маємо

$$x(t_{n+1}) = x(t_n) + \int_{t_n}^{t_{n+1}} f(t, x(t), x(t - \tau), \dot{x}(t - \tau)) dt. \quad (2.9)$$

Замінюючи інтеграл у рівності (2.9) квадратурною формулою, дістанемо:

$$\int_{t_n}^{t_{n+1}} f(t, x(t), x(t-\tau), \dot{x}(t-\tau)) dt = h[(1-\theta)f(t_n, y_n, v_n) + \theta f(t_{n+1}, y_{n+1}, v_{n+1})],$$

де $\theta \in [0; 1]$, v_n – апроксимація $x(t-\tau)$, w_n – апроксимація $\dot{x}(t-\tau)$ при $t = t_n$:

$$v_n = \begin{cases} \varphi(t_0 - \tau), & \text{при } n < m, \\ y_{n-m}, & \text{при } n \geq m, \end{cases} \quad w_n = \begin{cases} \dot{\varphi}(t_0 - \tau), & \text{при } n < m, \\ \frac{y_{n-m+1} - y_{n-m}}{h}, & \text{при } n \geq m. \end{cases}$$

Тоді одержуємо узагальнення θ різницевої схеми для початкової задачі (2.7-2.8)

$$y_{n+1} = y_n + h[(1-\theta)f(t_n, y_n, v_n) + \theta f(t_{n+1}, y_{n+1}, v_{n+1})]. \quad (2.10)$$

Зауваження 2. Якщо запізнення τ та крок h не є раціонально залежними, то апроксимацію $x(t_n - \tau)$ та $\dot{x}(t_n - \tau)$ здійснюємо за схемою

$$x(t_n - \tau) \approx v_n = \begin{cases} \varphi(t_n - \tau), & \text{якщо } t_n - \tau < 0, \\ v_n^i, & \text{якщо } t_n - \tau \geq 0, \end{cases}$$

$$\dot{x}(t_n - \tau) \approx w_n = \begin{cases} \dot{\varphi}(t_n - \tau), & \text{якщо } t_n - \tau < 0, \\ w_n^i, & \text{якщо } t_n - \tau \geq 0. \end{cases}$$

Значення v_n^j та w_n^j обчислюємо за алгоритмом

- 1) Знаходимо індекс j такий, що $t_j \leq t_n - \tau \leq t_{j+1}$.
- 2) Значення v_n^j та w_n^j знаходимо, використовуючи лінійну інтерполяцію

$$v_n^j = \frac{t_n - \tau - t_j}{h} y_{j+1} + \frac{t_{j+1} - t_n + \tau}{h} y_j,$$

$$w_n^j = \frac{y_{j+1} - y_j}{h}.$$

РОЗДІЛ 3. АЛГОРИТМ АВТОМАТИЗАЦІЇ МЕТОДУ КРОКІВ

3.1. Схема алгоритму

Розглядаємо початкову задачу для диференціально-різницевого рівняння запізнюючого типу

$$\dot{x}(t) = f(t, x(t), x(t - \tau)), \quad (3.1)$$

$$x(t) = \varphi(t), t \in E_{t_0}. \quad (3.2)$$

де $t \in R, x(t) \in R^n, \tau > 0$.

Метод послідовного інтегрування (метод кроків) полягає в тому, що на початку шукаємо розв'язок початкової задачі (3.1)-(3.2) при $t_0 \leq t \leq t_0 + \tau$, далі при $t_0 + \tau \leq t \leq t_0 + 2\tau$, і т. д.

1. На першому відрізку $t \in [t_0, t_0 + \tau]$ маємо задачу Коші:

$$\dot{x}(t) = f(t, x(t), z(t)), \quad (3.3)$$

$$z(t) = x(t - \tau), x(s) = \varphi(s), s \in [t_0 - \tau; t_0]. \quad (3.4)$$

Наближений розв'язок задачі (3.3)-(3.4) можна знайти за допомогою деякої різницевої схеми, оскільки $z(t)$ відома функція, як множину точок

$$x_i = x(t_i), t_i = t_0 + ih, i = 0, \dots, n, h = \frac{\tau}{n}. \quad (3.5)$$

2. На наступному відрізку $[t_0 + \tau, t_0 + 2\tau]$ розглянемо задачу Коші:

$$\dot{x}(t) = f(t, x(t), z_1(t)), \quad (3.6)$$

$$z_1(t) = x(t - \tau), x(s) = \varphi_1(s), s \in [t_0 - \tau; t_0], \quad (3.7)$$

де $\varphi_1(s)$ деяка апроксимація $x(t - \tau)$ за знайденими значеннями (3.5) на попередньому кроці.

Наближений розв'язок задачі (3.6)-(3.7) знаходимо за допомогою деякої різницевої схеми, оскільки $z_1(t)$ відома функція, як множину точок

$$x_i = x(t_i), t_i = t_0 + ih, i = 0, \dots, n, h = \frac{\tau}{n}. \quad (3.8)$$

3. Продовжуючи цей процес ми побудуємо різницеву апроксимацію для початкових задач із запізненням, об'єднуючи деякий метод інтегрування задач Коші для звичайних диференціальних рівнянь та методу апроксимації сіткових функцій.

3.2. Реалізація алгоритму

Для ефективної реалізації наведеного алгоритму необхідно на кожному кроці на основі знайдених наближених значень $x(t_i)$, на сітці

$$\Delta: t_i = t_k + ih, i = 0, 1, \dots, n, h = \frac{\tau}{n}, n \in \mathbb{N} \text{ на кроці } [t_k; t_{k+1}],$$

побудувати інтерполяційну функцію $\varphi_k(t)$, яка буде виступати як нова початкова функція для наступного інтервала знаходження розв'язку.

Наведемо два підходи до побудови функції $\varphi_k(t)$.

3.2.1. Лінійна інтерполяція

Розглянемо найбільш простий і швидкий метод знаходження інтерполяційної функцію $\varphi_k(t)$, в якому задані вузлові точки $(t_i; y_i)$, $i = 0, 1, \dots, n$ з'єднуються прямими. Геометрично це представляє ламану, що проходить через ці точки.

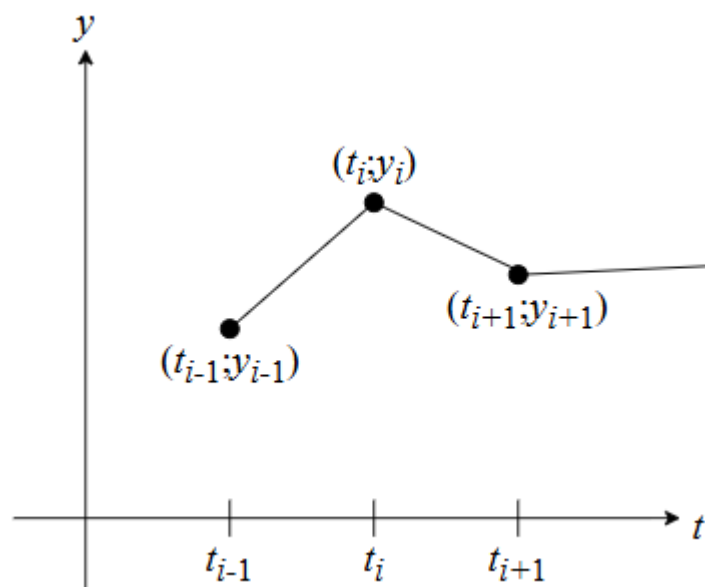


Рисунок 3.

Очевидно, що в цьому випадку маємо функцію $\varphi_k(t)$ як інтерполяційний сплайн першого порядку.

Означення. Функція $S(t) = S_1(t)$ називається інтерполяційним сплайном першого порядку для функції $f(t)$ на сітці Δ , якщо

$$1) S_1(t) \in P_1(t), \quad t \in [t_i; t_{i+1}],$$

- 2) $S_1(t) \in C[a; b]$,
- 3) $S_1(t) = f(t_i), i = 0, 1, \dots, n$,

де $P_1(t)$ – множина многочленів першого порядку.

Якщо $t \in [t_i; t_{i+1}]$, то для сплайна $S_1(x)$ на цьому проміжку маємо зображення

$$S_1^i(t) = \frac{t_{i+1}-t}{h}y_i + \frac{t-t_i}{h}y_{i+1}. \quad (3.9)$$

Для оцінки точності наближення сплайна $S_1(x)$ введемо величину

$$\omega(f, \delta) = \max_{\substack{t', t'' \in [a, b] \\ |t' - t''| < \delta}} |f(t') - f(t'')|,$$

яка називається модулем неперервності функції $f(t)$.

Лема [22]. Якщо $f(t) \in C[a, b]$, то $\max |S_1(t) - f(t)| \leq \omega(f, h)$.

Наслідок 1. Так як для неперервної функції $f(t)$ за теоремою Кантора $\omega(f, h) \rightarrow 0$ при $h \rightarrow 0$, то зменшуючи крок сітки h можна досягнути потрібної точності наближення.

Наслідок 2. Якщо потрібно знайти значення сплайна $S_1(\xi)$, $\xi \in [a; b]$, то потрібно:

- 1) знайти інтервал $\xi \in [t_i; t_{i+1}]$,
- 2) обчислити $S_1(\xi)$ згідно з формулою (3.9).

3.2.2. Кубічна інтерполяція

Будемо шукати інтерполяційну функцію $\varphi_k(t)$ у вигляді кубічного інтерполяційного сплайну.

Алгоритм побудови кубічного сплайну наведено в наступних пунктах.

Відзначимо, що ми використовуємо його в такому вигляді:

$$S(f; x) = f_i(1 - t^2)(1 + 2t) + f_{i+1}t^2(3 - 2t) + m_i ht(1 - t)^2 - m_{i+1} ht^2(1 - t), \quad (3.10)$$

де $t = \frac{x-x_i}{h}$, $m_i = \dot{S}(f; x_i)$.

Для однозначного знаходження кубічного сплайну ми використовуємо додаткові крайові умови

$$\dot{S}(f; a) = \dot{y}(a), \quad \dot{S}(f; b) = \dot{y}(b),$$

де похідні знаходяться із рівняння (3.1).

Точність апроксимації кубічних сплайнів виражає наступна лема.

Лема 2. [22] *Якщо $f(t) \in C^2[a; b]$, тоді справджується оцінка*

$$|f^{(i)}(t) - S_3^{(i)}(t)| \leq K_i h^k \omega(f, h), \quad i = 0, 1, 2, \quad K_2 = K_1 = 8K_0 = 5.$$

Наслідок 3. Якщо потрібно знайти значення сплайна $S_3(\xi)$, $\xi \in [a; b]$, то необхідно:

- 1) знайти інтервал $\xi \in [t_i; t_{i+1}]$,
- 2) обчислити $S_3(\xi)$ згідно з формулою (3.10).

3.3. Кубічні сплайни

3.3.1. Вступ

Апарат сплайн-функцій — це новий метод наближення функцій, який активно розвивається та має багато важливих застосувань в різноманітних прикладних задачах. Дослідження різних класів сплайн-функцій розпочалося в минулому столітті при вивченні задач інтерполювання для складних кривих з прикладної геометрії. Поширеним апаратом для вивчення таких задач були класичні інтерполяційні многочлени Лагранжа та Ньютона. У випадку великого числа вузлів інтерполювання практична побудова цих многочленів є складною задачею і крім того зростають похибки інтерполювання.

Апарат сплайн-наближень дозволяє усунути проблеми інтерполяції многочленами. Відзначимо основними перевагами такого підходу [22-24]:

- обчислювальна стійкість відносно локальних збурень, тобто поведінка сплайна в околі деякої точки не впливає на його поведінку в цілому, як, наприклад, це має місце при інтерполяції многочленами;
- апарат сплайн-інтерполяції володіє доброю збіжністю на відміну від многочленної інтерполяції;
- наявність простих алгоритмів для реалізації сплайн-функцій на ЕОМ. Зокрема, для побудови інтерполяційних кубічних сплайнів необхідно розв'язання системи лінійних алгебраїчних рівнянь з трьохдіагональною матрицею.

3.3.2. Алгоритм побудови

Нехай на відрізку $[a, b]$ у вузлах сітки Δ : $a = x_0 < x_1 < \dots < x_N = b$ задані значення деякої функції $f_i = f(x_i)$, $i = 0, 1, \dots, N$.

Означення 1. [22-24] Кубічним інтерполяційним сплайном на сітці Δ для функції $f(x)$ називається функція $S(x)$, що задовольняє умови:

- 1) $S(x) \in P_3(x)$, $x \in [x_{j-1}, x_j]$, $j = 1, 2, \dots, N$, де $P_3(x)$ - множина многочленів степеня 3,
- 2) $S(x) \in C^2[a, b]$,
- 3) $S(x_j) = f(x_j)$, $j = 0, 1, \dots, N$.

Перейдемо до безпосереднього описання алгоритму побудови інтерполяційних кубічних сплайнів.

Кубічний сплайн $S(x)$ на кожному із відрізків $[x_i, x_{i+1}]$ є кубічним многочленом, що має чотири коефіцієнти і тому для його визначення на всьому відрізку $[a, b]$ необхідно визначити $4N$ коефіцієнти. Із умови $S(x) \in C^2[a, b]$ дістаємо неперервність сплайну та його похідних $S^r(x)$, $r = 0, 1, 2$ в усіх внутрішніх вузлах x_i , $i = 1, 2, \dots, N - 1$, сітки Δ , що дозволяє отримати $4N - 2$ рівнянь. Дві додаткові умови зазвичай отримують задаючи обмеження на значення сплайну або його похідні на кінцях відрізка $[a, b]$ у вигляді крайових умов. Найчастіше використовується такі типи крайових умов:

$$\text{I. } \dot{S}(f; a) = \dot{f}(a), \quad \dot{S}(f; b) = \dot{f}(b).$$

$$\text{II. } \ddot{S}(f; a) = \ddot{f}(a), \quad \ddot{S}(f; b) = \ddot{f}(b).$$

$$\text{III. } S^{(r)}(f; a) = S^{(r)}(f; b), \quad r = 1, 2.$$

$$\text{IV. } \dddot{S}(f; x_p + 0) = \dddot{S}(f; x_p - 0), \quad p = 1, \dots, N - 1.$$

Крайові умови III називаються періодичними. Звісно, що їх треба вимагати тоді, коли функція $f(x)$ є періодичною з періодом $b - a$.

Перейдемо до опису алгоритму побудови інтерполяційних кубічних сплайнів з використанням крайових умов першого типу.

Введемо позначення:

$$\dot{S}(f; x_i) = m_i, \quad i = 0, 1, \dots, N. \quad (3.11)$$

Нехай у вузлах сітки Δ задані значення функції $f(x)$ і її похідної $\dot{f}(x)$:

$$f_i = f(x_i), \quad \dot{f}_i = \dot{f}(x_i), \quad i = 0, 1, \dots, N.$$

Із означення сплайна на кожному із проміжків $[x_i; x_{i+1}]$ маємо кубічний многочлен

$$S(x) = a_{i0} + a_{i1}(x - x_i) + a_{i2}(x - x_i)^2 + a_{i3}(x - x_i)^3.$$

Будемо вимагати щоб виконувалися умови

$$S(x_i) = f_i, \quad \dot{S}(x_i) = \dot{f}_i, \quad i = 0, 1, \dots, N.$$

Очевидно, в цьому випадку $S(x) \in C^1[a, b]$. Використовуючи умови інтерполяції, для знаходження коефіцієнтів $a_{i\alpha}$, $\alpha = 0, 1, 2, 3$, при $i = 0, 1, \dots, N$ маємо таку систему рівнянь

$$\begin{cases} S(x_i) = f_i, \\ S(x_{i+1}) = f_{i+1}, \\ \dot{S}(x_i) = \dot{f}_i, \\ \dot{S}(x_{i+1}) = \dot{f}_{i+1}. \end{cases}$$

Розв'язавши цю систему рівнянь, отримаємо на проміжку $[x_i; x_{i+1}]$

$$S(x) = \varphi_1(t)f_i + \varphi_2(t)f_{i+1} + \varphi_3(t)h_i\dot{f}_i + \varphi_4(t)h_i\dot{f}_{i+1}, \quad (3.12)$$

причому

$$\varphi_1(t) = (1 - t)^2(1 + 2t),$$

$$\varphi_2(t) = t^2(3 - 2t),$$

$$\varphi_3(t) = t(1 - t)^3,$$

$$\varphi_4(t) = -t^2(1 - t),$$

$$h_i = x_{i+1} - x_i,$$

$$t = (x - x_i)/h_i.$$

Кубічний сплайн, представлений в такому вигляді на кожному із проміжків $[x_i; x_{i+1}]$, буде неперервно диференційованою функцією всюди на $[a, b]$. Підберемо значення m_i таким чином, щоб друга похідна також була неперервною. Оскільки,

$$\ddot{S}(f; x) = \frac{(f_{i+1} - f_i)(6 - 12t)}{h_i^2} + \frac{m_i(-4 + 6t)}{h_i} + \frac{m_{i+1}(-2 + 6t)}{h_i} \quad (3.13)$$

і

$$\ddot{S}(f; x_i + 0) = \frac{6(f_{i+1} - f_i)}{h_i^2} - \frac{4m_i}{h_i} - \frac{2m_{i+1}}{h_i}, \quad (3.14)$$

$$\ddot{S}(f; x_i - 0) = \frac{-6(f_i - f_{i-1})}{h_{i-1}^2} + \frac{2m_{i-1}}{h_{i-1}} + \frac{4m_i}{h_{i-1}},$$

то умова неперервності другої похідної $\ddot{S}(f; x_i + 0) = \ddot{S}(f; x_i - 0)$ в точках $x_i, i = 1, 2, \dots, N - 1$, матиме вигляд

$$\lambda_i m_{i-1} + 2m_i + \mu_i m_{i+1} = 3(\mu_i \frac{f_{i+1} - f_i}{h_i} + \lambda_i \frac{f_i - f_{i-1}}{h_{i-1}}), \quad (3.15)$$

причому $\mu_i = h_{i-1}(h_{i-1} + h_i)^{-1}, \lambda_i = 1 - \mu_i$.

До рівняння (3.15) треба ще додати рівняння, які отримуються за рахунок крайових умов. Тоді, маємо систему із $N+1$ рівнянь для знаходження $N+1$ невідомих $m_i, i = 0, 1, \dots, N$.

Якщо крайові умови є у вигляді I та II видів, то

$$\begin{cases} 2m_0 + \mu_0^* m_1 = c_0^*, \\ \lambda_i m_{i-1} + 2m_i + \mu_i m_{i+1} = c_i, \quad i = 1, 2, \dots, N - 1, \\ \lambda_N^* m_{N-1} + 2m_N = c_N^*, \end{cases} \quad (3.16)$$

причому $c_i = 3(\mu_i \frac{f_{i+1} - f_i}{h_i} + \lambda_i \frac{f_i - f_{i-1}}{h_{i-1}})$.

Для крайової умови I типу маємо

$$\mu_0^* = \lambda_N^* = 0, \quad c_0^* = 2\dot{f}_0, \quad c_N^* = 2\dot{f}_N,$$

а для умови II типу згідно із (3.4а)

$$\mu_0^* = \lambda_N^* = 1, \quad c_0^* = 3 \frac{f_1 - f_0}{h_0} - \frac{h_0}{2} \ddot{f}_0, \quad c_N^* = 3 \frac{f_N - f_{N-1}}{h_{N-1}} + \frac{h_{N-1}}{2} \ddot{f}_N.$$

Щоб одержати систему рівнянь у випадку, коли крайові умови IV типу визначимо

$$\ddot{\ddot{S}}(f; x) = \frac{6}{h_i^2} \left(m_{i+1} + m_i - 2 \frac{f_{i+1} - f_i}{h_i} \right).$$

Звідси отримаємо умови, щоб функція $\ddot{\ddot{S}}(f; x)$ була неперервною в точках $x_p, p = 1, 2, \dots, N-1$:

$$\frac{1}{h_p^2} (m_{p+1} + m_p - 2 \frac{f_{p+1} - f_p}{h_p}) = \frac{1}{h_{p-1}^2} (m_p + m_{p-1} - 2 \frac{f_p - f_{p-1}}{h_{p-1}}).$$

Із цих умов ми можемо отримати рівняння, які відповідають умовам IV типу:

$$m_0 + (1 - \gamma_0^2) m_1 - \gamma_0^2 m_2 = 2 \left(\frac{f_1 - f_0}{h_0} - \gamma_0^2 \frac{f_2 - f_1}{h_1} \right), \quad (3.17)$$

$$- \gamma_N^2 m_{N-2} + (1 - \gamma_N^2) m_{N-1} + m_N = 2 \left(\frac{f_N - f_{N-1}}{h_{N-1}} - \gamma_N^2 \frac{f_{N-1} - f_{N-2}}{h_{N-2}} \right),$$

де $\gamma_0 = h_0/h_1$, $\gamma_N = h_{N-1}/h_{N-2}$. Якщо виключимо змінні m_0, m_N із рівнянь (3.15) та (3.17), то отримаємо таку систему:

$$\begin{cases} (1 + \gamma_0)m_1 + \gamma_0 m_2 = c_1^*, \\ \lambda_i m_{i-1} + 2m_i + \mu_i m_{i+1} = c_i, \quad i = 2, 3, \dots, N-2, \\ \gamma_N m_{N-2} + (1 + \gamma_N)m_{N-1} = c_{N-1}^*, \end{cases} \quad (3.18)$$

де

$$c_1^* = \frac{1}{3}c_1 + 2\gamma_0 \frac{f_2 - f_1}{h_1}, \quad c_{N-1}^* = \frac{1}{3}c_{N-1} + 2\gamma_N \frac{f_{N-1} - f_{N-2}}{h_{N-2}}.$$

Якщо $f(x)$ є періодичною функцією, то продовжуємо її періодичним чином, припускаючи що

$$f_0 = f_N, \quad f_{N+1} = f_1, \quad m_0 = m_N, \quad m_1 = m_{N+1}, \quad h_N = h_0,$$

можемо записати крайову умову в точці x_N . Система для визначення коефіцієнтів m_i буде мати такий вигляд:

$$\begin{cases} 2m_1 + \mu_1 m_2 + \lambda_1 m_N = c_1, \\ \lambda_i m_{i-1} + 2m_i + \mu_i m_{i+1} = c_i, \quad i = 2, \dots, N-1, \\ \mu_N m_1 + \lambda_N m_{N-1} + 2m_N = c_N. \end{cases} \quad (3.19)$$

Отже для побудови інтерполяційного кубічного сплайну згідно формули (3.12) необхідне знайти коефіцієнти m_i . Для крайових умов типів I та II вони знаходяться із системи (3.16), для умови IV типу із рівнянь (3.17) та (3.18), а у випадку періодичності із рівняння (3.19), припускаючи, що $m_0 = m_N$.

Матриці у всіх чотирьох випадках є матрицями в яких головна діагональ є переважаючою. Такі матриці невироджені і тому ці системи мають розв'язок, причому цей розв'язок – єдиний.

Теорема 1. [22] Інтерполяційний кубічний сплайн $S(x)$, однієї при виконання із чотирьох крайових умов типу I-IV існує і єдиний.

Розв'язки системи рівнянь для знаходження коефіцієнтів m_i знаходяться за допомогою методу прогонки.

3.4. Метод прогонки

Означення 2. Квадратна матриця $A = [a_{ij}]$ порядку n називається матрицею з переважаючою діагоналлю, якщо виконується умова:

$$|a_{ii}| - \sum_{j \neq i} |a_{ij}| = r_i > 0, \quad i, j = 1, 2, \dots, n. \quad (3.20)$$

Теорема 2. [25] Матриця з переважаючою діагоналлю є невиродженою.

Теорема 3. [25] Якщо матриця A є матрицею з переважаючою діагоналлю, то справедливою є оцінка

$$\|A^{-1}\| \leq \max_{1 \leq i \leq N} \frac{1}{r_i}.$$

Ряд задач теорії сплайнів вимагають знаходження розв'язків системи лінійних рівнянь $Ay = d$.

Величина $\mu(A) = \|A^{-1}\| \|A\|$ називається мірою (числом) обумовленості системи або матриці. Системи рівнянь і матриці з малим числом обумовленості називають добре обумовленими.

Якщо матриця A є матрицею з переважаючою діагоналлю, то із теореми 3 впливає оцінка

$$\mu(A) \leq \|A\| \max_{1 \leq i \leq N} \frac{1}{r_i},$$

де r_i визначається формулами (3.20). Очевидно, якщо величина $(\max_i |a_{ii}|)(\min_i r_i)^{-1}$ невелика, то A – добре обумовлена матриця.

Нехай в нас є система рівнянь записана у матричному вигляді

$$\begin{bmatrix} a_1 & b_1 & 0 & \dots & 0 & 0 & c_1 \\ c_2 & a_2 & b_2 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & c_{N-1} & a_{N-1} & b_{N-1} \\ b_N & 0 & 0 & \dots & 0 & c_N & a_N \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_{N-1} \\ y_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \dots \\ d_{N-1} \\ d_N \end{bmatrix}. \quad (3.21)$$

1) Розглянемо випадок, коли $c_1 = b_N = 0$. Розв’язок системи будемо шукати у вигляді

$$y_i = v_i y_{i+1} + u_i, \quad i = 1, 2, \dots, N-1. \quad (3.22)$$

Використовуючи тепер вираз для y_{i-1} із (3.22), виключимо ці змінну з i -того рівняння системи. Отримаємо:

$$(a_i + c_i v_{i-1}) y_i + b_i y_{i+1} = d_i - c_i u_{i-1}.$$

Порівнюючи це співвідношення з (3.22), виведемо рекурентні співвідношення для коефіцієнтів прогонки v_i, u_i (пряма прогонка):

$$v_0 = u_0 = 0, \\ v_i = -\frac{b_i}{a_i + c_i v_{i-1}}, \quad u_i = \frac{d_i - c_i u_{i-1}}{a_i + c_i v_{i-1}}, \quad i = 1, 2, \dots, N. \quad (3.23)$$

Очевидно, $y_N = u_N$. Всі інші коефіцієнти знаходимо згідно формул (3.22) (зворотна прогонка).

2) Нехай c_1 і b_N не дорівнюють 0. Будемо шукати y_i в такому вигляді

$$y_i = v_i y_{i+1} + w_i y_N + u_i, \quad i = 1, 2, \dots, N-1. \quad (3.24)$$

Підставимо вираз для y_{i-1} в i -те рівняння системи і отримаємо

$$(a_i + c_i v_{i-1})y_i + b_i y_{i+1} + c_i w_i y_N = d_N - c_i u_{i-1}.$$

Звідси отримуємо формули (13) для коефіцієнтів $v_i, u_i, i = 1, 2, \dots, N-1$.

Величини w_i знаходимо із таких співвідношень

$$w_0 = 1, \quad w_i = -\frac{c_i w_{i-1}}{a_i + c_i v_{i-1}}, \quad i = 1, 2, \dots, N-1. \quad (3.25)$$

Тепер виразимо всі y_i через y_N у такому вигляді

$$y_i = s_i y_N + t_i, \quad i = 1, 2, \dots, N-1. \quad (3.26)$$

Підставляючи вирази для y_{i+1} в (3.14), отримаємо

$$y_i = (v_i s_{i+1} + w_i) y_N + (v_i t_{i+1} + u_i).$$

Порівнюючи ці співвідношення із (3.16), можемо знайти рекурентні формули для величин s_i, t_i :

$$s_N = 1, \quad t_N = 0, \quad s_i = v_i s_{i+1} + w_i, \quad t_i = v_i t_{i+1} + u_i, \quad i = N-1, \dots, 1. \quad (3.27)$$

Підставимо y_1 та y_{N-1} із (3.26) в останнє рівняння системи (3.21) і знайдемо y_N

$$y_N = \frac{d_N - b_N t_1 - c_N t_{N-1}}{a_N + b_N s_1 + c_N s_{N-1}}, \quad (3.28)$$

після чого розв'язок знаходиться за формулами (3.16). Всього потрібно виконати $14N$ арифметичних операцій.

РОЗДІЛ 4. МОДЕЛЮВАННЯ ПОЧАТКОВИХ ЗАДАЧ НА ЕОМ

4.1. Технології для створення прикладного додатку

Для автоматизації процесу наближеного розв'язання початкових задач для диференціально-різницевих рівнянь за допомогою числових алгоритмів, описаних у роботі, створено прикладний програмний додаток. Розробка додатка здійснювалася із застосуванням мови програмування Python та фреймворку Django.

4.1.1. Python [26]

Python — це багатифункціональна, високорівнева мова програмування з динамічною типізацією, інтерпретацією, інтерактивністю та підтримкою об'єктно-орієнтованого підходу. Вона спрямована на підвищення ефективності розробки, зручність читання коду та забезпечення простого імпорту модулів, написаних на ній.

Основні переваги Python:

- Відкритий вихідний код: Python можна вільно використовувати, змінювати та розповсюджувати.
- Зрозумілий синтаксис: Простота у вивченні завдяки лаконічній структурі та мінімальній кількості ключових слів.
- Читабельний код: Код на Python легкий для розуміння, що полегшує підтримку та розвиток програм.
- Бібліотеки та сумісність: Python має безліч бібліотек, які легко імпортуються та працюють на Unix, Windows і macOS.
- Інтерактивний режим: Дає можливість тестувати й відлагоджувати код у процесі його написання.
- Робота з графічним інтерфейсом: Можливість створювати програми з візуальним інтерфейсом. Гнучкість програмування: Підтримка процедурного й об'єктно-орієнтованого стилю дає змогу вирішувати різноманітні завдання.
- Автоматичне управління пам'яттю: Відсутність вказівників і автоматичне виділення пам'яті спрощують роботу з ресурсами.

Python використовується у багатьох галузях: Data Science, розробка мобільних і веб-додатків, штучний інтелект, машинне навчання, обробка зображень та розробка програмного забезпечення.

Програмісти обирають Python з багатьох причин, зокрема:

- Висока продуктивність: Інтерпретована природа мови забезпечує швидкий цикл редагування, тестування й налагодження.
- Зручне налагодження: При виявленні помилок інтерпретатор створює винятки, а налагоджувач дозволяє перевіряти змінні, встановлювати точки зупинки й поетапно виконувати код.
- Розробка клієнт-серверних додатків: Використання фреймворків, таких як Flask і Django, для створення серверної частини забезпечує ефективний поділ завдань між клієнтом і сервером.

4.1.2. Фреймворк Django [27, 28]

Фреймворк Django – це високорівневий відкритий Python-фреймворк для створення вебсайтів.

У Django реалізовано принцип DRY ("Don't repeat yourself"), що дозволяє значно скоротити час розробки вебсайтів. Це означає, що під час роботи з Django не потрібно багаторазово дублювати один і той самий код. Фреймворк забезпечує можливість створення сайтів із готових компонентів, подібно до складання фортеці з конструктора Lego.

Особливості початкової розробки Django суттєво вплинули на його архітектуру: він надає інструменти, які дозволяють швидко створювати інформаційні вебсайти. Вбудований модуль управління вмістом звільняє розробників від необхідності самостійно розробляти адміністративну частину сайту. Цей модуль забезпечує функціонал для додавання, редагування чи видалення об'єктів сайту, а також дозволяє керувати користувачами та їхніми групами з призначенням відповідних прав.

Архітектура Django схожа на архітектуру MVC («Модель-Вигляд-Контролер»). Проте є певні відмінності: те, що в класичній моделі MVC

називають «контролером», в Django називають «view» (вигляд), а те що мало б бути «виглядом», називається «template» (шаблон). Таким чином, MVC модель у Django називається MVT («Модель-Вигляд-Шаблон»).

Крім вбудованої адміністративної панелі джанго має багато можливостей, які полегшують веброзробку, зокрема:

- django має багато вбудованих функцій, які розробник може використовувати та імпортувати замість того, щоб писати власний код
- фреймворк використовує мову Python, що робить його легким у вивченні та використанні;
- django дозволяє легко працювати з базами даних;
- цей інструмент полегшує обробку запитів від користувачів та надсилання повідомлень.

4.1.3. Використані бібліотеки Python

Модуль math у Python є стандартною бібліотекою, що забезпечує розширений набір інструментів для виконання математичних операцій. Він створений для роботи з базовими та спеціалізованими функціями, включаючи тригонометричні, логарифмічні, степеневі та інші обчислення.

Ця бібліотека підтримує різноманітні типи чисел, такі як числа з плаваючою точкою, цілі числа та дробові. Крім того, вона надає доступ до стандартних математичних констант, як-от π (pi) та e , а також функцій для обчислення факторіалу, коренів, логарифмів тощо.

Однією з головних переваг модуля math є його інтегрованість у Python, що робить його зручним і доступним для виконання завдань різної складності — від простих розрахунків до розв'язання складних математичних задач.

Модуль активно застосовується у програмуванні, аналізі даних, статистиці, математичному моделюванні, наукових дослідженнях і навіть для створення графіків. Часто його використовують разом із бібліотеками NumPy та SciPy, які доповнюють функціональність math і пропонують ще ширший набір можливостей для складних обчислень.

Окрім основних операцій, `math` забезпечує інструменти для роботи з числовими константами та обробки даних із високою точністю. Це робить його незамінним для розробки програмного забезпечення, наукових досліджень і інженерних задач, де необхідні точні математичні розрахунки.

`NumPy` — це потужна бібліотека для числових обчислень і роботи з масивами, яка стала стандартом у світі наукових обчислень з `Python`. Вона була створена у 2005 році Тревісом Оліфантом (Travis Oliphant) і розроблена для ефективної роботи з масивами даних, надаючи широкий набір інструментів для математичних операцій.

Основою `NumPy` є структура даних `ndarray` — n -вимірний масив, який значно перевершує списки `Python` за швидкістю і функціональністю. Завдяки підтримці векторизації та бродкастингу, `NumPy` забезпечує оптимізоване виконання обчислень і зручність у роботі.

Головною перевагою цієї бібліотеки є великий набір математичних функцій, що працюють безпосередньо з масивами, спрощуючи виконання складних операцій. Вона охоплює все — від лінійної алгебри до випадкових вибірок, роблячи її ідеальним вибором для числових обчислень.

`NumPy` також є фундаментом для багатьох інших бібліотек `Python`, таких як `Pandas`, `SciPy` та `Matplotlib`. Вона є ключовим інструментом у наукових дослідженнях, аналізі даних, машинному навчанні та інших областях, де потрібні високопродуктивні обчислення.

Розвиток `NumPy` підтримується завдяки активному внеску спільноти, що забезпечує її постійне вдосконалення. Бібліотека залишається однією з найпопулярніших у сфері обчислювальної науки та аналізу даних і продовжує розширювати свої можливості.

4.2. Опис розробленого прикладного додатку

Головне вікно розробленого додатку має вигляд

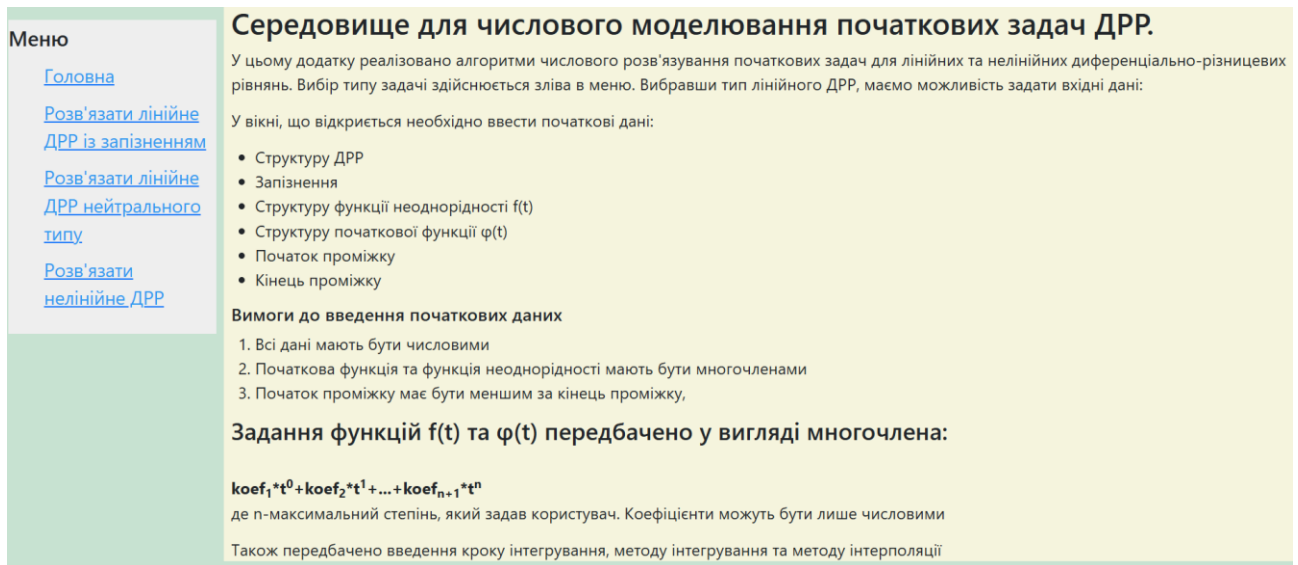


Рисунок 4.

Опис меню прикладного додатку

Пункт **Головна** надає загальні відомості про додаток та вимоги до введення вхідних даних.

Пункт **Розв'язати лінійне ДРР із запізненням** дозволяє розв'язати початкову задачу такого вигляду

$$\begin{aligned}\dot{x}(t) &= ax(t) + bx(t - \tau) + f(t), \\ x(t) &= \varphi(t) \quad t \in [t_0 - \tau; t_0],\end{aligned}$$

задавши необхідні початкові параметри задачі

Пункт **Розв'язати лінійне ДРР нейтрального типу** дозволяє розв'язати початкову задачу такого вигляду

$$\begin{aligned}\dot{x}(t) &= ax(t) + bx(t - \tau) + c\dot{x}(t - \tau) + f(t), \\ x(t) &= \varphi(t) \quad t \in [t_0 - \tau; t_0],\end{aligned}$$

задавши необхідні початкові параметри задачі

Пункт **Розв'язати нелінійне ДРР** дозволяє розв'язати логістичне рівняння

$$\begin{aligned}\dot{x}(t) &= 0.3x(t)(1 - x(t - 1)), \\ x(t) &= 0.1 \quad t \in [-1; 0]\end{aligned}$$

Рисунок 6.

Вибір методу знаходження наближеного розв'язку, типу інтерполяції та кроку інтегрування.

Меню

- [Головна](#)
- [Розв'язати лінійне ДРР із запізненням](#)
- [Розв'язати лінійне ДРР нейтрального типу](#)
- [Розв'язати нелінійне ДРР](#)

Проміжок [0;1]
Введіть кількість розбиття відрізка [0;1]
100 Метод Ейлера ▾
Метод Ейлера
Метод Рунге-Кутти

Проміжок [1;2]
Введіть кількість розбиття відрізка [1;2]
200 Метод Ейлера ▾ Лінійна інтерполяція ▾

Проміжок [2;3]
Введіть кількість розбиття відрізка [2;3]
100 Метод Ейлера ▾ Лінійна інтерполяція ▾

Обчислити

Рисунок 7.

Якщо відомий точний розв'язок задачі, то отримаємо результати обчислень на першому кроці такого вигляду:

Значення t	Наближене значення y	Точне значення y	Похибка
0.0	0.0	0.0	0.0
0.1	-0.08337	-0.08186	0.00151
0.2	-0.13609	-0.13386	0.00223
0.3	-0.16622	-0.16375	0.00247
0.4	-0.17968	-0.17725	0.00244
0.5	-0.18086	-0.17861	0.00225
0.6	-0.17297	-0.17098	0.002
0.7	-0.15841	-0.15669	0.00172
0.8	-0.13891	-0.13746	0.00145
0.9	-0.11578	-0.11458	0.00121
1.0	-0.08998	-0.08898	0.00099

Кількість розбиттів
Метод Ейлера

Рисунок 8.

Користувач може змінити крок інтегрування, вибрати інший метод для знаходження наближеного розв'язку та інший тип інтерполяції початкової функції, і в кінці натиснути на кнопку **Змінити параметри**.

Значення t	Наближене значення y	Точне значення y	Похибка
2.0	-0.04773	-0.04716	0.00057
2.1	-0.04215	-0.04138	0.00077
2.2	-0.03683	-0.03465	0.00218
2.3	-0.03242	-0.0271	0.00532
2.4	-0.02904	-0.01886	0.01019
2.5	-0.02657	-0.01004	0.01653
2.6	-0.02478	-0.00076	0.02401
2.7	-0.02342	0.0089	0.03231
2.8	-0.02228	0.01886	0.04114
2.9	-0.02119	0.02907	0.05026
3.0	-0.02003	0.03948	0.05951

Кількість розбиттів
100

Лінійна інтерполяція

Змінити параметри

Метод Ейлера

Метод Ейлера

Метод Рунге-Кутти

Рисунок 9.

4.3. Аналіз числових експериментів

Розглянемо числове моделювання початкових задач ДРР, за допомогою наведених в роботі алгоритмів. Будемо використовувати позначення:

t – значення незалежної змінної, x_m – точне значення, x_H^E – наближений розв’язок за методом Ейлера, x_H^{PK2} – наближений розв’язок за методом Рунге-Кутти 2 порядку, Δ_E – похибка методу Ейлера, Δ_{PK2} – похибка методу Рунге-Кутти 2 порядку, x_H^E (лін. інтерп.) – наближені значення методу Ейлера із використанням лінійної інтерполяції, x_H^{PK2} (куб. інтерп.) – наближені значення методу Рунге-Кутти 2 порядку із використанням кубічної інтерполяції

Приклад 1. Розглядаємо наближене знаходження розв’язку початкової задачі

$$\dot{x}(t) = x(t) + x(t - 1) + t,$$

$$x(t) = 1, \quad t \in [-1; 0].$$

$$x_T(t) = \begin{cases} 3e^t - t - 2, & t \in [0; 1], \\ (3t - 7)e^{t-1} + 3e^t + 1, & t \in [1; 2], \\ \left(\frac{3}{2}t^2 - 10t + 19\right)e^{t-2} + (3t - 7)e^{t-1} + 3e^t - t - 2, & t \in [2; 3]. \end{cases}$$

t	x_m	x_H^E	x_H^{PK2}	Δ_E	Δ_{PK2}
0.0	1.0	1.0	1.0	0.0	0.0
0.1	1.21551	1.21387	1.21551	0.00165	0.00001
0.2	1.46421	1.46057	1.4642	0.00364	0.00001
0.3	1.74958	1.74355	1.74956	0.00603	0.00002
0.4	2.07547	2.06659	2.07544	0.00888	0.00003
0.5	2.44616	2.4339	2.44612	0.01227	0.00004
0.6	2.86636	2.85009	2.8663	0.01627	0.00005
0.7	3.34126	3.32029	3.34119	0.02097	0.00007

0.8	3.87662	3.85015	3.87653	0.02648	0.00009
0.9	4.47881	4.4459	4.4787	0.03291	0.00011
1.0	5.15485	5.11444	5.15471	0.0404	0.00013

Рисунок 10. Результати числового моделювання початкової задачі на проміжку [0;1]

t	x_m	x_H^E (лін.інтерп.)	x_H^{PK2} (куб. інтерп.)	Δ_E	Δ_{PK2}
1.0	5.15485	5.15471	5.15471	0.00013	0.00013
1.1	5.92337	5.91756	5.9232	0.0058	0.00017
1.2	6.80758	6.79474	6.80737	0.01284	0.00021
1.3	7.82333	7.80184	7.82307	0.02149	0.00026
1.4	8.98849	8.95645	8.98817	0.03204	0.00032
1.5	10.32326	10.27846	10.32287	0.04481	0.00039
1.6	11.85044	11.79026	11.84996	0.06018	0.00048
1.7	13.59571	13.51713	13.59514	0.07858	0.00057
1.8	15.58808	15.48757	15.58739	0.10051	0.00069
1.9	17.8602	17.73365	17.85937	0.12655	0.00083
2.0	20.44889	20.29154	20.44798	0.15735	0.00099
2.1	23.39596	23.37505	23.39477	0.02091	0.00118
2.2	26.75017	26.70443	26.74876	0.04574	0.00141
2.3	30.56731	30.49104	30.56564	0.07627	0.00167
2.4	34.91081	34.79729	34.90883	0.11352	0.00198

2.5	39.85276	39.69407	39.85041	0.15869	0.00235
2.6	45.47509	45.26195	45.47232	0.21314	0.00277
2.7	51.8709	51.59244	51.86763	0.27846	0.00327
2.8	59.14594	58.78943	59.14208	0.35651	0.00386
2.9	67.42032	66.97092	67.41578	0.4494	0.00454
3.0	76.83043	76.27082	76.82509	0.5596	0.00534

Рисунок 11. Результати числового моделювання початкової задачі на проміжку [1;3]

Застосування методу Ейлера та лінійної інтерполяції при кроці $h = 0.01$ дає абсолютну похибку 0.5596, а відносну 0,73%, а використання методу Рунге-Кутти 2 порядку та кубічної інтерполяції дає абсолютну похибку 0,00534, відносну 0,007%.

Приклад 2. Розглядаємо наближене знаходження розв'язку початкової задачі нейтрального типу

$$\dot{x}(t) = x(t) + \dot{x}(t-1),$$

$$x(t) = 1, \quad t \in [-1; 0].$$

$$x_T(t) = \begin{cases} e^t, & t \in [0; 1], \\ e^t + (t-1)e^{t-1}, & t \in [1; 2], \\ e^t + e^{t-1} + 0.5(t-2)(t+2)e^{t-2}, & t \in [2; 3]. \end{cases}$$

t	x_m	x_H^E	x_H^{PK2}	Δ_E	Δ_{PK2}
0.0	1.0	1.0	1.0	0.0	0.0
0.1	1.10517	1.10462	1.10517	0.00055	0.0
0.2	1.2214	1.22019	1.2214	0.00121	0.0
0.3	1.34986	1.34785	1.34985	0.00201	0.00001
0.4	1.49182	1.48886	1.49181	0.00296	0.00001
0.5	1.64872	1.64463	1.64871	0.00409	0.00001
0.6	1.82212	1.8167	1.8221	0.00542	0.00002
0.7	2.01375	2.00676	2.01373	0.00699	0.00002
0.8	2.22554	2.21672	2.22551	0.00883	0.00003
0.9	2.4596	2.44863	2.45957	0.01097	0.00004
1.0	2.71828	2.70481	2.71824	0.01347	0.00004

Рисунок 12. Результати числового моделювання початкової задачі на проміжку [0;1]

t	x_m	x_H^E (лін.інтерп.)	x_H^{PK2} (куб. інтерп.)	Δ_E	Δ_{PK2}
1.0	2.71828	2.71824	2.71824	0.00004	0.00004
1.1	3.11468	3.11256	3.11462	0.00212	0.00006

1.2	3.5644	3.55971	3.56432	0.00469	0.00008
1.3	4.07425	4.06641	4.07415	0.00784	0.0001
1.4	4.65193	4.64025	4.6518	0.01168	0.00013
1.5	5.30605	5.28974	5.30589	0.01631	0.00016
1.6	6.0463	6.02442	6.0461	0.02188	0.0002
1.7	6.88357	6.85503	6.88333	0.02855	0.00025
1.8	7.83008	7.79361	7.82978	0.03648	0.0003
1.9	8.89954	8.85366	8.89918	0.04588	0.00036
2.0	10.10734	10.05035	10.1069	0.05698	0.00044
2.1	11.5868	11.57867	11.58789	0.00812	0.00109
2.2	13.27786	13.26006	13.27901	0.0178	0.00115
2.3	15.20997	15.18021	15.21118	0.02976	0.00121
2.4	17.41653	17.37213	17.41779	0.0444	0.00126
2.5	19.93548	19.87327	19.93678	0.0622	0.0013
2.6	22.80984	22.72611	22.81117	0.08373	0.00133
2.7	26.08844	25.97881	26.08979	0.10962	0.00135
2.8	29.82662	29.68599	29.82798	0.14063	0.00136
2.9	34.08713	33.90952	34.08847	0.17761	0.00135
3.0	38.94107	38.71951	38.94633	0.22156	0.00525

Рисунок 13. Результати числового моделювання початкової задачі на проміжку [1;3]

Застосування методу Ейлера та лінійної інтерполяції при кроці $h = 0.01$ дає абсолютну похибку 0.22156, а відносну 0,57%, а використання методу Рунге-

Кутти 2 порядку та кубічної інтерполяції дає абсолютну похибку 0,00525, відносну 0,013

Приклад 3. Розглядаємо наближене знаходження розв’язку логістичної початкової задачі.

$$\dot{x}(t) = 0.3x(t)(1 - x(t - 1)),$$

$$x(t) = 0.1, \quad t \in [-1; 0].$$

$$x_T(t) = \begin{cases} 0.1e^{0.27t}, & t \in [0; 1], \\ 0.1e^{\frac{73}{900}}e^{0.3t - \frac{1}{9}}e^{0.27(t-1)}, & t \in [1; 2]. \end{cases}$$

t	x_m	x_H^E	x_H^{PK2}	Δ_E	Δ_{PK2}
0.0	0.1	0.1	0.1	0.0	0.0
0.1	0.10274	0.1027	0.10274	0.00004	0.0
0.2	0.10555	0.10547	0.10555	0.00008	0.0
0.3	0.10844	0.10832	0.10844	0.00012	0.0
0.4	0.1114	0.11125	0.1114	0.00016	0.0
0.5	0.11445	0.11425	0.11445	0.0002	0.0
0.6	0.11759	0.11733	0.11759	0.00025	0.0
0.7	0.1208	0.1205	0.1208	0.0003	0.0
0.8	0.12411	0.12376	0.12411	0.00036	0.0
0.9	0.12751	0.1271	0.12751	0.00041	0.0
1.0	0.131	0.13053	0.131	0.00047	0.0

Рисунок 14. Результати числового моделювання початкової задачі на проміжку [0; 1]

t	x_m	x_H^E (лін.інтерп.)	x_H^{PK2} (куб. інтерп.)	Δ_E	Δ_{PK2}
1.0	0.131	0.131	0.131	0.0	0.0
1.1	0.13458	0.13453	0.13458	0.00005	0.00001
1.2	0.13824	0.13815	0.13824	0.00009	0.00001
1.3	0.142	0.14186	0.142	0.00014	0.00001
1.4	0.14584	0.14565	0.14584	0.00019	0.00001
1.5	0.14977	0.14953	0.14977	0.00024	0.00001
1.6	0.1538	0.15351	0.1538	0.00029	0.00001
1.7	0.15791	0.15757	0.15791	0.00034	0.00001
1.8	0.16213	0.16173	0.16213	0.0004	0.00001
1.9	0.16644	0.16598	0.16644	0.00046	0.00001
2.0	0.17084	0.17032	0.17084	0.00052	0.00001

Рисунок 15. Результати числового моделювання початкової задачі на проміжку [1;2]

Застосування методу Ейлера та лінійної інтерполяції при кроці $h = 0.1$ дає абсолютну похибку 0.00052, а відносну 0,3%, а використання методу Рунге-Кутти 2 порядку та кубічної інтерполяції дає абсолютну похибку 0,00001, відносну 0,006%.

Висновки

1. Кваліфікаційна роботи присвячена дослідженню алгоритмів знаходження наближених розв'язків початкових задач для диференціально-різницевих рівнянь.
2. Здійснено аналіз θ методу для наближеного розв'язання початкових задач для диференціальних рівнянь із сталим запізненням та одержано його узагальнення на випадок диференціальних рівнянь нейтрального типу.
3. Побудовано схему автоматизації методу кроків об'єднуючи деякий метод інтегрування задач Коші для звичайних диференціальних рівнянь з використанням лінійної інтерполяції та застосування кубічних інтерполяційних сплайнів.
4. Проаналізувавши числові результати для тестових задач, можна зробити висновок, що використання різницевих схем Рунге-Кутти другого порядку та кубічної інтерполяції дозволяє одержати значно точніші результати як в θ методі.
5. Розроблено прикладний додаток для реалізації запропонованих в роботі алгоритмів та проведено числові експерименти для модельних тестових прикладів.

Список літератури

1. Черевко Ігор, Щур Тетяна, Диренко Віктор. Числове моделювання диференціальних рівнянь із запізненням. VII Міжнародна науково-практична конференція «Моделювання, керування та інформаційні технології (МСІТ–2024)» 7–9 листопада 2024 року, м. Рівне. <https://mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgzQXJtFXrGCvbgXpdLMLDZqbKDr?projector=1&messagePartId=0.1>.
2. Corduneanu C., Li Y., Mahdavi M. Functional Differential Equations: Advances and Applications. – John Wiley & Sons, 2016. – 368 с.
3. Fathalla A. Rihan. Delay Differential Equations and Applications to Biology. Springer, 2021. – 303 с.
4. Gopalsamy K. Stability and Oscillation in Delay Differential Equations of Population Dynamics. – Dordrecht: Kluwer Academic Publishers, 1992. – Vol. 74. – 501 с.
5. Schiesser W.E. Time Delay ODE/PDE Models. Applications in Biomedical Science and Engineering. – Boca Rona, 2019. – 250 с.
6. Forrest-Owen O. Mathematical Modelling and it's Applications in Biology, Ecology and Population Study. – University of Chester, United Kingdom, 2016. – 124 с.
7. Halanay A. Differential equations. Stability. Oscillations. Time Lags. – New York ; London: Acad. Press, 1968. – 528 с.
8. J.K. Hale: Theory of Functional Differential Equations, Applied Mathematics Sciences 3, Springer-Verlag, New York-Heidelberg, 1977.
9. J.K. Hale and S.M. Verduyn Lunel: Introduction to functional differential equations, Applied Mathematical Sciences 99, Springer-Verlag, New York, 1993.
10. Слюсарчук В. Ю. Абсолютна стійкість динамічних систем із післядією. – Рівне: УДУВГ, 2003. – 288 с.

11. В.І. Фодчук, Я.Й. Бігун, І.І. Клевчук, І.М. Черевко, І.В. Якімов Регулярно і сингулярно збудені диференціально-функціональні рівняння. – Київ: Ін-т математики НАН України, 1996. – 210 с.
12. Митропольський Ю.А., Мартинюк Д. И. Периодические и квазипериодические колебания систем с запаздыванием.-К.:Вища шк.,1979. – 309 с.
13. Самойленко А.М., Ронто Н.И. Численно аналитические методы исследования решений краевых задач.-К.:Наук.думка, 1985. – 224 с.
14. Bellen A., Zenaro M. Numerical methods for delay differential equations. – New York : Oxford University Press, 2003. – 395 с.
15. Kuang J., Cong Y. Stability of Numerical Methods for Delay Differential Equations. – Elsevier Science, 2007. – 295 с.
16. Черевко І.М., Матвій О.В. Про апроксимацію систем із запізненням та їх стійкість // Нелінійні коливання. – 2004. – Т.7, № 2. – с. 208–216.
17. Дорош А.Б., Тузик І.І., Черевко І.М. Схеми апроксимації крайових задач для інтегро-диференціальних рівнянь із запізненням. Нелінійні коливання. 2023. Т. 26, № 1. с. 33–41.
18. H . Arndt. Numerical Solution of Retarded Initial Value Problems: Local and Global Error and Stepsize Control. Numer.Math.43(1984) с. 343–360.
19. K. Aboodh, R. Farah, I. Almardy, and A. Osman. Solving delay differential equations by aboodh transformation method. International Journal of Applied Mathematics & Statistical Sciences, 7(2): с. 55–64, 2018.
20. Yogita M. Mahatekar, Pallavi Scindia. A New family of methods for solving delay differential equations. 2021. [10.48550/arXiv.2102.06925](https://arxiv.org/abs/10.48550/arXiv.2102.06925).
21. Луник Т.В., Черевко І.М. Моделювання математичних моделей біології та імунології із запізненням. Буковинський математичний журнал. 2021. Том. 8, № 2 (2021). с. 92–98.
22. J.H. Ahlberg, E.N. Nilson and J.L. Walsh, The Theory of Splines and their Applications, Academic Press, New York, 1967.

23. C. de Boor A Practical Guide to Splines, Springer-Verlag, New York/Heidelberg/Berlin (1978).
24. Gary D. Knott. Interpolating Cubic Splines. Birkhäuser. Boston • Basel • Berlin .2000. – 215 с.
25. Гантмахер Ф. Р. Теорія матриць. – 2024. – 703 с.
<https://nebayduzhimath.azurewebsites.net/%d0%93%d0%b0%d0%bd%d1%82%d0%bc%d0%b0%d1%85%d0%b5%d1%80%d0%a2%d0%b5%d0%be%d1%80%d1%96%d1%8f%d0%9c%d0%b0%d1%82%d1%80%d0%b8%d1%86%d1%8c/>.
26. Мова програмування Python <https://acode.com.ua/intro-python/>.
27. Переваги та недоліки Django <https://hackr.io/blog/what-is-django-advantages-and-disadvantages-of-using-django>.
28. Фреймворк Django <https://uk.wikipedia.org/wiki/Django>.

Додаток 1 (метод Рунге-Кутти 2 порядку)

```

def find_for_Runge221(y_start,t,h,m,tau,koefa,koefb,koefc,t_old,spline,f,h_old,equal):
    res = dict()
    y = []
    y.append(y_start[-1])
    for i in range(1,len(t)):
        elem, index = find_bounds(t_old,t[i-1]-tau)
        suma1 = 0
        if(equal != 'log'):
            for c in range(0,len(f)):
                suma1+=f[c]*t[i-1]**c

        if(spline == 'cub'):
            t_new = (t[i-1]-tau-elem)/h_old
            S = y_start[index]*(1-t_new)**2*(1+2*t_new)+y_start[index+1]*t_new**2*(3-
            2*t_new)+m[index]*h_old*t_new*(1-t_new)**2-m[index+1]*h_old*t_new**2*(1-t_new)
            if(equal == 'netr'):
                S_ = y_start[index]*(6*t_new**2-6*t_new)+y_start[index+1]*(6*t_new-
                6*t_new**2)+m[index]*h_old*(1-4*t_new+3*t_new**2)-m[index+1]*h_old*(2*t_new-
                3*t_new**2)
                S_*=((round(tau/h_old)))
                t_new = (t[i-1]+(h/2)-tau-elem)/h_old
                S1_ = y_start[index]*(6*t_new**2-6*t_new)+y_start[index+1]*(6*t_new-
                6*t_new**2)+m[index]*h_old*(1-4*t_new+3*t_new**2)-m[index+1]*h_old*(2*t_new-
                3*t_new**2)
                S1_*=((round(tau/h_old)))

            t_new = (t[i-1]+(h/2)-tau-elem)/h_old
            S1 = y_start[index]*(1-t_new)**2*(1+2*t_new)+y_start[index+1]*t_new**2*(3-
            2*t_new)+m[index]*h_old*t_new*(1-t_new)**2-m[index+1]*h_old*t_new**2*(1-t_new)

        if(spline == 'lin'):
            S = (t[i-1]-tau-elem)/h_old*y_start[index+1]+(t_old[index+1]-t[i-
            1]+tau)/h_old*y_start[index]
            if(equal == 'netr'):
                S_ = (y_start[index+1]/h_old)-(y_start[index]/h_old)

                S1 = (t[i-1]+(h/2)-tau-elem)/h_old*y_start[index+1]+(t_old[index+1]-(t[i-
            1]+(h/2))+tau)/h_old*y_start[index]

        if(equal == 'usual'):
            K1 = koefa*y[-1]+koefb*S+suma1
            suma1 = 0
            for c in range(0,len(f)):
                suma1+=f[c]*(t[i-1]+h/2)**c
            K2 = koefa*(y[-1]+(h/2)*K1)+koefb*S1+suma1

        if(equal == 'netr'):
            K1 = koefa*y[-1]+koefb*S+koefc*S_+suma1
            suma1 = 0
            for c in range(0,len(f)):

```

```

        suma1+=f[c]*(t[i-1]+h/2)**c
    if(spline=="lin"):
        K2 = koefa*(y[-1]+(h/2)*K1)+koefb*S1+koefc*S_+suma1
    elif(spline=="cub"):
        K2 = koefa*(y[-1]+(h/2)*K1)+koefb*S1+koefc*S1_+suma1

    if(equal == 'log'):
        K1 = 0.3*y[-1]*(1-S)
        K2 = 0.3*(y[-1]+(h/2)*K1)*(1-S1)
    y.append(y[-1]+h*K2)

```

Додаток 2 (метод Ейлера)

```

def find_for_Euler(y_start,t,h,m,tau,koefa,koefb,koefc,t_old,spline,f,h_old,equal):
    res = dict()
    y = []
    p = 1
    y.append(y_start[-1])
    res.update({round(t[0],4):round(y[0],5)})
    for i in range(1,len(t)):

        suma1 = 0
        if(equal != 'log'):
            for c in range(0,len(f)):
                suma1+=f[c]*t[i-1]**c
            elem, index = find_bounds(t_old,t[i-1]-tau)
            if(spline == 'cub'):
                t_new = (t[i-1]-tau-elem)/h_old
                S_ = y_start[index]*(1-t_new)**2*(1+2*t_new)+y_start[index+1]*t_new**2*(3-
                2*t_new)+m[index]*h_old*t_new*(1-t_new)**2-m[index+1]*h_old*t_new**2*(1-t_new)
                if(equal == 'netr'):
                    S_ = y_start[index]*(6*t_new**2-6*t_new)+y_start[index+1]*(6*t_new-
                    6*t_new**2)+m[index]*h_old*(1-4*t_new+3*t_new**2)-m[index+1]*h_old*(2*t_new-
                    3*t_new**2)
                S_*= (round(tau/h_old))
            if(spline == 'lin'):
                S = (t[i-1]-tau-elem)/h_old*y_start[index+1]+(t_old[index+1]-t[i-
                1]+tau)/h_old*y_start[index]
            if(equal == 'netr'):
                S_ = (y_start[index+1]/h_old)-(y_start[index]/h_old)
            if(equal == 'usual'):
                y.append(y[-1]+h*(koefa*y[-1]+koefb*S+suma1))
            if(equal == 'netr'):
                y.append(y[-1]+h*(koefa*y[-1]+koefb*S+koefc*S_+suma1))
            if(equal == 'log'):
                y.append(y[-1]+h*(0.3*y[-1]*(1-S)))
            if(p%10 == 0):
                res.update({round(t[i],4):round(y[-1],5)})
            p+=1
    return y, res

```

Додаток 3 (обчислення коефіцієнтів кубічного сплайну)

```
def coeffs_spline(t,y,n,tau,l,koefa,koefb,y_first,f,equal='log'):
    mju = []
    lambdas = []
    c = []
    suma = 0
    if(equal != 'log'):
        for o in range(0,len(f)):
            suma+=f[o]*(t[0])**o
            c.append(2*(koefa*y[0]+koefb*y_first+suma))
    elif(equal == 'log'):
        c.append(0.2)
    h = tau/n[1]
    mju.append(0)
    matr_koefs = []
    matr_koefs.append([0]*(len(t)))
    matr_koefs[0][0] = 2
    matr_koefs[0][1] = mju[0]
    for i in range(1,len(t)-1):
        mju.append(h*(h+h)**-1)
        matr_koefs.append([0]*(len(t)))
        lambdas.append(1-mju[-1])
        matr_koefs[i][i-1] = lambdas[i-1]
        matr_koefs[i][i] = 2
        matr_koefs[i][i+1] = mju[i]
        c.append(3*(mju[-1]*((y[i+1]-y[i])/h)+lambdas[-1]*((y[i]-y[i-1])/h)))
    matr_koefs.append([0]*(len(t)))
    matr_koefs[len(t)-1][len(t)-1] = 2
    lambdas.append(0)
    suma = 0
    if(equal != 'log'):
        for o in range(0,len(f)):
            suma+=f[o]*(t[-1])**o
            c.append(2*(koefa*y[-1]+koefb*y[0]+suma))
    elif(equal == 'log'):
        c.append(2*(0.1*exp(0.27)))
    v = []
    u = []
    v.append(0)
    u.append(0)

    v.append(0)
    u.append(c[0]/matr_koefs[0][0])
    m = []
    for i in range(1,len(matr_koefs)-1):
        v.append(-matr_koefs[i][i+1]/
            (matr_koefs[i][i]+matr_koefs[i][i-1]*v[-1]))
        u.append((c[i]-matr_koefs[i][i-1]*u[-1])/
            (matr_koefs[i][i]+matr_koefs[i][i-1]*v[i]))

    v.append(0)
    u.append((c[-1]-matr_koefs[len(matr_koefs)-1][len(matr_koefs)-2]*u[-1])/matr_koefs[len(matr_koefs)-1][len(matr_koefs)-1])
```

```

m.append(u[-1])
for i in range(len(u)-2,0,-1):
    m.append(v[i]*m[-1]+u[i])
m.reverse()
return m

```

Додаток 4 (розв'язання різних видів ДРР)

```

def solvlog(start,end,n,method,spline):
    show = []
    res = dict()
    starts = []
    h = 1/n[0]
    p = 1
    starts.append(start)
    t = t_i(start,h,n[0])
    y = []
    y.append(0.1)
    y_first = y[0]
    tau = 1
    res.update({round(t[0],4):round(y[-1],5)})
    for i in range(1,len(t)):
        if(method[0] == 'eil'):
            y.append(y[-1]+h*(0.27*y[-1]))
        elif(method[0] == 'rung'):
            K1 = 0.27*y[-1]
            K2 = 0.3*(y[-1]+(h/2)*K1)*0.9
            y.append(y[-1]+h*K2)
    if(p%10 == 0):
        res.update({round(t[i],4):round(y[-1],5)})
        p+=1

    res.update({'n':n[0],'method':method[0],
               'spline':spline[0],'starting':0})
    show.append(res)
    m=[]
    res1 = dict()
    flag = 0
    for k in range(start+tau,end,tau):
        start+=tau
        starts.append(start)
        t_old = t
        h_old = h
        m = coeffs_spline(t,y,n,tau,flag,1,1,y_first,1,'log')
        flag+=1
        y_first = y[0]
        h = tau/n[flag]
        t = t_i(start,h,n[flag])
        if(method[flag] == 'eil'):
            y,res1 = find_for_Euler(y,t,h,m,tau,1,1,1,t_old,spline[flag],1,h_old,'log')
        elif(method[flag] == 'rung'):
            y,res1 = find_for_Runge221(y,t,h,m,tau,1,1,1,t_old,spline[flag],1,h_old,'log')

```

```

res1.update({'n':n[flag],'method':method[flag],
            'spline':spline[flag],'starting':k})
show.append(res1)
return show, starts

```

```

def solvnetr(start,end,tau,n,method,spline,koefa,koefb,koefc,f,fi):
    show = []
    res = dict()
    starts = []
    h = tau/n[0]
    p = 1
    starts.append(start)
    t = t_i(start,h,n[0])
    y = []
    suma = 0
    for c in range(0,len(fi)):
        suma+=fi[c]*(t[0])**c
    y.append(suma)
    y_first = y[0]
    res.update({round(t[0],4):round(y[-1],5)})
    for i in range(1,len(t)):
        suma1 = 0
        suma2 = 0
        suma3 = 0
        for c in range(0,len(fi)):
            suma1+=fi[c]*(t[i-1]-tau)**c
        for c in range(0,len(f)):
            suma2+=f[c]*t[i-1]**c
        for c in range(1,len(fi)):
            suma3+=fi[c]*c*(t[i-1]-tau)**(c-1)
        if(method[0] == 'eil'):
            y.append(y[-1]+h*(koefa*y[-1]+koefb*suma1+koefc*suma3+suma2))
        elif(method[0] == 'rung'):
            K1 = koefa*y[-1]+koefb*suma1+koefc*suma3+suma2
            suma1 = 0
            suma2 = 0
            suma3 = 0
            for c in range(0,len(fi)):
                suma1+=fi[c]*(t[i-1]+(h/2)-tau)**c
            for c in range(0,len(f)):
                suma2+=f[c]*(t[i-1]+h/2)**c
            for c in range(1,len(fi)):
                suma3+=fi[c]*c*(t[i-1]+(h/2)-tau)**(c-1)
            K2 = koefa*(y[-1]+(h/2)*K1)+koefb*suma1+koefc*suma3+suma2
            y.append(y[-1]+h*K2)

```

```

def solvnetr(start,end,tau,n,method,spline,koefa,koefb,koefc,f,fi):
    show = []
    res = dict()
    starts = []
    h = tau/n[0]

```

```

p = 1
starts.append(start)
t = t_i(start,h,n[0])
y = []
suma = 0
for c in range(0,len(fi)):
    suma+=fi[c]*(t[0])**c
y.append(suma)
y_first = y[0]
res.update({round(t[0],4):round(y[-1],5)})
for i in range(1,len(t)):
    suma1 = 0
    suma2 = 0
    suma3 = 0
    for c in range(0,len(fi)):
        suma1+=fi[c]*(t[i-1]-tau)**c
    for c in range(0,len(f)):
        suma2+=f[c]*t[i-1]**c
    for c in range(1,len(fi)):
        suma3+=fi[c]*c*(t[i-1]-tau)**(c-1)
    if(method[0] == 'eil'):
        y.append(y[-1]+h*(koefa*y[-1]+koefb*suma1+koefc*suma3+suma2))
    elif(method[0] == 'rung'):
        K1 = koefa*y[-1]+koefb*suma1+koefc*suma3+suma2
        suma1 = 0
        suma2 = 0
        suma3 = 0
        for c in range(0,len(fi)):
            suma1+=fi[c]*(t[i-1]+(h/2)-tau)**c
        for c in range(0,len(f)):
            suma2+=f[c]*(t[i-1]+h/2)**c
        for c in range(1,len(fi)):
            suma3+=fi[c]*c*(t[i-1]+(h/2)-tau)**(c-1)
        K2 = koefa*(y[-1]+(h/2)*K1)+koefb*suma1+koefc*suma3+suma2
        y.append(y[-1]+h*K2)
    if(p%10 == 0):
        res.update({round(t[i],4):round(y[-1],5)})
    p+=1

res.update({'n':n[0],'method':method[0],
            'spline':spline[0],'starting':0})
show.append(res)
m=[]
res1 = dict()
flag = 0
K2=0
for k in range(start+tau,end,tau):
    start+=tau
    starts.append(start)
    t_old = t
    h_old = h
    m = coeffs_spline(t,y,n,tau,flag,koefa,koefb,y_first,f,'netr')
    flag+=1

```

```

y_first = y[0]
h = tau/n[flag]
t = t_i(start,h,n[flag])
if(method[flag] == 'eil'):
    y,res1 = find_for_Euler(y,t,h,m,tau,koefa,koefb,koefc,t_old,spline[flag],f,h_old,'netr')
elif(method[flag] == 'rung'):
    y,res1 = find_for_Runge221(y,t,h,m,tau,koefa,koefb,koefc,t_old,spline[flag],f,h_old,'netr')
res1.update({'n':n[flag],'method':method[flag],
            'spline':spline[flag],'starting':k})
show.append(res1)
return show, starts

```

```

def solvdif(start,end,tau,n,method,spline,koefa,koefb,f,fi):
    koefc = 0
    show = []
    res = dict()
    starts = []
    h = tau/n[0]
    p = 1
    starts.append(start)
    t = t_i(start,h,n[0])
    y = []
    suma = 0
    for c in range(0,len(fi)):
        suma+=fi[c]*(t[0])**c
    y.append(suma)
    y_first = y[0]
    res.update({round(t[0],4):round(y[-1],5)})
    for i in range(1,len(t)):
        suma1 = 0
        suma2 = 0
        for c in range(0,len(fi)):
            suma1+=fi[c]*(t[i-1]-tau)**c
        for c in range(0,len(f)):
            suma2+=f[c]*t[i-1]**c

        if(method[0] == 'eil'):
            y.append(y[-1]+h*(koefa*y[-1]+koefb*suma1+suma2))
        elif(method[0] == 'rung'):
            K1 = koefa*y[-1]+koefb*suma1+suma2
            suma1 = 0
            suma2 = 0
            for c in range(0,len(fi)):
                suma1+=fi[c]*(t[i-1]-tau+(h/2))**c
            for c in range(0,len(f)):
                suma2+=f[c]*(t[i-1]+h/2)**c
            K2 = koefa*(y[-1]+(h/2)*K1)+koefb*suma1+suma2
            y.append(y[-1]+h*K2)
    if(p%10 == 0):
        res.update({round(t[i],4):round(y[-1],5)})
        p+=1
    res.update({'n':n[0],'method':method[0],

```

```

        'spline':spline[0],'starting':0})
show.append(res)
m=[]
res1 = dict()
for k in range(start+tau,end,tau):
    start+=tau
    starts.append(start)
    t_old = t
    h_old = h
    m = coeffs_spline(t,y,n,tau,k-1,koefa,koefb,y_first,f,'usual')
    y_first = y[0]
    h = tau/n[k]
    t = t_i(start,h,n[k])
    if(method[k] == 'eil'):
        y,res1 = find_for_Euler(y,t,h,m,tau,koefa,koefb,koefc,t_old,spline[k],f,h_old,'usual')
    elif(method[k] == 'rung'):
        y,res1 = find_for_Runge221(y,t,h,m,tau,koefa,koefb,koefc,t_old,spline[k],f,h_old,'usual')

    res1.update({'n':n[k],'method':method[k],
                'spline':spline[k],'starting':k})
    show.append(res1)
return show,starts

```

Додаток 5 (клієнтська частина додатку)

Головна сторінка

```
{% extends 'main/layout.html' %}
```

```
{% block title %}
```

```
Головна сторінка
```

```
{% endblock %}
```

```
{% block content %}
```

```
<div class="main">
```

```
    <h1>Середовище для числового моделювання початкових задач ДРР.</h1>
```

```
    <p>У цьому додатку реалізовано алгоритми числового розв'язування початкових задач для
        лінійних та нелінійних диференціально-різницевих рівнянь. Вибір типу задачі
здійснюється
```

```
        зліва в меню. Вибравши тип лінійного ДРР, маємо можливість задати вхідні дані:
```

```
    <!-- Для задання необхідних вхідних даних, потрібно вибрати пункт <b>Розв'язати</b> --
>
```

```
</p>
```

```
<p>У вікні, що відкриється необхідно ввести початкові дані:
```

```
    <ul>
```

```
        <li>Структуру ДРР</li>
```

```
        <li>Запізнення</li>
```

```
        <li>Структуру функції неоднорідності  $f(t)$ </li>
```

```
        <li>Структуру початкової функції  $\varphi(t)$ </li>
```

```
        <li>Початок проміжку</li>
```

```
        <li>Кінець проміжку</li>
```

```
    </ul>
```

```
</p>
```

```
<h4>Вимоги до введення початкових даних</h4>
```



```

{% if maxfunc != 0 %}
<label>t<sup>0</sup>+</label>
{% endif %}
{% if maxfunc == 0 %}
<label>t<sup>0</sup></label>
{% endif %}

{% with "|center:maxfunc as range %"
{% for _ in range %}
    {% if forloop.counter != maxfunc %}
        <input type="text" class="coef" name="f{{ forloop.counter }}" style="width: 80px;">
        <label for="count">t<sup>{{ forloop.counter }}</sup>+</label>
    {% endif %}
    {% if forloop.counter == maxfunc %}
        <input type="text" class="coef" name="f{{ forloop.counter }}" value="{{ f.maxfunc }}"
style="width: 80px;">
        <label for="count">t<sup>{{ forloop.counter }}</sup></label>
    {% endif %}

{% endfor %}
{% endwith %}

<h3>Введіть коефіцієнти функції  $\varphi(t)$ </h3>
<label> $\varphi(t)$  = </label>
{% if maxfi != 0 %}
<input type="text" class="coef" name="fi0" value="{{ fi.0 }}" style="width: 80px;">
<label for="count">t<sup>0</sup>+</label>
{% endif %}
{% if maxfi == 0 %}
<input type="text" class="coef" name="fi0" value="{{ fi.0 }}" style="width: 80px;">
<label for="count">t<sup>0</sup></label>
{% endif %}
{% with "|center:maxfi as range %"
{% for _ in range %}
    {% with forloop.counter as num %}
    {% if forloop.counter != maxfi %}
        <input type="text" class="coef" name="fi{{ forloop.counter }}" style="width: 80px;">
        <label for="count">t<sup>{{ forloop.counter }}</sup>+</label>
    {% endif %}
    {% if forloop.counter == maxfi %}
        <input type="text" class="coef" name="fi{{ forloop.counter }}" value="{{ fi.maxfi }}"
style="width: 80px;">
        <label for="count">t<sup>{{ forloop.counter }}</sup></label>
    {% endif %}
    {% endwith %}
{% endfor %}
{% endwith %}

<input type="number" name="start" value="{{ start }}" hidden>
<input type="number" name="end" value="{{ end }}" hidden>
<input type="number" name="late" value="{{ late }}" hidden>
<input type="number" name="koefa" value="{{ koefa }}" hidden>
<input type="number" name="koefb" value="{{ koefb }}" hidden>
<input type="number" name="maxfunc" value="{{ maxfunc }}" hidden>

```

```

<input type="number" name="maxfi" value="{{maxfi}}" hidden>
{% if typedif == 'netr' %}
  <input name = "koefc" value="{{koefc}}" hidden>
  <input name = "latec" value="{{latec}}" hidden>
  <input name="typedif" value="netr" hidden>
{% endif %}

{% if typedif == 'usual' %}
  <input name="typedif" value="usual" hidden>
{% endif %}
<br>
<h2 style=color:red>{{error}}</h2>
<button type="submit">Далі</button>
</form>
</div>
{% endblock %}

{% extends 'main/layout.html' %}
{% load index %}
{% block title %}
Розбиття
{% endblock %}
{% block content %}
<div class="main">
{% if typedif == 'usual' %}
<form id="myForm" method="post" action="{% url 'solv' %}">
  <input name="typedif" value="usual" hidden>
{% endif %}
{% if typedif == 'netr' %}
<form id="myForm" method="post" action="{% url 'netral' %}">
{% endif %}
{% if typedif == 'log' %}
<form id="myForm" method="post" action="{% url 'logistic' %}">
{% endif %}
  {% csrf_token %}
  {% for elem1, elem2 in ranges.items %}
  <div class="form-group">
    <h5>Проміжок {{elem2}}</h5>
    <label for="count">Введіть кількість точок розбиття відрізка {{elem2}}</label>
    <input type="number" name="{{elem1}}" style="width: 120px;">
    <select name="method{{elem1}}">
      <option value="eil">Метод Ейлера</option>
      <option value="rung">Метод Рунге-Кутти</option>
    </select>
    {% if elem1 != start %}
    <select name="spline{{elem1}}">
      <option value="lin">Лінійна інтерполяція</option>
      <option value="cub">Кубічна інтерполяція</option>
    </select>
    {% endif %}
  </div>
  {% endfor %}
  <input type="number" name="start" value="{{start}}" hidden>

```

```

<input type="number" name="end" value="{{end}}" hidden>
<input type="number" name="late" value="{{late}}" hidden>
<input type="number" name="koefa" value="{{koefa}}" hidden>
<input type="number" name="koefb" value="{{koefb}}" hidden>
<input type="number" name="maxfunc" value="{{maxfunc}}" hidden>
<input type="number" name="maxfi" value="{{maxfi}}" hidden>
{% if typedif == 'netr'%}
    <input type="number" name="koefc" value="{{koefc}}" hidden>
    <input type="number" name="latec" value="{{latec}}" hidden>
    <input name="typedif" value="netr" hidden>
{% endif %}

{% if typedif == 'log' %}
    <input name="typedif" value="log" hidden>
{% endif %}
{% if typedif != 'log'%}
<input name="f0" value="{{f.0}}" hidden>
{% with "|center:maxfunc as range %}
{% for _ in range %}
<input name="f{{forloop.counter}}" value="{{f|index:forloop.counter}}" hidden>
{% endfor %}
{% endwith %}

<input name="fi0" value="{{fi.0}}" hidden>
{% with "|center:maxfi as range %}
{% for _ in range %}
<input name="fi{{forloop.counter}}" value="{{fi|index:forloop.counter}}" hidden>
{% endfor %}
{% endwith %}
{% endif %}
<button type="submit">Обчислити</button>
</form>
</div>
{% endblock %}

{% extends 'main/layout.html' %}

{% block title %}
Рівняння із запізненням нейтрального типу
{% endblock %}

{% block content %}
<div class="main">
<form id="myForm" method="post" action="{% url 'divide' %}">
    {% csrf_token %}
    <h4>Логістичне рівняння  $x'(t) = 0.3x(t)(1-x(t-1))$ </h4>
    <div class="form-group">
        <label for="start">Введіть початок проміжку</label>
        <input type="text" name="start" value="{{start}}" style="width: 80px;">
    </div>
    <div class="form-group">
        <label for="end">Введіть кінець проміжку</label>
        <input type="text" name="end" value="{{end}}" style="width: 80px;">

```

```

</div>
<h2 style=color:red>{{error}}</h2>
<input type="number" name="late" value="1" hidden>
<input name="typedif" value="log" hidden>
<button type="submit">Далі</button>
</form>
</div>
{% endblock %}

```

```
{% extends 'main/layout.html' %}
```

```

{% block title %}
Рівняння із запізненням нейтрального типу
{% endblock %}

```

```

{% block content %}
<div class="main">
<form id="myForm" method="post" action="{% url 'coeffunc' %}">
  {% csrf_token %}
  <h2>Заповніть поля</h2>
  <h3>Введіть коефіцієнти</h3>
  <h4>Задача  $x'(t) =$  <input type="text" name="koefa" value="{{koefa}}" style="width:
40px;"/> $x(t) +$  <input type="text" name="koefb" value="{{koefb}}" style="width: 40px;"/> $x(t) \cdot$ 
<input type="text" name="late" value="{{late}}" style="width: 40px;"/> <input type="text"
name="koefc" value="{{koefc}}" style="width: 40px;"/>  $x'(t) \cdot$  <input type="text" name="latec"
value="{{latec}}" style="width: 40px;"/>  $+ f(t)$ </h4>
  <div class="form-group">
    <label for="start">Введіть максимальний степінь функції  $f(t)$ </label>
    <input type="number" name="maxfunc" value="{{maxfunc}}" style="width: 80px;">
  </div>
  <div class="form-group">
    <label for="start">Введіть максимальний степінь початкової функції  $\varphi(t)$ </label>
    <input type="number" name="maxfi" value="{{maxfi}}" style="width: 80px;">
  </div>
  <div class="form-group">
    <label for="start">Введіть початок проміжку</label>
    <input type="text" name="start" value="{{start}}" style="width: 80px;">
  </div>
  <div class="form-group">
    <label for="end">Введіть кінець проміжку</label>
    <input type="text" name="end" value="{{end}}" style="width: 80px;">
  </div>
  <h2 style=color:red>{{error}}</h2>
  <input name="typedif" value="netr" hidden>
  <button type="submit">Далі</button>
</form>
</div>
{% endblock %}

```

```
{% extends 'main/layout.html' %}
```

```

{% block title %}
Розв'язок
{% endblock %}

{% block content %}
<div class="main">
<form id="myForm" method="post" action="{% url 'coeffunc' %}">
  {% csrf_token %}
  <h2>Заповніть поля</h2>
  <h3>Введіть коефіцієнти</h3>
  <h4>Задача  $x'(t) =$  <input type="text" name="koefa" value="{{ koefa }}" style="width:
40px;"/> $x(t) +$  <input type="text" name="koefb" value="{{ koefb }}" style="width: 40px;"/> $x(t)$ -
<input type="text" name="late" value="{{ late }}" style="width: 40px;"/>)+ $f(t)$ </h4>
  <div class="form-group">
    <label for="start">Введіть максимальний степінь функції  $f(t)$ </label>
    <input type="number" name="maxfunc" value="{{ maxfunc }}" style="width: 80px;">
  </div>
  <div class="form-group">
    <label for="start">Введіть максимальний степінь початкової функції  $\varphi(t)$ </label>
    <input type="number" name="maxfi" value="{{ maxfi }}" style="width: 80px;">
  </div>
  <div class="form-group">
    <label for="start">Введіть початок проміжку</label>
    <input type="text" name="start" value="{{ start }}" style="width: 80px;">
  </div>
  <div class="form-group">
    <label for="end">Введіть кінець проміжку</label>
    <input type="text" name="end" value="{{ end }}" style="width: 80px;">
  </div>
  <h2 style=color:red>{{ error }}</h2>
  <input name="typedif" value="usual" hidden>
  <button type="submit">Далі</button>
</form>
</div>
{% endblock %}

```

Вивід результатів

```

{% extends 'main/layout.html' %}
{% load index %}
{% block content %}
<div class="main">

{% if typedif == 'usual' %}
<form method="post" action="{% url 'solv' %}">
{% endif %}

{% if typedif == 'netr' %}
<form method="post" action="{% url 'netral' %}">
{% endif %}

{% if typedif == 'log'%}
<form method="post" action="{% url 'logistic' %}">

```

```

{% endif %}
{% for elem in arrs %}
  {% csrf_token %}
  <table>
    <tr>
      <th>Значення t</th>
      <th>Наближене значення y</th>
    </tr>
    <tr>
      {% for key, value in elem.items %}
        {% if key != 'n' and key != 'method' and key != 'spline' and key != 'starting'%}
          <td>{{key}}</td>
          <td>{{value}}</td>
        {% endif %}
      </tr>
    {% endfor %}
  </table>
  <div class="divid">
    <label for="count">Кількість розбиттів</label>
    <input type="number" name="{{elem.starting}}" value="{{elem.n}}">

    <select name="method{{elem.starting}}">
      {% if elem.method == 'eil' %}
        <option value="eil" selected>Метод Ейлера</option>
        <option value="rung">Метод Рунге-Кутти</option>
      {% endif %}

      {% if elem.method == 'rung' %}
        <option value="eil">Метод Ейлера</option>
        <option value="rung" selected>Метод Рунге-Кутти</option>
      {% endif %}
    </select>

    <select name="spline{{elem.starting}}">
      {% if elem.spline == 'lin' %}
        <option value="lin" selected>Лінійна інтерполяція</option>
        <option value="cub">Кубічна інтерполяція</option>
      {% endif %}

      {% if elem.spline == 'cub' %}
        <option value="lin">Лінійна інтерполяція</option>
        <option value="cub" selected>Кубічна інтерполяція</option>
      {% endif %}
    </select>
  </div>
{% endfor %}
<input type="number" name="start" value="{{start}}" hidden>
<input type="number" name="end" value="{{end}}" hidden>
<input type="number" name="late" value="{{late}}" hidden>

<input type="number" name="koefa" value="{{koefa}}" hidden>
<input type="number" name="koefb" value="{{koefb}}" hidden>
<input type="number" name="maxfunc" value="{{maxfunc}}" hidden>

```

```

{% if typedif == 'netr' %}
  <input name="koefc" value="{{ koefc }}" hidden>
{% endif %}
{% if typedif != 'log' %}
  <input type="number" name="maxfi" value="{{ maxfi }}" hidden>
  <input name="f0" value="{{ f.0 }}" hidden>
  {% with "|center:maxfunc as range %}" %}
  {% for _ in range %}
    <input name="f{{ forloop.counter }}" value="{{ f|index:forloop.counter }}" hidden>
  {% endfor %}
  {% endwith %}

  <input name="fi0" value="{{ fi.0 }}" hidden>
  {% with "|center:maxfi as range %}" %}
  {% for _ in range %}
    <input name="fi{{ forloop.counter }}" value="{{ fi|index:forloop.counter }}" hidden>
  {% endfor %}
  {% endwith %}
{% endif %}
<button type="submit">Змінити параметри</button>
</form>

</div>
{% endblock %}

```