# ВСТУП

В багатьох наукових дисциплінах виникають задачі, пов'язані з диференціальними рівняннями, а зокрема із системами диференціальних рівнянь. Вони є надважливим класом задач, тому не дивно, що на даний момент існує дуже багато точних та чисельних методів для їх розв'язання, в тому числі і для розв’язання задачі Коші для систем з *m* звичайних диференціальних рівнянь першого порядку.

Одним з таких методів є ітеративна процедура Пікара послідовних наближень, проте її застосування на практиці, на жаль, часто обмежується ілюстративними прикладами в межах курсів чисельних методів, що пов'язано, серед іншого, із складністю імплементації цього методу на ЕОМ, через потребу багаторазового інтегрування потенційно неполіноміальних функцій, що може призвести до надзвичайно складних квадратур, які можуть навіть і не виражатися в елементарних функціях.

Вирішенню цієї проблеми, власне, і присвячена ця робота. Особлива увага приділяється розробленню математичного та програмного забезпечення для ефективного застосування метода Пікара послідовних наближень на ЕОМ для систем диференціальних рівнянь. Додатковою вимогою є знаходження розв’язків у аналітичному вигляді, на відміну від табличних методів, таких як метод Рунге-Кутта, які потребують додаткового знаходження інтерполяційного многочлена між вузлами сітки.

Кінцевий продукт може бути використаний у різних інженерних середовищах для наближеного розв'язування диференціальних рівнянь, а також в якості демонстраційного інструменту для поліпшення засвоєння методу Пікара студентами-математиками.

# 1 ПОСТАНОВКА ЗАДАЧІ

Дана робота полягає у розроблені системи розв’язання систем диференціальних рівнянь із заданими початковими умовами методом Пікара послідовних наближень.

У якості вхідних даних, забезпечення приймає, перш за все, саму систему рівнянь, записану у вигляді

,

де функції – визначені, неперервні та ліпшециві у певній області інтегрування в . До цієї системи також додаються початкові умови :

Рівняння, область та початкові умови задаються користувачем.

Іншими словами, на вході система отримує задачу Коші для системи з *m* звичайних диференціальних рівнянь першого порядку.

Вихідними ж даними є послідовності наближень інтегральних кривих розв’язків кожного з рівнянь цієї системи, представлені у вигляді графіків цих наближень на кожній з ітерацій методу.

Для функцій , представлених у вигляді многочленів, користувач отримує коефіцієнт α стискаючого відображення (детальніше, див. розділ опису математичного забезпечення), аналіз збіжності методу, розрахунок необхідної кількості ітерацій методу для досягнення певної точності, а також аналітичний запис наближеного розв’язку на останній ітерації. Для інших типів функцій, користувач отримує лише графік кінцевого наближення та область збіжності методу, звісно, за умови, що метод збіжний для даної функції. Користувач також може визначити значення розв’язку у будь-якій точці області G.

# 2 ОГЛЯД ІСНУЮЧИХ МАТЕМАТИЧНИХ МЕТОДІВ ТА ПРОГРАМНИХ ЗАСОБІВ

2.1 Математичні методи розв’язання задачі Коші для системи з *m* звичайних диференціальних рівнянь

Почнемо з того, що нагадаємо формулювання задачі Коші для системи з *m* звичайних диференціальних рівнянь першого порядку, які розглядаються в рамках цієї роботи. Скористаємось визначенням, наданим у праці [1]:

Для заданої точки , необхідно знайти такі функції , що визначені, принаймні, в певному околі точки , та що задовольняють систему рівнянь:

За умов:

Або, у векторному варіанті: для заданої точки , знайти вектор-функцію , визначену, принаймні, в певному околі точки , та яка задовольняє рівняння:

За умови:

Надалі ми часто користуватимемось векторним записом у силу його компактності.

Існує багато методів для вирішення таких задач Коші, які можна розділити на три основні категорії:

* точні (або аналітичні) методи: вони дають результат у аналітичному вигляді.
* Чисельні (або табличні) методи, які дають результат у вигляді таблиці значень наближеного рішення задачі Коші
* та методи, які на виході дають результат у вигляді послідовності елементарних функцій. Ці методи також зазвичай називають «чисельними», але щоб уникнути двозначності, ми називатимемо їх «наближеними»

Чисельні методи також часто поділяють на два підкласи: «однокрокові» та «багатокрокові», залежно від того, скільки попередніх значень використовують для знаходження чергового наближення. Однокрокові методи, як можна здогадатися з назви, використовують лише одне наближене значення розв’язку (знайдене на попередньому кроці) для знаходження наступного наближення, тоді як багатокрокові методи – використовують декілька попередніх наближень.

Точні методи диференціальних рівнянь вивчаються у курсах теорії диференціальних рівнянь.

Серед чисельних методів однокрокових найпоширенішими є метод Ейлера, модифікований метод Ейлера та методи Рунге-Кутта, а до багатокрокових відносяться метод Адамса, метод забігання вперед та метод Мілна.

Наближені методи не є дуже поширеними через складність їх застосування та реалізації на ЕОМ, але серед них можна виділити деякі популярні методи, такі як метод Чаплигіна, метод узагальненого степеневого ряду та метод Пікара послідовних наближень.

Кожен з цих методів має свої переваги та недоліки, які ми розглянемо у наступних підрозділах.

2.1.1 Однокрокові чисельні методи

Метод Ейлера був, історично, найпершим і найпримітивнішим методом розв’язання задачі Коші для звичайних диференціальних рівнянь. На практиці, він фактично не використовується, адже як метод Ейлера, так і його модифікація : уточнений метод Ейлера мають дуже низький порядок точність – відповідно, 1-й та 2-й. Тим не менш, ці методи часто виявляються на диво корисними для доведення різних теорем, у тому числі, наприклад, теорему Пікара існування і єдиності розв’язку задачі Коші можна довести опираючись на [Колмогоров, Фомін].

Ці два методи легко виводяться, як в одновимірному [2], так і в багатовимірному випадках (при чому, використовуючи векторне представлення, доведення є таким же, як в одновимірному випадку), прямо із задачі Коші (2.3)-(2.4), описаної в попередньому підрозділі.

Обидва методи легко отримати із задачі Коші (2.3)-(2.4), як продемонстровано у [2]. Для цього спочатку розіб’ємо інтервал інтегрування на *l* рівних частин , довжиною *h*. І далі проінтегруємо рівняння (2.3) на інтервалі і отримаємо:

Одразу відмітимо, що під відніманням у лівій частині мається на увазі векторне (поелементне) віднімання від , а інтеграл у правій частині застосовується до кожного елемента вектор-функції .

Далі, чисельно наблизимо інтеграл у правій частині за методом прямокутників і отримаємо:

У якості початкового наближення береться умова (2.4).

Модифікований (або також «уточнений») метод Ейлера отримується із тієї ж формули . Тільки цього разу наближення інтеграла у правій частині шукається за методом трапецій, тоді:

Цікавою особливістю даного методу є той факт, що на кожному кроці необхідно обчислювати значення . Зазвичай, це роблять за допомогою звичайного метода Ейлера (саме тому цей метод – називається «уточненим»)

Іншим класом алгоритмів є методи Рунге-Кутта, що мають набагато вищу точність, порівняно з методами Ейлера (зазвичай, 3-го чи 4-го порядку, але можна вивести методи і значно вищих порядків) і саме тому це є, ймовірно, найбільш популярні методи розв’язання диференціальних рівнянь і їх систем на ЕОМ. Тим не менш, ці методи не без недоліків і за високу точність доводиться платити набагато більшим часом обчислень, у порівнянні з методами Ейлера.

Іншим підкласом чисельних методів є так звані методи Рунге-Кутта, що знову ж таки, отримуються з тієї ж формули (2.5), цього разу, наближуючи інтеграл за формулою Сімпсона. Зважаючи на те, що для застосування цієї формули наближеного обчислення інтеграла, необхідно мати три точки, то сам інтеграл рахується трохи інакше, а саме: вводиться проміжна точки , яка лежить посередині між точками та , що дозволяє переписати формулу (2.5) наступним чином:

І нарешті, ми можемо застосувати метод Сімпсона із кроком :

Отримали так зване «неявне» представлення, адже та у правій частині – невідомі. Формула (2.7) далі може бути використана для отримання різних методів Рунге-Кутта з різними порядками точності. Зокрема, в рамках цієї роботи, ми використовуємо метод Рунге-Кутта 4-го порядку точності для порівняння з методом Пікара. Детальніше, цей метод описано у розділі 3.

2.1.2 Багатокрокові методи

Як вже було відмічено, багатокрокові методи є іншим підкласом чисельних методів. Вони відрізняються від однокрокових методів, таких як Рунге-Кутта тим, що для знаходження наступного наближення, вони використовують знайдені значення розв’язку не тільки з останнього, а з декількох попередніх наближень.

Часто, це дозволяє отримати набагато більш точне наближення розв’язку за ту ж кількість ітерацій, адже на кожному кроці ці методи використовують більшу кількість інформації. Тим не менш, ці методи страждають від декількох недоліків. Одним з них є наприклад той факт, що декілька початкових наближень потрібно знаходити, користуючись іншими методами, як от, методом Рунге-Кутта.

Власне, в цьому розглянемо більш детально деякі з популярних багатокрокових методів.

І почнемо з методу Адамса, який є чи не найбільш поширеним багатокроковим методом, що опирається на формулу інтерполяції Ньютона назад [3]. Цей метод теж легко отримується з рівняння (2.5). Припустимо, як і раніше, що увесь наш інтервал інтегрування розбито на *l* підінтервалів і нехай ми маємо наближення розв’язку на перших *k* точках розбиття цього інтервалу , тоді запишемо знову рівняння (2.5)

І замінимо в ньому підінтегральну функцію на інтерполяційний многочлен Ньютона інтерполяції назад. Звідки ми й отримуємо формулу Адамса. Вона є відносно громіздкою у своєму загальному вигляді, тому тут наведемо лише формулу, коли нам відомо, наприклад, останні 3 наближення (а не *k*):

Подібним багатокроковим методом є також метод Мілна, який є досить поширеним та, у той же час, відносно простим у реалізації. Відмінність від методу Адамса тут полягає лише у тому, що замість інтерполяційного многочлена Ньютона для інтерполяції назад, тут використовується інтерполяційний многочлен Ньютона для інтерполяції вперед[3].

Щоб отримати основну формулу метода, проінтегруємо початкове рівняння (2.3) на інтервалі і отримаємо:

Підінтегральну функцію замінимо тоді таким виразом:

Це і є інтерполяційний многочлен Ньютона, інтерполяції «вперед», на сітці . Нагадаємо також, що у формулі , коефіцієнти обчислюються за допомогою кінцевих різниць *і*-го порядку для функції ***f****,* що мають вигляд (2.14).

Підставивши цей вираз у (2.12) та поклавши і , після відносно нескладних спрощень і розрахунків, отримаємо «першу формулу Мілна» (2.15).

Аналогічно, інтегруючи знову рівняння (2.3), тепер на інтервалі , й замінюючи у записі (2.13) інтерполяційного многочлена та , отримаємо тоді «другу формулу Мілна» (2.16).

Одразу відмітимо тут, що першу й другу формулу Мілна використовують, як правило, за схемою предиктор-коректор, відповідно (див., наприклад, [3] для деталей).

Є й багато інших різних багатокрокових методів, серед яких можна також відмітити метод «з забіганням вперед», але в більшості випадків, використовується точно така ж ідея, як і у всіх попередньо-розглянутих методах : проінтегрувати рівняння (2.3) на певному інтервалі та наблизити інтеграл у правій частині певною інтерполяційною схемою.

2.1.3 Наближені методи

Так звані «наближенні методи» фундаментально відрізняються від чисельних методів, що були розглянуті у попередніх підрозділах, у тому числі тим, що на виході вони дають, як правило, якусь комбінацію елементарних функцій (або їх послідовність, яка в решті решт збігається до точного розв’язку), що дає наближення розв’язку на інтервалі інтегрування. Перевагою таких методів є те, що ми можемо одразу отримати значення розв’язку у будь-якій обраній точці області збіжності, без необхідності здійснювати інтерполяцію табличних значень, як це потрібно робити при застосуванні чисельних методів.

Іншою безумовною перевагою є можливість відносно простого керування похибкою : у переважній більшості методів, достатньо лише прийняти до уваги оцінку похибки та обрати відповідну кількість ітерацій. Варто відзначити, що розглянуті до цього «чисельні» методи теж можуть бути уточнені за допомогою, наприклад, відомої процедури Рунге-Кутта. Її недоліком, тим не менш, залишається той факт, що проте вона потребує досить складних й громіздких обчислень.

На жаль, ці методи, як вже було відмічено, досить рідко використовуються на практиці через їх алгоритмічну складність та складність їх імплементації на ЕОМ; радше, вони представляють теоретичний інтерес. Тим не менш, в окремих випадках застосування цих методів є цілком практичним, як правило це означає, що праві частини рівнянь системи повинні мати певний спеціальний вигляд : наприклад, як ми побачимо далі у розділі 3, якщо усі праві частини представлені поліномами, то застосування методу Пікара послідовних наближень, який також є представником наближених методів, стає досить тривіальним, а обчислення – простими.

У цьому розділі, ми розглянемо деякі з найвідоміших «наближених» методів.

Почнемо з методу Чаплигіна, який базується на однойменній теоремі і хоч оригінально він застосовується для наближеного вирішення одновимірних ЗДР, його легко узагальнити на випадок системи диференціальних рівнянь [4]. Сама теорема заслуговує окремої роботи і є досить складної, щоб можна було її описати коротко, тому тут наведемо лише алгоритм роботи методу, який сам по собі є відносно простим (знову використовуватимемо векторне представлення системи для спрощення викладок):

1. Спочатку знаходяться дві вектор-функції та такі, що вони задовольняють рівняння (2.3) в певній точці за умови (2.4) і для яких на певному напівінтервалі (] справджується:

По суті, ці функції являють собою певного роду “бар’єри”, всередині яких лежить функція справжнього розв’язку . Таким чином, ці функції, виступають у ролі чергового наближення розв’язку.

1. Надалі ітераційна процедура будується таким чином: якщо нам відомі якісь наближення розв’язку та , то наступне наближення визначається за формулами:

Тут, – константа Ліпшиця для функції , адже, згідно умови, вона є Ліпшицевою.

Тут важливо підкреслити той факт, що так як метод Чаплигіна є фактично модифікацією методу Ньютона вирішення ЗДР, то швидкість збіжності може бути оцінена таким чином [5]: , для певного *n*, що відповідає кроку метода.

В решті решт, є також метод Пікара послідовних наближень, якому повністю і присвячена ця робота, цей наближений метод часто можна зустріти в літературі метод, хоча він, як уже неодноразово відмічалося, дуже рідко використовується на практиці. Він базується на ітеративній процедурі, яка прямо випливає із теореми Банаха про нерухому точку, яка також гарантує лінійну швидкість збіжності цього методу (іншими словами, збіжність «гарантована» за нескінченну кількість ітерацій). До поганої швидкості збіжності, серед недоліків, можна також віднести складність його реалізації на ЕОМ, адже на кожному кроці необхідно знаходити інтеграл правої частини диференціального рівняння, що може, у загальному випадку, приводити до надзвичайно складних квадратур, що можуть навіть і не виражатися у елементарних функціях (це ж саме, насправді кажучи, стосується і метода Чаплигіна й багатьох інших наближених методів, що покладаються на пряме інтегрування для знаходження чергового наближення)[5]. Більш детально цей метод розглядається у наступних розділах (зокрема, у розділі 3), тому не повторюватимемо ту ж саму інформацію тут.

Існує й багато інших наближених методів: можна навести у приклад метод узагальненого степеневого ряду, але він використовується зазвичай для спеціальних типів систем диференціальних рівнянь, тому не деталізуватимемо його тут.

2.2 Існуючі програмні засоби для вирішення задачі Коші для систем звичайних диференціальних рівнянь першого порядку

На сьогоднішній день, існує величезна кількість програмних продуктів для розв’язання, фактично, будь-яких математичних задач, що виникають на практиці, у тому числі і задач Коші для систем звичайних диференціальних рівнянь.

Ці інженерні системи реалізовані у багатьох різних форматах, але ймовірно найпопулярнішими залишаються різноманітні веб-сервіси такі як «Wolfram Alpha», що позиціонує себе як «науковий онлайн-калькулятор» та здатен розв’язувати дуже широкий клас задач як аналітичними, так і чисельними методами. Зокрема, у ньому реалізовано метод Рунге-Кутта для систем диференціальних рівнянь, а також, у відносно обмеженому варіанті, є функціонал для застосування методу Пікара для одновимірної задачі Коші.

Серед інших онлайн-сервісів вирішення диференціальних рівнянь та їх систем, можна навести такі відомі веб-застосунки, як «dcode.fr», «symbolab.com», «emathhelp.net» та багато інших.

Велика кількість таких застосунків пояснюється, звичайно, простотою та прямолінійністю імплементації чисельних методів, як метод Рунге-Кутта, на ЕОМ, що дають відносно точні результати, не вимагаючи при цьому значних обчислень.

Варто також відмітити появу новітніх рішень для розв’язання складних математичних задач (власне, у тому числі і задач Коші) за допомогою методів штучного інтелекту. Наприклад, за останні декілька років, з’явилась певна кількість систем обробки природної мови, таких як ChatGPT, що здатні розв’язувати примітивні системи диференціальних рівнянь у текстовому форматі. Існують також спеціалізовані системи штучного інтелекту, призначені для розв’язання диференціальних рівнянь, що опираються, серед іншого, на архітектуру ODE-RNN, яка використовує для обчислень той факт, що, як виявляється, існує пряма відповідність між рекурсивними диференціальними рівняннями та звичайними диференціальними рівняннями.

У той же час, програмних рішень, що б використовували метод ітерацій Пікара, доступних широкому загалу є неймовірно мала кількість, що, в свою чергу, легко пояснюється значною складністю реалізації цього методу на ЕОМ, а також з кількістю обчислень необхідних для знаходження розв’язку із достатньою точністю. У рамках пошуку подібних систем, у вільному доступі було знайдено буквально лише одну «демонстрацій» на базі «Wolfram Alpha», про яку уже було загадано раніше.

* 1. Висновки до розділу

За результатами цього розділу, було описано в грубих лініях основні сучасні методи вирішення задачі Коші для систем звичайних диференціальних рівнянь першого порядку. Також було проведено порівняльний аналіз їх головних переваг та недоліків, зокрема у Пікара.

Результати цього аналізу зібрані в таблиці 2.1.

Таблиця 2.1 – порівняння методів розв’язання задачі Коші для систем звичайних диференціальних рівнянь першого порядку

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Збіжність\* | Складність імплементації | Тип розв’язку |
| Точні методи | Гарантована за одну «ітерацію» | Висока | Аналітичний |
| Метод Ейлера |  | Дуже низька | Сітковий (Табличний) |
| Модифікований метод Ейлера |  | Дуже низька | Сітковий (Табличний) |
| Метод Рунге-Кутта | , де *n=4* як правило | Дуже низька | Сітковий (Табличний) |
| Метод Адамса | , де *n* залежить від кількості попередніх наближень | Низька | Сітковий (Табличний) |
| Метод Мілна | , де *n* залежить від кількості попередніх наближень | Низька | Сітковий (Табличний) |
| Метод Чаплигіна |  | Висока | Аналітичний |
| Метод Пікара | Лінійна | Висока | Аналітичний |

Ремарка\*, аналіз збіжності у даній таблиці, виконувався у припущенні того, що розв’язок існує, є стійким і розглядається у області збіжності.

Зрештою, основна увага у даній роботі, звісно, приділяється методу Пікара послідовних наближень. Основною перевагою цього методу безсумнівно є аналітичний вигляд розв’язку, тобто, як було відмічено, немає, наприклад, необхідності знаходити інтерполяційну функцію між вузлами сітки, щоб знайти значення розв’язку між значенням таблиці, як це робиться у табличних методах. Також, іншою перевагою, є простота уточнення розв’язку: достатньо просто збільшити кількість ітерацій методу. До недоліків можна віднести повільну збіжність: теорема Банаха про стискуючий оператор гарантує лише лінійну збіжність (до речі, у цьому, ми матимемо змогу переконатись у розділі 5). Іншим недоліком є також складність операції інтегрування, на якій будується ітеративна процедура, що значно ускладнює імплементацію цього методу на ЕОМ.

# 3 ОПИС МАТЕМАТИЧНИХ МЕТОДІВ

В цьому розділі, ми, спочатку, наведемо деякі з необхідних нам для подальших викладок понять та визначень, у тому числі для формулювання принципу Пікара-Банаха про нерухому точку, на якому фактично повністю і базується метод Пікара, що розглядається у рамках цієї роботи. У цих викладках ми також використовуватимо уже отримані результати, зокрема ті, що стосуються доведення збіжності методу Пікара [Самойленко, Колмогоров, Мій диплом, Моя стаття]. Після цього, ми пояснимо у чому, зі строго математичної точки зору, полягає модифікація методу Пікара, що пропонується нами для застосування на ЕОМ, а також доведемо її збіжність. Для неї ж буде отримано «глобальну» оцінку похибки та буде показано, як побудувати розв’язок на інтервалі, більшому за область збіжності. В решті решт, ми також пропонуємо прийом для економії обчислювальних ресурсів та оперативної пам’яті ЕОМ, при застосуванні згаданої модифікації.

3.1 Основні поняття з теорії диференціальних рівнянь

В цьому підрозділі, ми зазначимо лише найбільш важливі визначення та поняття з теорії диференціальних рівнянь, що знадобляться нам надалі. Ці та інші визначення запозичені із відомого підручника [1].

І почнемо ми із визначення того виду систем диференціальних рівнянь, які будуть розглядатися у цій роботі.

Нехай *D* – певна область в просторі і – є координати у даному просторі. Тоді, кожна вектор-функція визначає у цій області систему диференціальних рівнянь.

Відзначимо, що тут ми користуємося «векторним» позначенням, введеним у розділі 2, тобто : виділені жирним змінні та функції є, насправді, відповідно векторами та вектор-функціями. Дійсно, у випадку рівняння (3.1), воно може бути переписано у наступному вигляді.

Проте так як такий запис є досить громіздким, ми надаватимемо перевагу запису виду (3.1).

Перейдемо тепер до центрально означення даної роботи.

Визначення 3.1: диференційована у кожній точці певного інтервалу вектор-функція , що належить області *D* та яка задовольняє рівняння [систему] (3.1) у всіх точках інтервалу *І*, називається розв’язком рівняння (3.1) на цьому інтервалі.

Нагадаємо одразу, що під диференційованістю вектор-функції, мається на увазі диференційованість кожної компоненти цієї вектор-функції. Іншими словами, для того, щоб вектор-функція була диференційовною, кожна з її складових повинна бути диференційовною, тобто мати похідну.

Визначення 3.2: графіки розв’язку системи (3.1) називаються інтегральними кривими цієї системи.

Відзначимо тут, що як правило диференціальні рівняння та їх системи мають нескінченну кількість розв’язків, що представляються у вигляді певної параметричної сім’ї. Поняття задачі Коші, яке ми уже використовували в розділі 2, але яке ми введемо в наступному визначенні більш строго, покликано виділити з цієї нескінченної сім’ї розв’язків лише один розв’язок, або принаймні певну конкретну підмножину розв’язків, що задовольнятимуть задану початкову умову.

Визначення 3.3: Якщо на додачу до системи (3.1) також задано певні початкові умови.

(3.3)

Або, у компактному вигляді,

(3.4)

І де () є , то кажуть, що задано задачу Коші для системи (3.1) та з початковими умовами (3.4).

Власне, задача Коші полягає у знаходженні такого розв’язку цієї системи, що задовольняв би умову (3.4) та був би визначеним принаймні у деякому околі точки ().

З геометричної точки зору, задачу Коші можливо інтерпретувати, як задачу знаходження таких інтегральних кривих системи, що проходять через точку (), задану у умові (3.4).

В решті решт, введемо надважливе для методу Пікара (та й узагалі для теорії систем диференціальних рівнянь) визначення.

Визначення 3.4:Точка () із області називається точкою єдиності розв’язку задачі Коші (3.1)-(3.4), якщо для будь-яких двох розв’язків цієї задачі Коші, існує такий інтервал , на якому , і при чому

Визначення 3.5:Кажуть, що розв’язок задачі Коші є єдиним на певному інтервалі *І*, якщо кожна точка цього розв’язку на інтервалі *І* є точкою єдиності задачі Коші.

3.2 Основні поняття з теорії функціонального аналізу

У цьому розділі ми представимо необхідні поняття з функціонального аналізу, які потрібні для формулювання теореми Банаха. Ми також надамо доведення цієї теореми, оскільки сам метод послідовних наближень базується на ній. Для ознайомлення з визначеннями цього розділу можна звернутися до підручника [6].

Розгляд ми почнемо з метричного простору, який можна розглядати як узагальнення множини дійсних чисел. У метричному просторі визначена відстань між будь-якими двома елементами множини.

Означення метричного простору включає в себе пару (P, ρ), де P - певна множина (простір), а ρ(x, y) - невід’ємна та однозначна дійсна функція (метрика), що визначена для будь-яких двох елементів x та y з множини P. Якщо метрика задовольняє трьом аксіомам, то ми можемо говорити, що маємо справу з метричним простором.

Трійка аксіом, які повинна задовольняти метрика, включає: невід’ємність, симетричність та нерівність трикутника. Ці аксіоми необхідні для забезпечення відповідних властивостей метричного простору. Власне кажучи, вони формулюються наступним чином :

1) ρ(x, y) = 0 тоді і тільки тоді, коли x = y

2) ρ(*x, y*) = ρ(*y, x*)

3) ρ(*x, z*) ρ(*x, y*) + ρ(*y, z*)

Хоча простір дійсних чисел може служити прикладом метричного простору (у цьому дуже легко переконатися, скориставшись тим фактом, що остання аксіома, у випадку дійсних чисел – це нерівність трикутника), існує безліч інших множин, які можуть задовольняти аксіомам метричного простору. Наприклад, ми можемо розглянути простір елементами якого є вектори з m функцій, визначених, неперервних на певному сегменті і таких, що , де константу визначено з умови . Цей простір ми позначаємо і ще повернемося до нього у наступному підрозділі. Але поки що у рамках демонстрації, для цього простору можемо просто ввести метрику, що задовольняє аксіомам метричного простору :

(3.5)

Легко перевірити, що всі три аксіоми виконуються для такого вибору метрики. Детально ми зупинятися на цьому не будемо, але відзначимо, що це добре продемонстровано, наприклад, у підручнику [Колмогоров].

Далі введемо поняття околу елементу (іноді, «точки») метричного простору.

Визначення 3.6: сукупність точок певного метричного простору , які задовольняють умову:

Називають точки цього простору (також, її називають відкритою кулею, згідно геометричної інтерпретації цього об’єкту у просторі ) [7].

Нам також буде потрібно визначення збіжності у метричному просторі. Його у підручнику [Колмогоров] формулюють наступним чином:

Визначення 3.7: послідовність елементів певного метричного простору називають збіжною до елементу цього простору, якщо:

Ясно, що це визначення можна переформулювати також іншим чином: послідовність збігається до , якщо будь-який окіл точки містить усі точки послідовності , починаючи з деякої. Точку тоді називають границею .

Перейдемо до чергового важливого для нас поняття: повного метричного простору.

Визначення 3.8:послідовність точок з простору називають фундаментальною, якщо

Це є аналогом критерію Коші з дійсного випадку.

Із третьої аксіоми метричного простору випливає безпосереднім чином, що всяка збіжна послідовність – є фундаментальна. І дійсно, нехай маємо послідовність – що збіжна до *x*, тоді для заданого ми завжди можемо знайти такий номер N, що для будь-яких : та , і тоді

Однак, обернене твердження уже не завжди вірне, в чому не важко переконатися. Що приводить нас до наступного визначення.

Визначення 3.9:Метричні простори, в яких всяка фундаментальна послідовність є збіжна, називаються повними.

Доведемо, наприклад, що простір неперервних на відрізку функцій – є повний (що є, до речі, лише частковим випадком простору ).

Нехай, маємо повну фундаментальну послідовність цього простору, тоді, за визначенням, матимемо

Цілком ясно, що так як

А це, якраз, і є визначенням рівномірної збіжності функціональної послідовності. Тоді, із наслідку теореми про граничний перехід, а саме : «границя рівномірно збіжної послідовності функцій є неперервною функцією», слідує той факт, що границя *f* цієї послідовності є неперервна. Перейдемо тоді до границі при у попередній нерівності і матимемо:

Для усіх значень *t,* а отже і для максимумапри . Це й доводить збіжність послідовності у сенсі метрики цього простору.

Це й же результат легко узагальнюється на випадок простору , введеного трохи раніше, але викладки для нього є трохи більш громіздкими (хоч і мало у чому відрізняються від розглянутого випадку). Власне тому доведення для цього простору ми не наводитимемо тут.

Розглянемо тепер інші два дуже важливих визначення. Нехай *P* – метричний простір із ρ – метрикою у цьому просторі.

Визначення 3.10: Відображення *А* називають стискуючим, якщо знайдеться таке число α < 1, що для будь-яких двох векторів і цього простору *P*, виконується така нерівність:

Визначення 3.11: Елемент простору *P* називають нерухомою точкою відображення *А.* Інакше кажучи, нерухомі точки – це, по суті, розв’язки рівняння *.*

Теорема Пікара-Банаха, або принцип стискаючих відображень: Будь-яке стискаюче відображення, визначене у повному метричному просторі, має одну і лише одну нерухому точку.

Доведення*:* Для початку, відмітимо, що будь-яке стискаюче відображення є також і неперервним.

І справді, якщо , то за визначенням оператора стиску, вірно також, що .

Взявши далі довільну точку із даного метричного простору (повного) у якості початкової точки ітераційної процедури, побудуємо таку послідовність:

Видно, що побудована послідовність – є фундаментальна. Дійсно, припустивши, що n m, матимемо:

І так як , то взявши досить велике значення *n*, цю величину можемо зробити якзавгодно малою. Таким чином виходить, що послідовність й справді фундаментальна, а отже має границю у даному метричному просторі.

Наприклад, нехай

Тоді використовуючи той факт, що відображення А є неперервним, ми матимемо наступний ланцюг:

Який доводить існування нерухомої точки. Тепер достатньо довести єдиність такої цієї точки:

Якщо , то за визначенням стискаючого оператора:

І так як це означає, що єдине можливе значення для (отже ), тобто .

Теорема доведена.

3.3 Класичний Метод Пікара послідовних наближень для систем диференціальних рівнянь

Нарешті, ввівши усі необхідні нам визначення та надавши доведення теореми Пікара-Банаха, можемо перейти до самої суті цієї роботи – розгляду методу Пікара. І почнемо ми, звичайно, з короткого нагадування того, у чому полягає класичний метод Пікара послідовних наближень. Заодно, ми безпосередньо пояснимо у чому полягає дослідницька задача у даній роботі.

Отже, нехай маємо задачу Коші для системи рівнянь. У якості винятка, у цьому розділі користуватимемося «розлогим» записом, щоб спростити розуміння викладок і запишемо, власне, систему наступним чином:

За умов:

Нехай тепер знову – певна область, визначена в . Розглянемо визначені, неперервні та ліпшицеві у цій області функції , .

Одразу тут же нагадаємо, що умова Ліпшиця в випадку систем диференціальних рівнянь формулюється [Самойленко] наступним чином.

Вектор-функція називається локально ліпшицевою, щодо, якщо для кожної точки знайдуться такі числа та, що :

Тут – позначає -вимірну кулю, радіуса r із центром у точці .

Також відомо, що на певному інтервалі існуватиме єдиний розв’язок такої задачі Коші [2, с. 392].

Слідуючи [1, с. 80] і як уже було згадано, розглянемо простір , елементами якого є вектори з m функцій, визначених, неперервних на певному сегменті і таких, що , де константу визначено з умови .

Метрика у цьому просторі визначається наступним чином :

Як ми уже відзначали, цей метричний простір є повним і до того ж, наступний оператор є стискуючим оператором у цьому просторі :

(3.23)

Таким чином, згідно принципу Пікара-Банаха стискаючих відображень, ми можемо побудувати ітераційну процедуру знаходження наближеного розв’язку задачі Коші (3.20)-(3.21) :

Тут за допомогою – позначається черговий крок ітерації, а – звичайно, позначає індекс рівняння, для відповідного розв’язку.

Відзначимо також, що у якості можна брати абсолютно будь-які неперервні в D функції, але здебільшого беруться константи (часто хорошим початковим наближенням є , адже згідно початкової умови, розв’язки рівняння повинні проходити через ці точки, тому ми одразу починаємо «там де потрібно»).

Тут же одразу і видно, що, у випадку, коли функції є неполіноміальними, то інтегрування у ітераційній процедурі може привести до надзвичайно складних квадратур, які можуть навіть і не виражаються у елементарних функціях. Це, звичайно, значним чином ускладнює застосування цього методу на практиці, зокрема на ЕОМ. Саме тому, ми пропонуємо у цій роботі, модифікувати цей метод так, щоб уникнути інтегрування «складних» функцій та, в певному сенсі, обмежитися лише інтегруванням поліномів, що спрощує розрахунки на ЕОМ. Ця модифікація описана у наступному підрозділі.

## 3.4 Опис модифікація методу

Перейдемо тепер до опису модифікації методу Пікара послідовних наближень, який дозволяє спростити його застосування на ЕОМ і який є центральною частиною цієї роботи. У цьому розділі також використаємо певні з викладок, описаних у роботах [Колмогоров, моя стаття, мій диплом].

Отже, як вже було відмічено у попередньому розділі, наша мета полягає в уникненні повторного інтегрування складних функцій у процесі ітерацій класичного методу Пікара. Один зі способів вирішення цієї проблеми полягає в наближенні підінтегральних функцій поліномами. Для цього ми можемо застосувати формулу Тейлора для функцій однієї змінної до функцій , розклавши їх до k-го порядку в точці a (нехай розглядається інтервал [a, b]) та відкинути залишковий член. Припускається, що функції мають необхідну кількість частинних похідних по всім змінним. Для нульового наближення ми також використовуємо достатньо гладку функцію. Отже таким чином:

Уточнимо, що тут, під ми розуміємо повну n-ту похідну функції , як похідну складної функції аргументу , тобто наприклад

При заміні підінтегральних функцій їх частковими сумами ряду Тейлора, оператор (3.23), який для уникнення плутанини ми в подальших викладках позначатимемо як , приймає наступний вигляд :

Звісно, виникає питання про те, чи збігається метод при такому підході до наближення підінтегральної функції і, відповідно, до нової форми оператора. Ми розглянемо це питання детальніше у наступному розділі.

3.5 Збіжність модифікації методу Пікара

Для того, щоб довести збіжність цієї модифікації, ми покажемо, що оператор (3.27), при такій заміні підінтегральної функції, залишається оператором стиску.

Дійсно, розглянемо для цього відстань між та :

Тоді зафіксуємо значення n, і розглянемо отриману різницю під знаком суми.

Приймаючи до уваги той факт, що є, фактично, функціями багатьох змінних , то можемо застосувати до них теорему Лагранжа про скінченні прирости для випадку функцій багатьох змінних і для зручності запису, далі позначатимемо ):

Важливою відмітити, що – завжди визначене та існує, в сенсі максимума неперервної на відрізку функції.

Підберемо тепер кінець інтервалу інтегрування b таким чином, щоб виконувалась наступна умова:

Матимемо, тоді :

Так як кінець інтервалу інтегрування ми можемо вибирати на наш розсуд, то це дозволяє зробити .

Отже, маємо, що:

Де , це і доводить факт того, що оператор є стискуючим оператором.

## 3.6 Оцінка похибки модифікації методу

Хоч ми і довели, що метод збіжний, нам це, насправді, мало про що каже. Адже далі нам ще потрібно довести, що при застосуванні такої модифікації методу, вона дасть результат близький до того, який ми би отримали, якби не застосовували розкладення підінтегральної в ряд Тейлора.

Для цього, ми можемо оцінити різницю, у сенсі метрики простору , , між наближенням розв’язку системи, яке ми б отримали з класичним методом Пікара і наближенням, яке ми отримаємо про описаній модифікації методу.

Знову будемо позначати як оператор , де підінтегральну функцію розкладаємо у ряд Тейлора, тоді достатньо оцінити різницю між діями цих двох операторів і на одну і ту ж вектор-функцію .

Цілком зрозуміло, що так як різниця між операторами і полягає тільки в тому, що в одному з них підінтегральну функцію розкладають у ряд Тейлора, то різницею їх дії на одну і ту ж вектор-функцію буде інтеграл від залишкового члену ряду, тобто:

У цій формулі, ми можемо підібрати кінець інтервалу інтегрування *b* таким чином, щоб отримане значення було меншим за певне задане число

Тепер розглянемо :

Точно таким же чином, ми можемо отримати, що

І так далі до s-того наближення :

Зважаючи на те, що факторіал зростає швидше за будь-який скінченний поліном, то для забезпечення збіжності модифікації методу до справжнього розв’язку, достатньо просто розкладати підінтегральну функцію до «досить високого» порядку (звичайно, треба також зважати на вибір кінця інтервалу інтегрування b).

Отриману формулу, разом з похибкою метода Пікара, можемо вважати загальною похибкою застосування цього методу при розкладенні підінтегральної функції у ряд Тейлора.

Таким чином, нами доведено, що метод Пікара для систем диференціальних рівнянь, при розкладенні у ньому підінтегральної функції у ряд Тейлора, дійсно буде збіжним до справжнього розв’язку.

## 3.7 Прийом для спрощення розрахунків на ЕОМ

У цьому підрозділі, ми пропонуємо спеціальний прийом для спрощення розрахунків та економії оперативної пам’яті при застосуванні цього алгоритму на ЕОМ.

Приймаючи до уваги те, як працюють сучасні ЕОМ, при практичних розрахунках, не завжди зручно зберігати у пам’яті усі попередні наближення, які ми знайшли. Дійсно, для знаходження наступного наближення розв’язку системи, при реалізації цього алгоритму на ЕОМ, оператор застосовується до правих частин рівнянь не одночасно, а «по порядку», тобто спочатку знаходимо наступне наближення , далі і так далі аж до . Звичайно, ми припускаємо, що код не є паралелізованим, але паралелізація і векторизація цього алгоритму виходить за рамки даної роботи і потребує подальших досліджень

Власне, сам прийом опирається на ту ж ідею, що й метод Гауса-Зейделя, і полягає у тому, що так як для знаходження наступного наближення використовуються, власне кажучи, попередні наближення розв’язків, при знаходженні ми можемо використати уже знайдені наближення , замість . Це може трохи пришвидшити збіжність алгоритму, а також дозволяє ефективніше використовувати оперативну пам’ять ЕОМ.

До того ж, така модифікація є цілком обґрунтованою з точки зору збіжності самого алгоритму, адже як так і є поліномами (тобто належать ), а отже можемо застосувати до , а не до і так як є оператором стиску у цьому просторі , то алгоритм дійсно буде збіжним. Звичайно, тип збіжності так і залишиться лінійним, тобто нам усе ще теоретично необхідно було б зробити нескінченну кількість ітерацій, щоб отримати точний розв’язок, але емпірично при такій модифікації алгоритму, він швидше збігається до наближеного розв’язку із заданою точністю (див. результати у відповідному розділі).

## 3.8 Процедура спряження локальних розв’язків на краях області збіжності

Отже, в попередніх розділах нами було доведено збіжність модифікації методу Пікара, при заміні підінтегральних функцій їх рядами Тейлора, до справжнього розв’язку, але разом з тим, при доведенні нам довелося накласти певні додаткові умови на вектор-функції в правій частині системи. До того ж, у цьому випадку, збіжність методу Пікара значним чином залежить від вибору кінця інтервалу інтегрування. Це і не дивно, адже ряд Тейлора дає «непогане» наближення лише локально.

Зважаючи на це, ще раз трохи модифікуємо алгоритм методу, щоб прийняти до уваги ці обмеження та мати змогу інтегрувати систему на інтервалі більшому, ніж початкова область збіжності. Введемо процедуру спряження локальних розв’язків на краях області збіжності. Не обмежуючи загальності, ми можемо припустити, що початкові умови задано на початку інтервалу інтегрування, тобто , де інтервал інтегрування *І* – [*a, b*].

Розіб’ємо тоді інтервал *І* на *n* частин, які позначатимемо [], де відповідно . .

Далі розкладемо кожну компоненту вектор-функції у ряд Тейлора у точці (тобто там, де задана початкова умова) і знайдемо наближення розв’язку за ітераційною процедурою на першому із підінтервалів [], у припущенні, що інтервал розбито таким чином, що кінець інтервалу було підібрано так, що умови збіжності, отримані вище, були виконані.

Можемо тепер знайти наближене значення . І так як лежить на тих же інтегральних кривих, що і , то розв’язок задачі Коші для системи із початковими умовами (якщо виконані умови збіжності, тобто він існує і єдиний!) проходитиме через , тому можемо повторити всі ті ж самі викладки для точки . І так далі аж до . В результаті матимемо *n* вектор-функцій, які на відповідному інтервалі наближають одні й ті самі інтегральні криві.

Суть описаного алгоритму проілюструємо на рисунку 3.1. Для простоти зобразимо випадок з лише одним рівнянням:

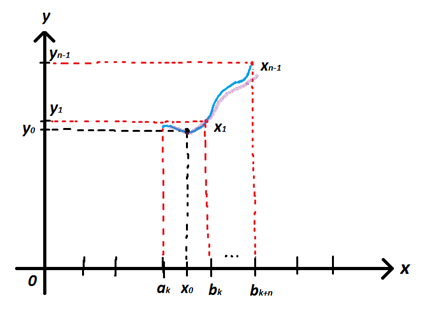


Рисунок 3.1 - Схема знаходження наближення розв’язку за методом Пікара при розкладенні підінтегральної функції у ряд Тейлора

Тут синім виділено наближення отримане при застосуванні розкладення у ряд Тейлора на кожному з підінтервалів, а фіолетовим – точну інтегральну криву.

Власне, варто зазначити, що чим більшим обрати *n* і чим до вищого порядку ми розкладатимемо підінтегральну функцію, тим точніше отримаємо наближення розв’язку задачі Коші на усьому інтервалі.

## 3.9 Метод Рунге-Кутта 4-го порядку

Ми уже частково розглядали методи Рунге-Кутта у розділі 2, проте ми включаємо більш детальний опис цього методу тут, адже він безпосередньо буде використаний у системі розв’язання систем диференціальних рівнянь, для порівняння результатів із методом Пікара.

Таким чином, з основною ідеєю методу ми уже ознайомилися, тепер отримаємо метод Рунге-Кутта 4-го порядку точності із неявної формули, отриманої у розділі 2, як це пропонується у роботі [2].

І почнемо з розгляду, власне, з цієї неявної формули (знову ж таки, називається вона неявною, адже нам невідомі значення ) :

Для простоти запису, позначимо .

Неявну частину можна обчислити наступним чином:

(3.37)

Отриманий метод можемо записати у вигляді такої ітеративної процедури:

Зважаючи на те, що метод Рунге-Кутта опирається на інтерполяційний метод Сімпсона, що у свою чергу має 4-й порядок точності відносно кроку, то і метод Рунге-Кутта матиме також 4-й порядок точності [детальніше, 2]. Метод дає досить точні результати, якщо обраний крок є відносно невеликим, тому, зважаючи на простоту реалізації цього методу, його було доцільно використати для порівняння результатів, отриманих за методом Пікара послідовних наближень, що і зроблено у даній роботі.

## 3.10 Висновки до розділу

За результатами цього розділу, було досліджено математичне забезпечення для системи розв'язання матричних диференціальних рівнянь методом Пікара послідовних наближень. В першу чергу, були розглянуті фундаментальні для викладок, поняття з теорії диференціальних рівнянь, такі як поняття розв'язку системи, задача Коші для систем, існування розв'язку та постановка задачі. Також, нами було розглянуто деякі з основних понять з теорії функціонального аналізу, у тому числі метричні простори, повні метричні простори, збіжність і фундаментальність у метричних просторах, доведення принципу Пікара-Банаха стискаючих відображень.

Основною суттю даного розділу, тим не менш, було введення модифікації класичного методу Пікара, шляхом заміни у ньому підінтегральних функцій, їх рядами Тейлора, що покликано уникнути інтегрування складних неполіноміальних функцій і таким чином спростити розрахунки на практиці, а зокрема на ЕОМ. Було доведено, що така модифікація буде збіжна і саме до справжнього розв'язку. Для цієї модифікації було також оцінено "глобальну" похибку. Таким чином, застосування на ЕОМ стало значно більш ефективним і простим.

Для подальшого спрощення розрахунків на ЕОМ був введений спеціальний прийом обчислення наступного. Було показано, що такий прийом дозволяє підвищити точність розрахунків, зменшити час їх виконання та зекономити оперативну пам’ять ЕОМ.

Ми також продемонстрували яким чином можливо провести спряжування розв'язків на краях області збіжності таким чином, щоб можна було проводити інтегрування систем диференціальних рівнянь за нашим методом не тільки у області збіжності, а й на інтервалах, більших за неї. Тим не менш, таке спряжування, звичайно, тягне за собою накопичення похибки, дослідження якої виходить за рамки цієї роботи.

На сам кінець, ми коротко описали метод Рунге-Кутта 4-го порядку точності, відносно кроку, для його подальшого порівняння з модифікованим методом Пікара.

Підводячи підсумки, результати даного дослідження є важливими як з практичної, так і з наукової точок зору. З одного боку, вони дозволяють зменшити час розрахунків на ЕОМ та зробити їх більш ефективними. А з іншого, представляють теоретичний інтерес з перспективи продовження досліджень, стосовно збіжності наближених методів розв’язання систем диференціальних рівнянь та оцінки їх похибок.

# 4 ОПИС ПРОГРАМНИХ ЗАСОБІВ

## 4.1 Опис розроблених алгоритмів

У цьому розділі будуть описані розроблені, на основі результатів отриманих у розділі 3, програмні засоби. Ми почнемо із загального огляду побудованої системи та розглядатимемо кожен з підмодулів у більших деталях у подальших розділах. Ми також наведемо певні теоретичні результати, які стосуватимуться алгоритмічної складності розробленого модифікованого алгоритму методу Пікара послідовних наближень. Насамкінець будуть наведені тестові випадки для, власне кажучи, тестування корректного функціонування системи, які будуть виконані та їх результати наведені у розділі 5 цієї роботи. Отже, перейдемо нарешті до розгляду системи.

Розроблена система представлена у вигляді веб-орієнтованого застосунку, розміщеного на безкоштовному остигну pythonenywhere. Користувач взаємодіє з даним застосунком за допомогою графічного інтерфейсу головної веб-сторінки застосунку, вводячи дані, як кількість рівнянь у системі, самі рівняння, початкові умови тощо у відповідні поля.

Користувач має змогу розв’язати введену систему диференціальних рівнянь або модифікованим методом Пікара, що був описаний в деталях у розділі 3, або методом Рунге-Кутта четвертого порядку точності (або, звичайно, обома методами для порівняння).

Веб-застосунок може бути проконсультований за наступною веб-адресою: <http://picardsolver.pythonanywhere.com/>

Також, на додачу до лістингу програми, наведеного у додатках, увесь програмний код доступний за наступною адресою: <https://github.com/vladherasymenko/PicardSolver>. Код отримуватиме регулярні оновлення саме на платформі GitHub, тому перевагу, при виборі вихідного коду для реімплементації на власній ЕОМ, варто надавати саме наведеному веб-посиланню на GitHub репозиторій.

Відмітимо одразу, що звичайно, основним функціоналом системи – є розв’язання задачі Коші для систем диференціальних рівнянь модифікованим методом Пікара, а метод Рунге-Кутта було імплементовано лише для порівняння з методом Пікара. Останній, тим не менш, фундаментально відрізняється від методу Пікара, адже він є табличним (також «сітковим») методом, у той час як метод Пікара – це метод «наближений», що дає розв’язок у вигляді певної аналітичної функції. Саме тому порівняння між двома методами пропонується проводити візуально, адже не існує достатньо інформативної метрики, яка могла б виміряти різницю точності між функцією і таблицею значень.

Переходячи до розгляду самих алгоритмів, реалізованих у рамках розробки системи, наведемо на рисунку 4.1 загальну схему функціонування системи, а у подальших підрозділах розглянемо кожну з підсистем більш детально.

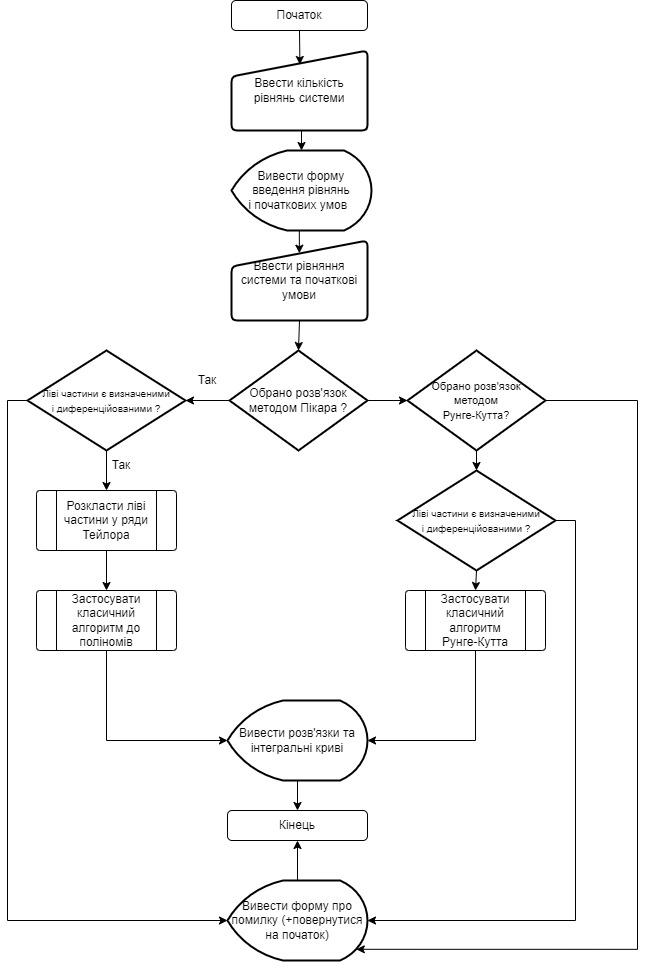


Рисунок 4.1 - Загальний алгоритм функціонування системи

Далі детальніше розглянемо кожну із розроблених підсистем. Почнемо із способу знаходження похідних, що використовується у роботі, після чого перейдемо до розгляду модуля розкладання функцій в ряди Тейлора, адже на нього опирається модифікований алгоритм методу Пікара.

### 4.1.1 Знаходження похідних

Фундаментально, є два основних підходи до програмного знаходження похідних диференційованих функцій: символьний та чисельний [Introduction to Numerical Methods in Differential Equations" авторства Mark H. Holmes].

Символьне диференціювання полягає в знаходженні аналітичної формули для похідної функції. Це може бути корисно, коли потрібно отримати точну формулу для подальшого аналізу, або коли потрібно отримати точні значення похідної для всіх точок в заданому діапазоні. Пакунки для символьного диференціювання, такі як scipy або diff, використовують символьні маніпуляції для обчислення похідної функції.

Однак символьне диференціювання має свої недоліки. Деякі складні функції можуть бути важко або навіть неможливо аналітично диференціювати. Крім того, символьне диференціювання може бути дуже обчислювально витратним, особливо для складних функцій з багатьма змінними.

З іншого боку, чисельне диференціювання полягає в апроксимації похідної функції, використовуючи числові методи. Це зазвичай швидше, ніж аналітичне диференціювання, і може бути використане для обчислення похідної функції в будь-якій точці з заданою точністю. Наприклад, можна використовувати метод скінченних різниць, який ми і використовуємо у даній роботі, щоб обчислити похідну функції у точці, наблизивши його до лівого та правого сусідів.

Незважаючи на ці переваги, чисельне диференціювання має свої недоліки. Чисельні апроксимації можуть бути неточними, особливо якщо функція має сильні зміни у своїй похідній. Крім того, чисельне диференціювання є операцією погано визначеною, особливо для функцій, які мають велику похідну або які є неперервними. Це може призвести до помилкових результатів або недооцінки точності. Тому, перед використанням чисельних методів, необхідно ретельно вивчити їх характеристики та обмеження, щоб забезпечити точність і надійність результатів.

У підсумку, обидва методи диференціювання мають свої переваги та недоліки, і їх вибір залежить від потреб користувача та особливостей задачі. Як правило, символьне диференціювання корисне для отримання точних аналітичних формул, тоді як чисельне диференціювання корисне для швидкого і точного обчислення значень похідних у конкретних точках.

Тим не менш, наш вибір зупинився, як було уже сказано, на числьному диференціюванні і його реалізовано у програмі за допомогою готового модулю scipy.misc.derivative(), який використовує класичну різницеву схему.

На рисунку 4.2 наведемо спрощений алгоритм знаходження похідної.

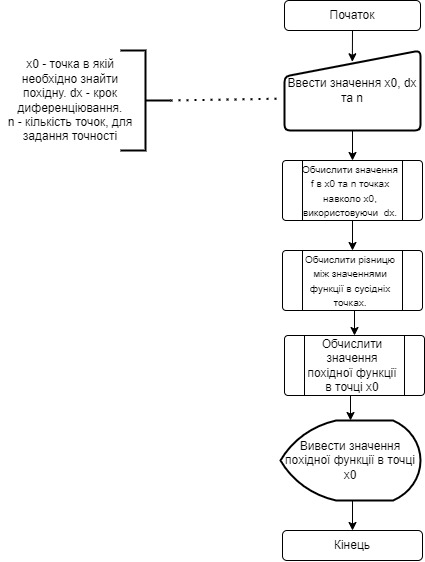


Рисунок 4.2 - Спрощений алгоритм різницевої схеми

Зазначимо ще, що функція scipy.misc.derivative() дозволяє одразу знайти похідну *n*-uj порядку, що буде дуже корисно для нас при реалізації алгоритму розкладення у ряд Тейлора.

Варто також відзначити, що безсумнівною перевагою цієї функції є те, що вона дозволяє знайти чисельне значення похідної від, фактично будь-якої диференційовної у заданій точці функції, адже вона опирається на вбудовану функцію eval() мови програмування python, яка дозволяє знайти чисельне значення будь-якого виразу чи будь-якої функції у заданій точці, що є єдиною вимогою до функціонування різницевої схеми.

### 4.1.2 Розкладення в ряд Тейлора

Модуль розкладу заданої функції у ряд Тейлора (або, вірніше, модуль знаходження перших *n* членів ряду Тейлора) імплементовано у коді функцією to\_taylor\_series(), яка отримує на вхід рядок-функцію для розкладу, точку розкладу, порядок розкладу, а також попередні поліноміальні наближення для , які одразу підставляються замість відповідних значень.

Перед використанням цієї функції, тим не менш, потрібно знати, як працює функція factorial(), яка необхідна для знаходження факторіалу натурального числа.

Власне, на рисунку 4.3 наведено блок-схему алгоритму роботи функції factorial(). Вона є допоміжною функцією, яка використовується при обчисленні формули Тейлора.

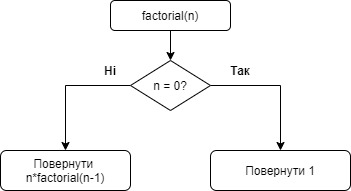


Рисунок 4.3 – Cхема роботи алгоритму знаходження факторіалу

Таким чином, це просто класичний рекурсивний алгоритм для знаходження факторіалу заданого натурального числа.

Нарешті, можемо безпосередньо перейти до розгляду модуля, що реалізовує розклад заданої функції (правої частини диференціального рівняння системи) у ряд Тейлора. Ясно, що даний модуль повністю базується на відомій формулі Тейлора (4.1) для розкладу функції у точці *x0* в степеневий ряд:

Звичайно, неможливо повныстю представити нескінчений стпеневий ряд на ЕОМ, тому у рамках даної роботи, ми обмежуємося першими *k* членами цього ряду та відкидаємо його залишковий член. Таким чином:

Маючи «на руках» функції обчислення похідних та факторіалів, нескладно імплементувати формулу (4.2) програмно. Власне, наведемо на рисунку 4.4 блок-схему алгоритму функції, що і реалізовує обчислення цієї формули.

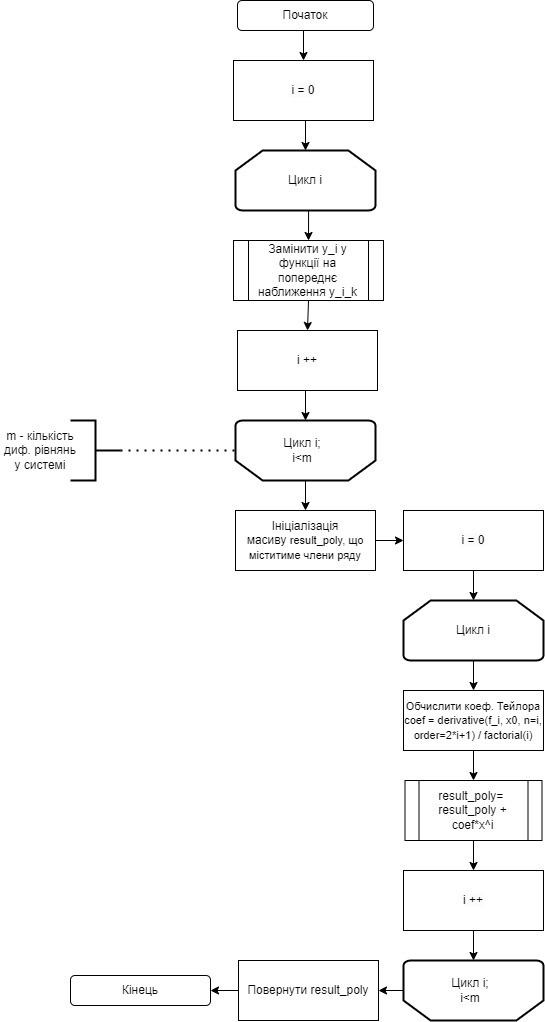


Рисунок 4.4 - Блок-схема алгоритму роботи функції знаходження перших *n* членів ряду Тейлора

### 4.1.3 Розв’язання задачі Коші для систем звичайних диференціальних рівнянь першого порядку

Як було показано на блок-схемі алгоритму роботи розробленої системи на самому початку цього розділу, у даному веб-додатку імплементовано два методи розв’язання задачі Коші для систем звичайних диференціальних рівнянь першого порядку, а саме модифікована процедура Пікара-Банаха, а такод метод Рунге-Кутта 4-го порядку точності для порівняння результатів

Наведемо ще раз запис ітеративної процедури Пікара, що була отримана у розділі 3:

Тут – це *j*-те наближення інтегральної кривої *i-*го диференціального рівняння системи. У якості можна брати будь-яку неперервну функцію на інтервалі *I* інтегрування. Як уже було обговорено, часто у якості доцільно брати константу з початкової умови () задачі Коші для системи.

З самого запису (4.3) добре видно і практичне правило реалізації цього алгоритму, яке полягає у інтегруванні функцію на підінтервалі [], тобто інтегрування правої частини даного диференціального рівняння, де замість підставляється останнє наближення розв’язку.

Видно також, що найважчою частиною тут є необхідність інтегрування функції . Проте, інтегрування поліномів відносно просто імплементувати програмно і, власне, саме у наближенні підінтегральних функцій поліномами і їх подальшому інтегруванні і полягає модифікація методу Пікара, що розглядається в рамках даної роботи, як описано у розділі 3.

Отже, ми вже маємо модуль to\_taylor\_series(), що дозволяє наблизити підінтегральну функцію у правій частині (4.3) поліномом. Залишається лише розглянути сам процес інтегрування, який, звичайно, реалізовано окремою функцією. Інтегрування поліномів є досить простою операцією алгоритмічно, тому можемо обійтися без використання чисельних методів, як метод трапецій чи Сімпсона, а одразу імплементувати . До того ж, так як результатом інтегрування полінома є інший поліном, а, власне кажучи, поліноми (з натуральними степенями) можна програмно цілком представити одним масивом його коефіцієнтів, то наш модуль інтегрування оперуватиме не символьними представленнями, а масивами, що значно спрощує написання коду та обчислення, уникає необхідності парсингу, а також економить оперативну пам’ять. Дійсно, наприклад, поліном можемо представити у вигляді масиву , а тоді його квадратура запишеться як , масив якох відповідно буде .

Для ясності наведемо ще блок-схему алгоритму модуля інтегрування на рисунку 4.5.

Рисунок 4.5 - Блок-схема алгоритму інтегрування для поліномів

Перейдемо до розгляду другого алгоритму. Нагадаємо, що суть його полягає у тому, що увесь інтервал інтегрування , для початку, розбивається на *k* менших інтервалів .

Далі, за методом Пікара, знаходиться наближення розв’язку диференціального рівняння на тому з підінтервалів, який містить точку з початкової умови. Для визначеності, нехай це буде інтервал []. І позначимо розв’язок, який ми отримаємо на цьому інтервалі, як ).

При чому, щоб отримати це наближення, на кожному кроці методу Пікара, функція розкладається у ряд Тейлора у точці , щоб отримати поліном, який можна проінтегрувати функцією integrate\_polynomial(). Зрозуміло, що це наближення інтегральної кривої буде досить неточним, адже до похибки методу додасться похибка розкладу функції у ряд Тейлора (детальніше див. доведення у розділі 3.3.3). Проте, чим більше взяти підінтервалів, тим менш суттєвою буде ця похибка, бо тим меншим буде кожен інтервал . У реалізованій системі, з міркувань оптимізації швидкості обчислень, взято 15 підінтервалів, а функція розкладається у ряд Тейлора до 4 степені, тому метод може бути не завжди збіжним.

Після цього, повторюємо усе те ж саме для інтервалу справа: [], якщо такий є. Тільки змінюємо початкову умову з на таку:

Так як на інтервалі , то припускаючи, що він лежить у області існування та єдиності задача Коші та те, що розв’язок – асимптотично стійкий, отримаємо, що:

Саме тому ми і беремо це у якості початкової умови.

Продовжуємо так далі аж поки не дійдемо до останнього інтервалу = . Після чого знову повторюємо те ж саме, тільки рухаючись від інтервалу у напрямку наліво, взявши, спочатку у якості початкової умови:

Важливим моментом, тим не менш, є те, що обумовленим такий алгоритм є лише у області збіжності методу, тобто у області існування і єдиності розв’язку задачі Коші, адже за її межами, навіть немає гарантій існування інтегральної кривої.

Область збіжності знаходиться у програмі, згідно формул, виведених у теоретичній частині. Варто відмітити, що у деяких з цих формул необхідно знаходити максимум певної функції. Це реалізовано програмно за допомогою оптимізаційного методу спряжених напрямів Пауелла, імплементованого у бібліотеці scipy для мови Python. Обрано цей метод було радше емпірично: для функцій, які тестувалися у даній роботі, саме цей метод давав результат, відносно, швидко, а найголовніше – рідше, ніж інші методи зупинявся у локальних мінімумах. Тим не менш, оптимізація функцій у розробленій системі, як правило, займає набагато менше часу, ніж процес знаходження інтегральних кривих, тому можна було б обрати і будь-який інший сучасний метод, як-от BFGS.

Додамо насамкінець, що процедура спрощення виразів, яка зустрічаються майже у всіх наведених блок-схемах – представляє собою комбінацію символьного множення поліномів (як-от перемноження дужок між собою) та бінома Ньютона для розкриття степенів. Ці алгоритми є загальновідомими, тому недоцільно їх наводити тут. За необхідності, читач може ознайомитися з їх програмною реалізацією у лістингу програми – це функції symbol\_multiplication(), term\_multiplication(), sum\_up() та binomial(); вони комбінуються у функції prepare\_polynomial().

Рисунок 4.6 - Блок-схема алгоритму методу Пікара для поліномів

## 4.2 Оцінки алгоритмічної складності модулів

Алгоритмічна складність коду для знаходження розв'язку системи диференціальних рівнянь модифікованим методом Пікара залежить від кількості ітерацій, тобто значення параметру num\_iters, та від кількості членів у поліномах, тобто значення параметру taylor\_order.

Оцінимо алгоритмічну складність функції integrate\_linear - ця функція складається з циклу з len(coefs) ітерацій, тобто має складність O(n), де n = len(coefs).

Функція list\_to\_formula складається з циклу з len(coefs) ітерацій, тому її складність також O(n), де n = len(coefs).

Алгоритм множення поліномів multiply\_polys має вкладений цикл, тому його складність становить O(n^2), де n - довжина списку poly\_list1.

Функція poly\_to\_power множить поліном на самого себе power разів, тому її складність - O(n^2 \* k), де n - довжина списку poly\_list, а k - значення параметру power.

Функція add\_arr має складність O(n), де n - довжина списку a або b, тому що вона містить лише один цикл.

Алгоритм обчислення факторіала factorial також має складність O(n), де n - значення параметру n.

Алгоритм обчислення значення ряду Тейлора у функції to\_taylor\_series має складність O(n^2), де n - значення параметру n. Використовувані тут функції custom\_func та derivative мають складність, яка залежить від точності обчислення похідних.

Отже, алгоритмічна складність коду для знаходження розв'язку системи диференціальних рівнянь модифікованим методом Пікара заміненої підінтегральних функцій поліномами залежить від кількості ітерацій, кількості членів у полінома.

Алгоритмічна складність даного коду складається з декількох компонент:

Обчислення поліномів - у функції multiply\_polys і poly\_to\_power використовується поділ і множення масивів, що має алгоритмічну складність O(n^2), де n - довжина масиву.

Обчислення інтегралів - у функції integrate\_linear обчислюється сума n елементів, де n - ступінь полінома, що має алгоритмічну складність O(n).

Обчислення похідних - у функції to\_taylor\_series використовується функція derivative з бібліотеки scipy.misc, яка виконує обчислення чисельного значення похідної функції, що має алгоритмічну складність O(n^2), де n - порядок похідної.

Обчислення значень функцій - у функції custom\_func використовується функція eval, яка виконує обчислення значення функції, що має алгоритмічну складність O(n), де n - довжина рядка.

Загальна алгоритмічна складність функції picard\_general залежить від кількості ітерацій, необхідних для досягнення потрібної точності. Зазвичай, кількість ітерацій залежить від властивостей функції і початкових умов.

Теоретично, складність функції picard\_general може бути визначена як O(n^2) або O(n^3), де n - кількість ітерацій, що необхідні для досягнення потрібної точності. Однак, на практиці, складність може бути нижчою, якщо метод збігається швидше, або вищою, якщо потрібна більш висока точність.

## 4.3 Тестування програмного забезпечення

У даному розділі наведено набір тестових випадків, що покликані якнайкраще покрити код програми, щоб забезпечити її безперебійну та коректну роботу.

Тестові випадки наведені у таблицях нижче. При чому, у кожному розділі, спочатку розміщено тестові випадки для коректних вхідних даних, а в кінці – для некоректних.

### 4.3.1 Диференціальні рівняння

Таблиця 4.1 – тестовий випадок Д.Р. 1

|  |  |
| --- | --- |
| Номер | Д.Р. 1 |
| Назва | Розв’язати задачу Коші методом Пікара, з заданими початковими умовами та інтервалом інтегрування |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле коректний запит. Наприклад, «Метод Пікара y' = x^3 + y + 1/2 від -2 до 2, x0 = 0, y0 = 5» | Запит відображається у полі |
| Натиснути кнопку « = » | Завантажується сторінка з розв’язком та аналізом точності |

Таблиця 4.2 – тестовий випадок Д.Р. 2

|  |  |
| --- | --- |
| Номер | Д.Р. 2 |
| Назва | Розв’язати задачу Коші методом Пікара, без задання інтервалу інтегрування |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле коректний запит, але не вказувати інтервал інтегрування. Наприклад, «Метод Пікара y' = x^3 + y + 1/2 x0 = 0, y0 = 5» | Запит відображається у полі |
| Натиснути кнопку « = » | Завантажується сторінка з розв’язком та аналізом точності. Інтервал інтегрування автоматично встановлено рівним [-1, 1] |

Таблиця 4.3 – тестовий випадок Д.Р. 1

|  |  |
| --- | --- |
| Номер | Д.Р. 3 |
| Назва | Розв’язати диференціальне рівняння методом Пікара на задному інтервалі, не вказуючи початкову умову |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |

Продовження таблиці 4.3

|  |  |
| --- | --- |
| Кроки | Очікуваний результат |
| Увести в головне поле запит, що не містить початкової умови. Наприклад, «Метод Пікара y' = x^3 + y + 1/2 від -5 до 5» | Запит відображається у полі |
| Натиснути кнопку « = » | Завантажується сторінка з розв’язком та аналізом точності. Початкова умова встановлена рівною y(0) = 1 |

Таблиця 4.4 – тестовий випадок Д.Р. 4

|  |  |
| --- | --- |
| Номер | Д.Р. 4 |
| Назва | Розв’язати диференціальне рівняння методом Пікара, без задання початкової умови і без вказання інтервалу інтегрування |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле запит, що не містить початкової умови і інтервалу. Наприклад, «Метод Пікара y' = x^3 + y + 1/2» | Запит відображається у полі |
| Натиснути кнопку « = » | Завантажується сторінка з розв’язком та аналізом точності. Початкова умова встановлена рівною y(0) = 1, а інтервал інтегрування рівним – [-1, 1] |

Таблиця 4.5 – тестовий випадок Д.Р. 5

|  |  |
| --- | --- |
| Номер | Д.Р. 5 |
| Назва | Розв’язати диференціальне без вказання методу та інтервалу |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле запит, що містить лише диф. рівняння. Наприклад, «y' = x^3 + y + 1/2» | Запит відображається у полі |
| Натиснути кнопку « = » | Завантажується сторінка з розв’язком та аналізом точності. Початкова умова встановлена рівною y(0) = 1, а інтервал інтегрування рівним – [-1, 1], метод розв’язання за замовчуванням – метод Пікара. |

Таблиця 4.6 – тестовий випадок Д.Р. 6

|  |  |
| --- | --- |
| Номер | Д.Р. 6 |
| Назва | Розв’язати задачу Коші методом Рунге-Кутта |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле запит з вказанням методу Рунге-Кутта. Наприклад, «Метод Рунге y' = x^3 + y + 1/2 від -1 до 2, x0 = -1, y0 = 2» | Запит відображається у полі |

Продовження таблиці 4.6

|  |  |
| --- | --- |
| Кроки | Очікуваний результат |
| Натиснути кнопку « = » | Завантажується сторінка з розв’язком, наведено графік наближеної інтегральної кривої |

Таблиця 4.7 – тестовий випадок Д.Р. 7

|  |  |
| --- | --- |
| Номер | Д.Р. 7 |
| Назва | Розв’язати задачу Коші методом Рунге-Кутта та Пікара одночасно |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле запит з вказанням методів Рунге-Кутта та Пікара. Наприклад, «Метод Пікара та Рунге y' = x^3 + y + 1/2 від -1 до 2, x0 = -1, y0 = 2» | Запит відображається у полі |
| Натиснути кнопку « = » | Завантажується сторінка з розв’язком, наведено графіки наближених інтегральних кривих, отриманих двома методами |

Перед цим тестовим випадком нагадаємо, що для розв’язання диф. рівнянь, заданих не поліномами у методі Пікара використовується трохи змінений алгоритм.

Таблиця 4.8 – тестовий випадок Д.Р. 8

|  |  |
| --- | --- |
| Номер | Д.Р. 8 |
| Назва | Розв’язати нелінійне диф. рівняння методом Пікара |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле нелінійне диф. рівняння. Наприклад, «y' = 2\*sin(x\*y)» | Запит відображається у полі |
| Натиснути кнопку « = » | Завантажується сторінка з розв’язком, наведено графік наближеної інтегральної кривої. |

Таблиця 4.9 – тестовий випадок Д.Р. 9

|  |  |
| --- | --- |
| Номер | Д.Р. 9 |
| Назва | Ввести некоректні дані |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести набір символів. Наприклад, «1а№пи».Натиснути кнопку « = » | Отримано повідомлення про неможливість інтерпретації вводу |
| Увести набір символів, замість диф. рівняння. Наприклад, «y' =r231@».Натиснути кнопку « = » | Отримано повідомлення про неможливість інтерпретації вводу |

Продовження таблиці 4.9

|  |  |
| --- | --- |
| Кроки | Очікуваний результат |
| Увести набір символів, замість меж інтервалу. Наприклад, «Метод Пікара та Рунге y' = x^3 + y + 1/2 від –@ до @, x0 = -1, y0 = 2».Натиснути кнопку « = » | Отримано повідомлення про неможливість інтерпретації вводу |
| Увести набір символів, замість початкової умови. Наприклад, «Метод Пікара y' = x^3 + y + 1/2 від 0 до 1, x0 = a, y0 = k»*.* Натиснути кнопку « = » | Отримано повідомлення про неможливість інтерпретації вводу |

Таблиця 4.10 – тестовий випадок Д.Р. 10

|  |  |
| --- | --- |
| Номер | Д.Р. 10 |
| Назва | Ввести диф. рівняння, для якого не виконується умова Ліпшиця |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле диф. рівняння для якого не виконується умова Ліпшиця. Наприклад, *«y' =1/x».* Натиснути кнопку « = » | Отримано повідомлення про розбіжність методу та невиконання умови Ліпшиця. |

Таблиця 4.11 – тестовий випадок Д.Р. 11

|  |  |
| --- | --- |
| Номер | Д.Р. 11 |
| Назва | Точка x0 лежить не на межі інтервалу інтегрування у методі Рунге-Кутта |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле запит, з вказанням методу Рунге-Кутта. При чому, точка лежить всередині інтервалу. Наприклад, «Метод Рунге y' = sin(y) від 0 до 1, x0 = 0.5, y0 = 1». Натиснути кнопку « = » | Отримано повідомлення про неможливість застосування методу Рунге-Кутта |
| Увести в головне поле запит, з вказанням методу Рунге-Кутта та методу Пікара. При чому, точка лежить всередині інтервалу. Наприклад, «Метод Пікара та Рунге y' = sin(y) від 0 до 1, x0 = 0.5, y0 = 1». Натиснути кнопку « = » | Отримано розв’язок методом Пікара та повідомлення про неможливість застосування методу Рунге-Кутта |

Таблиця 4.12 – тестовий випадок Д.Р. 12

|  |  |
| --- | --- |
| Номер | Д.Р. 12 |
| Назва | Точка x0 не належить інтервалу інтегрування |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле запит, де точка x0 лежить поза межами інтервалу інтегрування. Наприклад, «Метод Пікара y' = sin(y) від 0 до 1, x0 = -5, y0 = 1». Натиснути кнопку « = » | Отримано повідомлення про те, що точка x0 з початкової умови повинна лежати в межах інтервалу інтегрування |

Таблиця 4.13 – тестовий випадок Д.Р. 13

|  |  |
| --- | --- |
| Номер | Д.Р. 13 |
| Назва | Початок інтервалу інтегрування співпадає з кінцем або перевищує його |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле запит, де інтервал інтегрування задано некеректно. Наприклад, «Метод Пікара y' = sin(y) від 10 до 0, x0 = -5, y0 = 1». Натиснути кнопку « = » | Отримано повідомлення про некоректно заданий інтервал |

Продовження таблиці 4.13

|  |  |
| --- | --- |
| Кроки | Очікуваний результат |
| Увести в головне поле запит, де довжина інтервалу інтегрування по рівна нулю. Наприклад, «Метод Пікара y' = sin(y) від 1 до 1, x0 = -5, y0 = 1». Натиснути кнопку « = » | Отримано повідомлення про некоректно заданий інтервал |

### 4.3.2 Похідні

Цей розділ, звісно, не відноситься до основної теми роботи, проте обчислення деяких типів похідних – це одна з функціональних можливостей системи, яку теж варто протестувати. Підтримуються лише похідні поліноміальних функцій та таких, що містять sin, cos, log або e.

Таблиця 4.14 – тестовий випадок П.1

|  |  |
| --- | --- |
| Номер | П. 1 |
| Назва | Знайти похідну елементарної функції, різних порядків |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле запит для знаходження похідної. Наприклад, «Похідна f = 10\*x^2».Натиснути кнопку « = » | Отримуємо запис та графік похідної |

Продовження таблиці 4.14

|  |  |
| --- | --- |
| Кроки | Очікуваний результат |
| Увести в головне поле запит для знаходження похідної, що містить y. Наприклад, «Похідна f = 10\*x^2\*y».Натиснути кнопку « = » | Отримуємо запис та графік похідної |
| Увести в головне поле запит для знаходження похідної, що містить sin(), cos, e, log. Наприклад, «Похідна f = sin(y\*x)+log(2\*x\*y)+e^(log(2\*x))».Натиснути кнопку « = » | Отримуємо запис та графік похідної |
| Увести в головне поле запит для знаходження похідної порядку 3. Наприклад, «Похідна порядку 2 по x f = x^3\*y + 10\*y^5\*sin(x^3\*y)».Натиснути кнопку « = » | Отримуємо запис та графік похідної |

Таблиця 4.15 – тестовий випадок П.2

|  |  |
| --- | --- |
| Номер | П. 2 |
| Назва | Знайти частинну похідну функції по у, різних порядків |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле запит для знаходження частинної похідної. Наприклад, «Похідна по у f = 10\*x^2\*y».Натиснути кнопку « = » | Отримуємо запис та графік похідної |

Продовження таблиці 4.15

|  |  |
| --- | --- |
| Кроки | Очікуваний результат |
| Увести в головне поле запит для знаходження частинної похідної порядку 2. Наприклад, «Похідна по у порядку 2 f = 10\*x^2\*y».Натиснути кнопку « = » | Отримуємо запис та графік похідної |

Таблиця 4.15 – тестовий випадок П.3

|  |  |
| --- | --- |
| Номер | П. 3 |
| Назва | Знайти похідну в точці |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести в головне поле запит для знаходження похідної в точці. Наприклад, «Похідна порядку 2 по x f = x^3\*y + 10\*y^5\*sin(x^3\*y) в точці (10,1)».Натиснути кнопку « = » | Отримуємо запис та графік похідної |

Таблиця 4.16 – тестовий випадок П.4

|  |  |
| --- | --- |
| Номер | П. 4 |
| Назва | Ввести некоректні дані для знаходження похідної |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Увести набір символів, замість порядку похідної. Наприклад, «Похідна порядку @ f = 10\*x».Натиснути кнопку « = » | Отримано повідомлення про неможливість інтерпретації вводу |
| Увести набір символів, замість функції. Наприклад, «Похідна f = 123abc».Натиснути кнопку « = » | Отримано повідомлення про неможливість знаходження похідної |
| Увести набір символів, замість точки, у якій необхідно знайти похідну. Наприклад, «Похідна f = 10\*x\*y в точці (1, a)».Натиснути кнопку « = » | Отримано повідомлення про неможливість інтерпретації вводу |

### 4.3.3 Ряди Тейлора та Маклорена

Так, як і похідні з минулого розділу, розклад в ряди – це лише одна з підзадач, яка виникає при застосуванні методу Пікара і не є основною темою даної роботи. Проте їх варто протестувати, як частину системи.

Таблиця 4.17 – тестовий випадок Р.Т.1

|  |  |
| --- | --- |
| Номер | Р.Т. 1 |
| Назва | Знайти розклад елементарної функції в ряд Тейлора |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Ввести у головному полі команду для розкладу в ряд Тейлора. Наприклад: «Ряд Тейлора до степені 5 в точці -1 f = sin(cos(x))\*x» | Отримуємо запис ряду |
| Ввести у головному полі команду для розкладу в ряд Тейлора без вказання степені. Наприклад: «Ряд Тейлора в точці -1 f = sin(cos(x))\*x» | Отримуємо запис ряду, розкладення здійснено до степені 4. |
| Ввести у головному полі команду для розкладу в ряд Тейлора, без вказання точки. Наприклад: «Ряд Тейлора до степені 3 f = sin(2\*x)\*x» | Отримуємо запис розкладення функції в ряд Маклорена |
| Ввести у головному полі команду для розкладу в ряд Тейлора, без вказання точки і степені. Наприклад: «Ряд Тейлора f = sin(cos(x))\*x» | Отримуємо запис розкладення функції в ряд Маклорена до степені 4 |

Таблиця 4.18 – тестовий випадок Р.Т.2

|  |  |
| --- | --- |
| Номер | Р.Т. 2 |
| Назва | Ввести некоректні дані |
| Передумови | Відкрита головна сторінка веб-застосунку |
| Кроки | Очікуваний результат |
| Ввести набір символів, замість вказання степені розкладання. Наприклад: «Ряд Тейлора до степені abc& в точці -1 f = sin(cos(x))\*x» | Отримано повідомлення про неможливість інтерпретації вводу |
| Ввести набір символів, замість вказання точки розкладання. Наприклад: «Ряд Тейлора до степені 3 в точці @ f = sin(cos(x))\*x» | Отримано повідомлення про неможливість інтерпретації вводу |
| Ввести набір символів, замість вказання функції. Наприклад: «Ряд Тейлора до степені 3 в точці -1 f = abc» | Отримано повідомлення про неможливість інтерпретації вводу |

## 4.4 Висновки до розділу

Даний розділ був присвячений програмній реалізації навчальної системи.

Було, зокрема, описано алгоритми трьох основних функціональних можливостей розробленого веб-застосунку: символьного знаходження похідних, розкладання функцій у ряд Тейлора та, звісно, розв’язання задачі Коші методами Пікара і Рунге-Кутта. При чому, відмітимо, що перші дві з низ – це просто підзадачі, які були вирішені під час реалізації методу Пікара. Їх було вирішено додати у систему, зважаючи на їх потенційну корисність при розв’язанні задач Коші, адже система призначена, у першу чергу, для студентів.

Для методу Пікара, згідно теоретичних результатів з розділу 3, було реалізовано два окремих алгоритми: для функцій, заданих поліномами та таких, що представлені іншими елементарними функціями.

Значна увага була приділена розробці тестових випадків для розробленої системи, покликаних забезпечити якомога стабільнішу її роботу.

# 5 ЕКСПЕРИМЕНТАЛЬНІ РОЗРАХУНКИ

У цьому розділі продемонструємо роботу розробленої системи на конкретних прикладах та виконаємо тестові випадки, щоб переконатися в коректності роботи програми. Доступ до веб-застосунку у Мережі, можна отримати за наступним посиланням: <https://picardsolver.herokuapp.com/>

## 5.1 Демонстрація роботи програми

Щоб продемонструвати коректність роботи програми і наглядно показати його збіжність, візьмемо таку просту задачу Коші:

Це чи не найпростіше рівняння з відокремлюваними змінними. Аналітичний розв’язок цієї задачі Коші – це:

Введемо тоді це рівняння у систему і порівняємо результати. Їх наведено на рисунку 5.1.

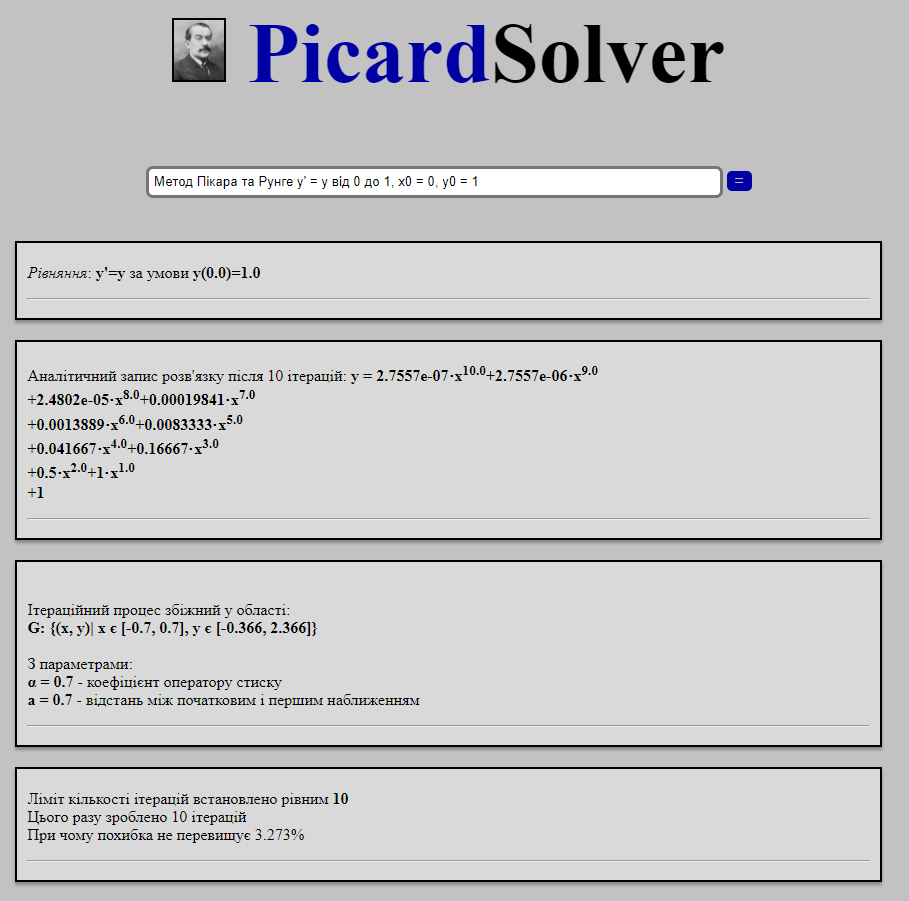


Рисунок 5.1 – Аналітична частина наближеного розв’язку задачі Коші

Щоб переконатися у правильності отриманого результату варто звернути увагу на поле аналітичного запису розв’язку вище, адже це ряд Тейлора експоненти. Дійсно:

Тобто якраз саме те, що ми й отримали у відповідному полі.

Це, до того ж, вкотре підтверджує лінійну збіжність методу, адже ряд Тейлора має нескінченну кількість членів, а отже, щоб отримати точний розв’язок за цим методом потрібно зробити нескінченну кількість ітерацій.

Хоча, зазвичай, непогану точність вдається досягти за, відносно, невелику кількість ітерацій. Наприклад, після десяти ітерацій, для даної задачі Коші, наближена, за методом Пікара, інтегральна крива, фактично, співпадає з наближенням за методом Рунге-Кутта, як видно з графіків на рисунку 5.2.

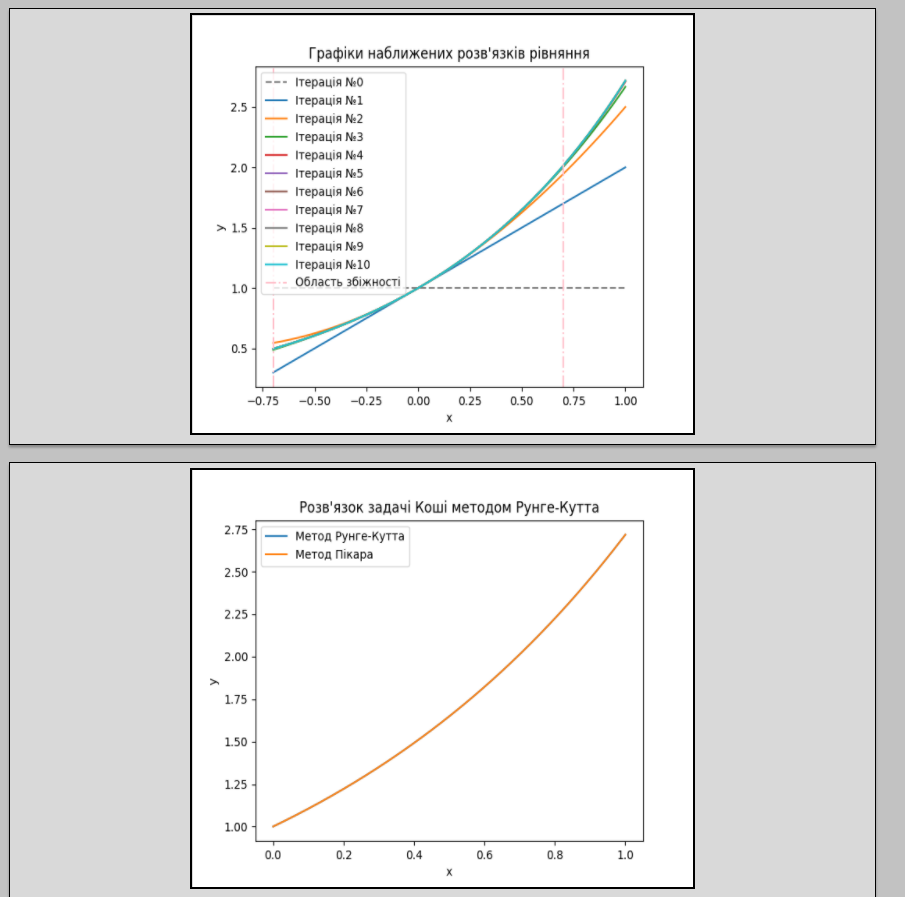


Рисунок 5.2 – Наближені інтегральні криві для даної задачі Коші

## 5.2 Перевірка тестових випадків

Перейдемо до перевірки тестових випадків. У якості демонстрації тестових випадків Д.Р. 1 – Д.Р. 5, можна взяти результати, отримані у попередньому розділі. Зображення усіх цих п’яти випадків – є недоцільним, зважаючи на те, що розв’язок надається досить об’ємний, але, фактично, однаковий. За виключенням цього, будемо виконувати решту тестових випадків у тому порядку, у якому вони зазначені у відповідному розділі. Екранні форми з тестовими випадками наведені на рисунках 5.3 – 5.32.

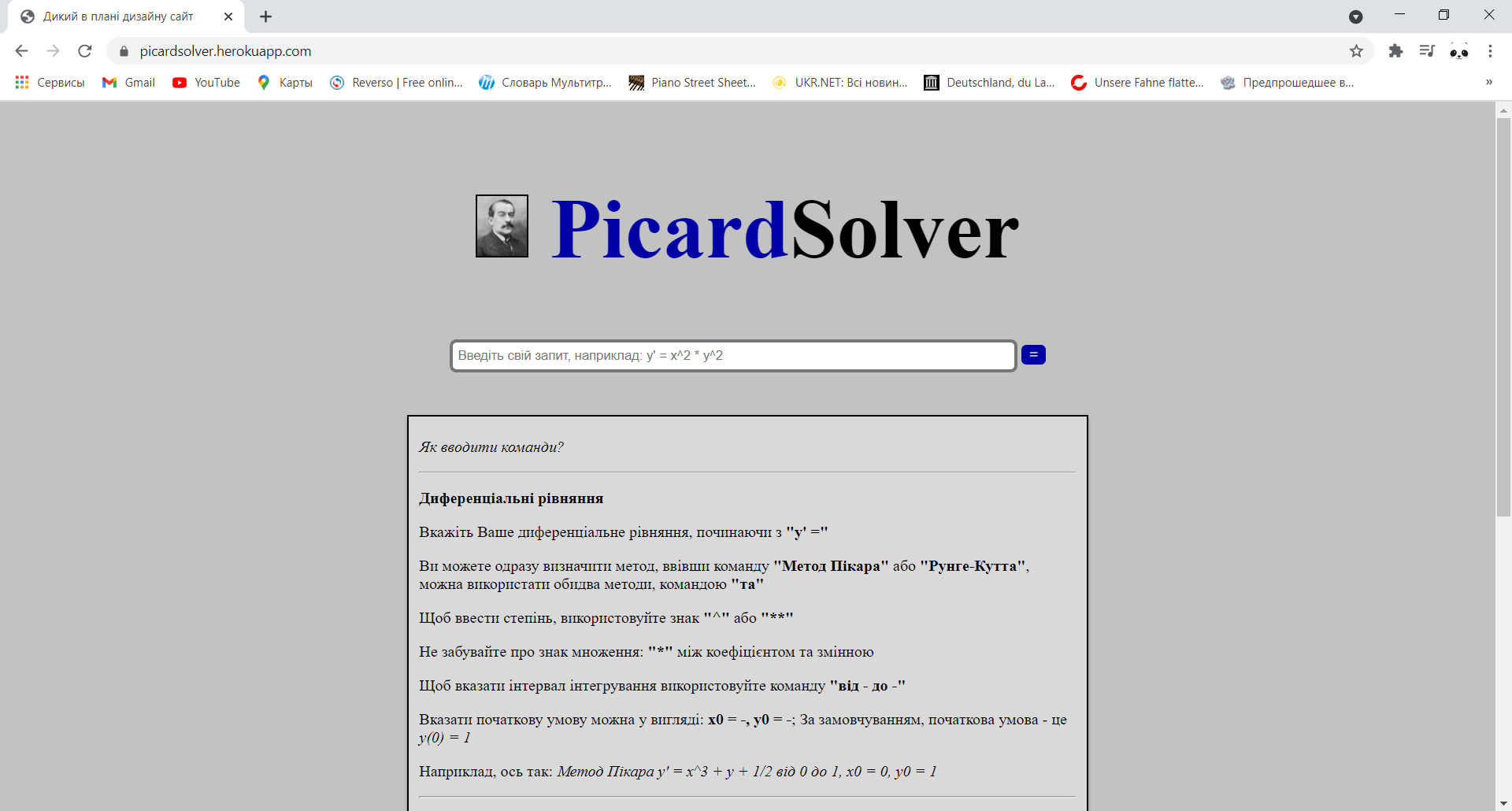


Рисунок 5.3 - Головна сторінка застосунку. Запити вводяться у поле у центрі

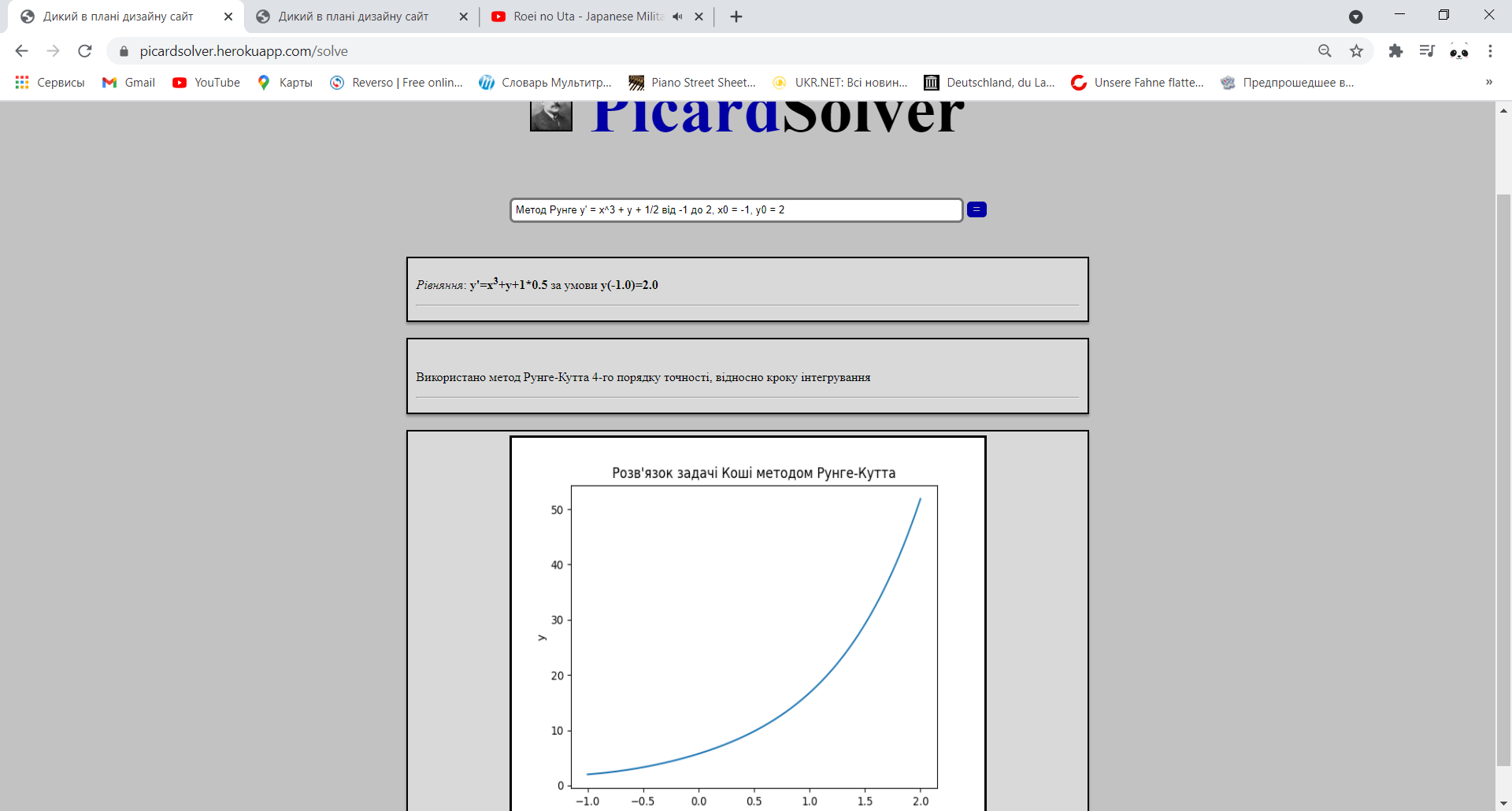


Рисунок 5.4 - Тестовий випадок Д.Р.6, відображення розв’язку методом Рунге-Кутта

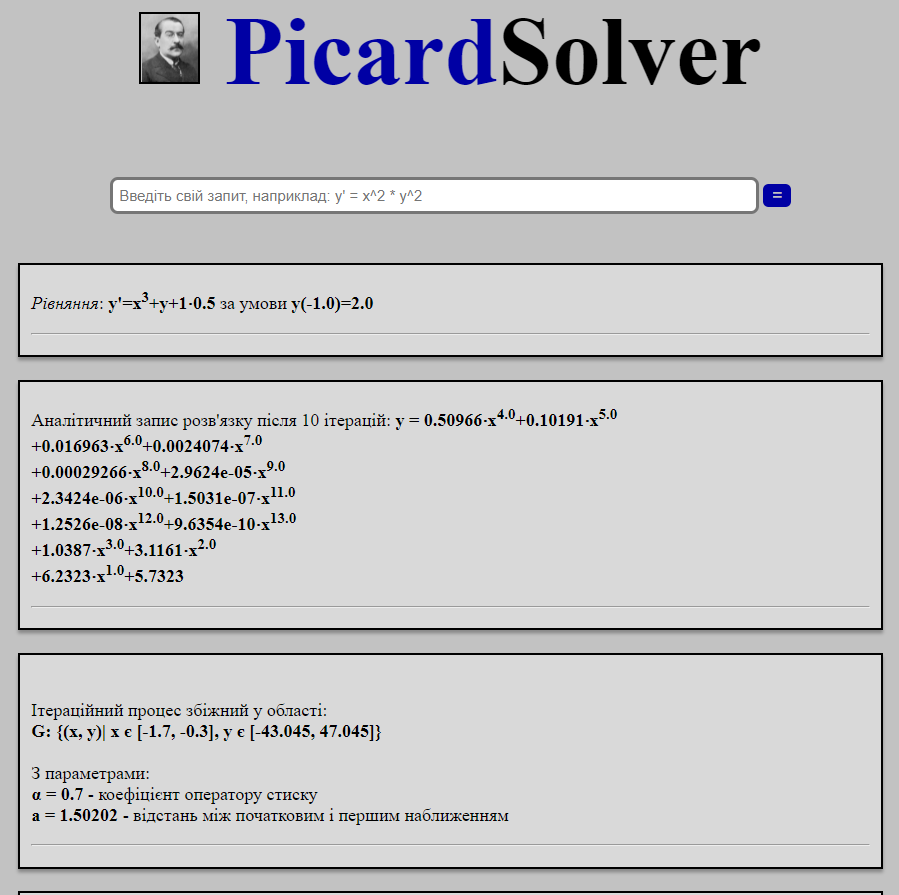


Рисунок 5.5 - Тестовий випадок Д.Р.7, відображення розв’язку, частина 1



Рисунок 5.6 - Тестовий випадок Д.Р.7, відображення розв’язку, частина 2

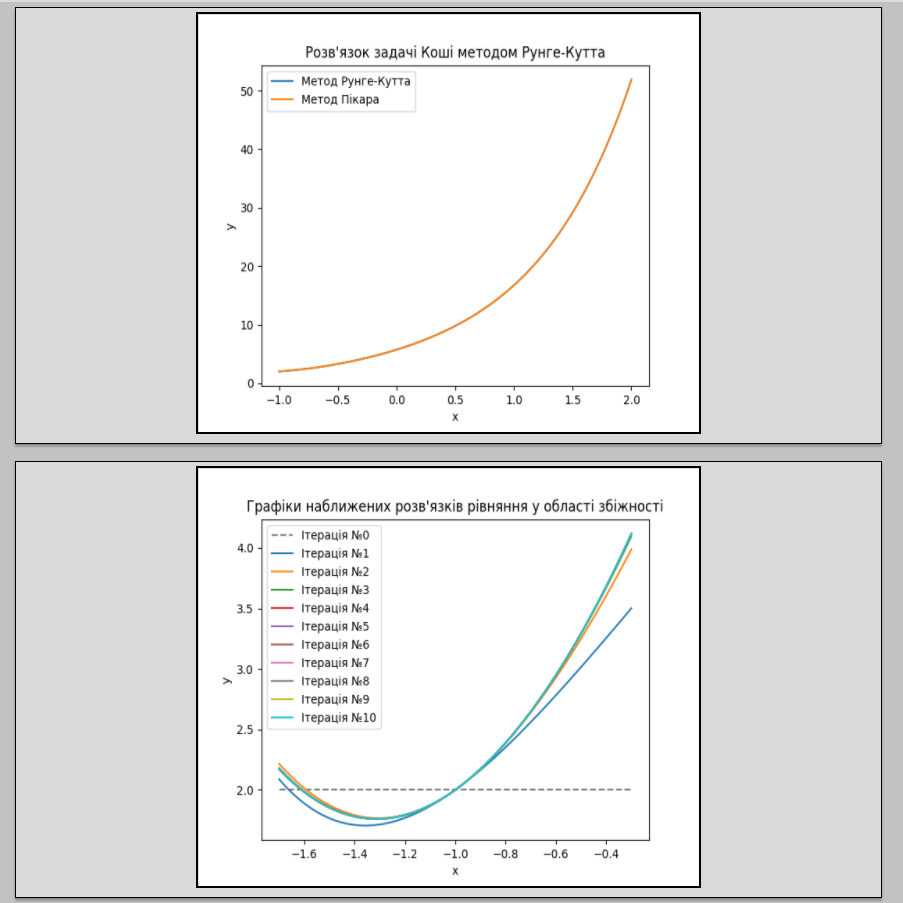


Рисунок 5.7 - Тестовий випадок Д.Р.7, відображення розв’язку, частина 3

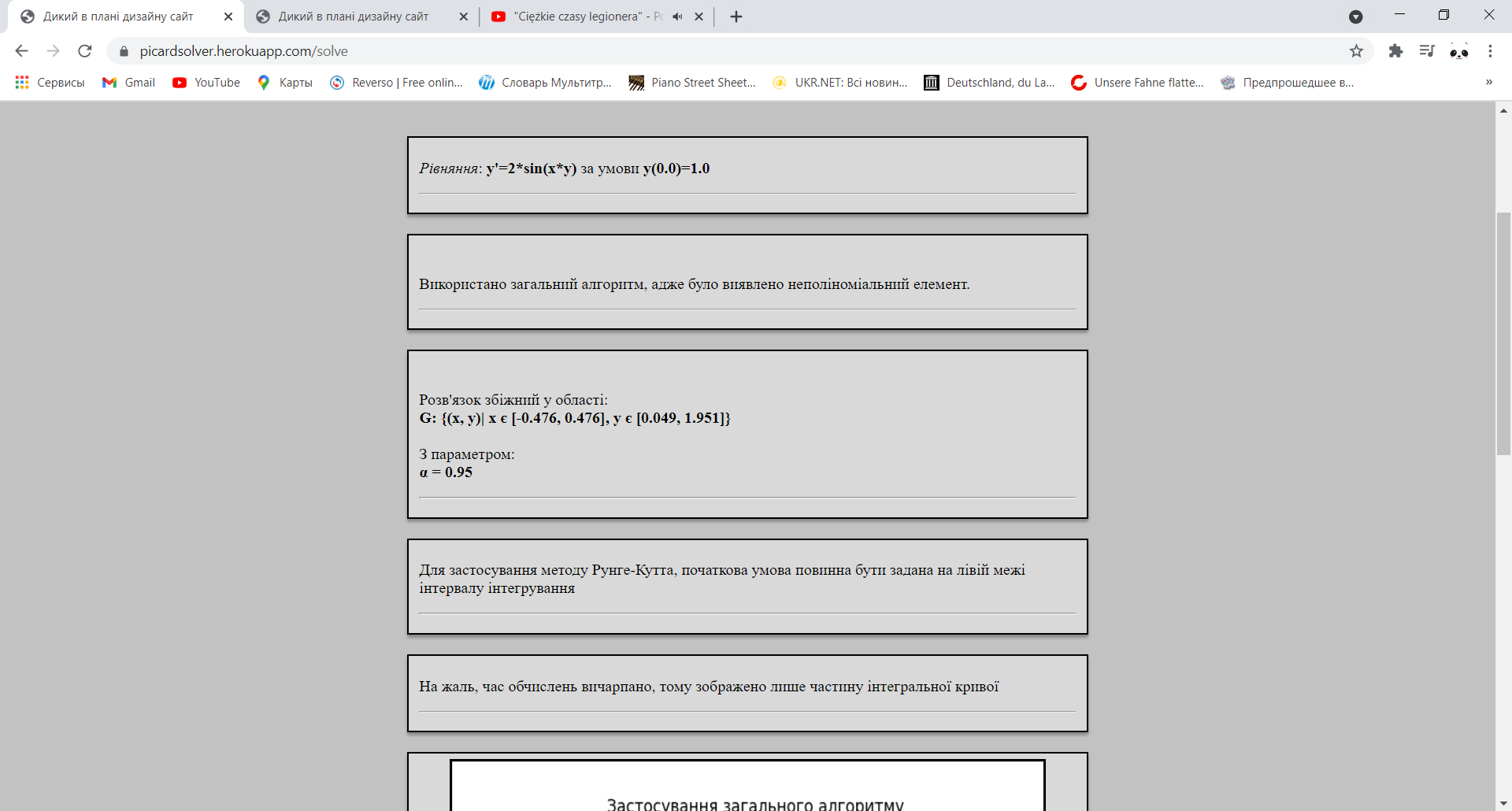


Рисунок 5.8 - Тестовий випадок Д.Р.8, відображення розв’язку, частина 1

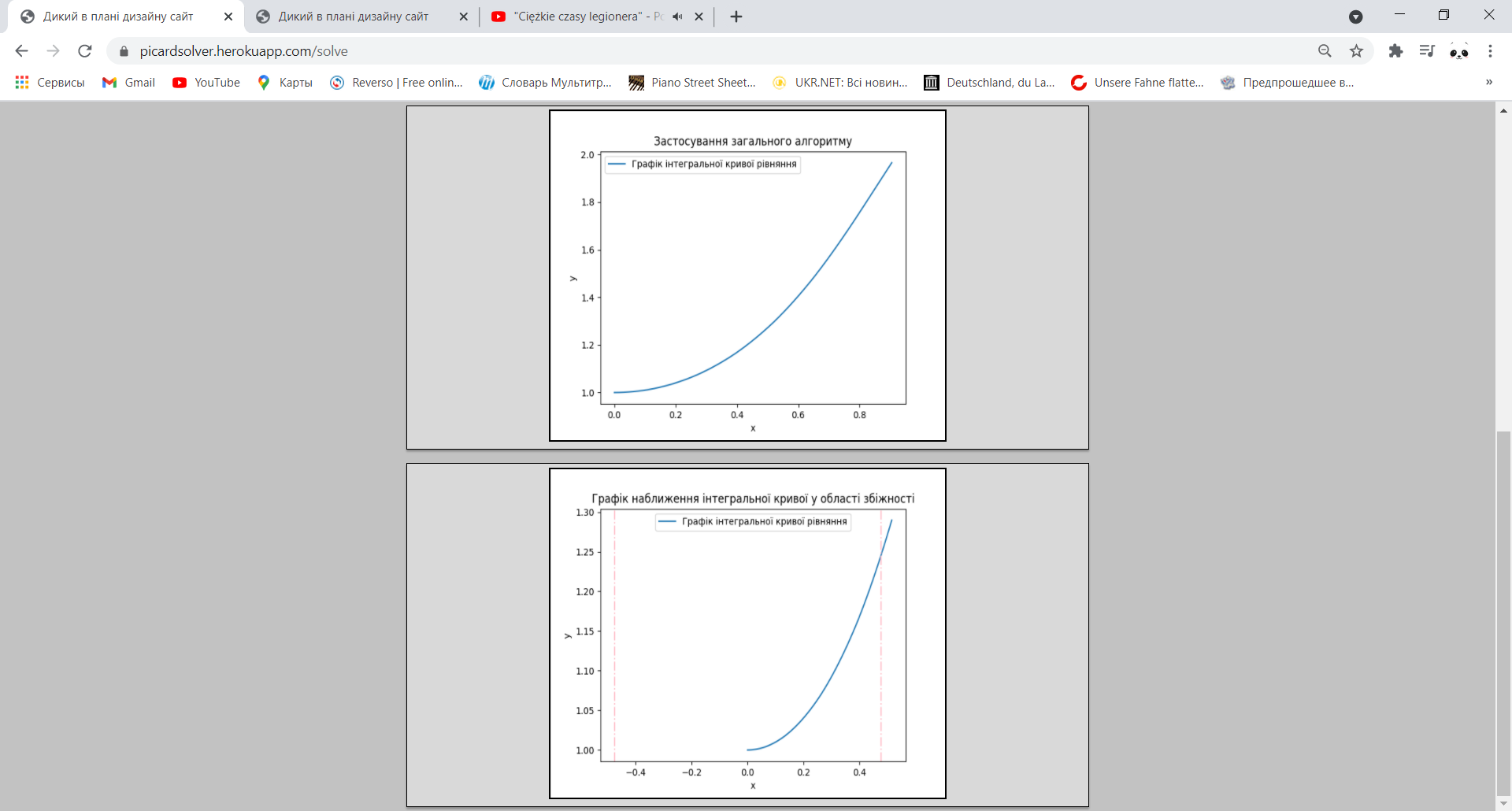


Рисунок 5.9 - Тестовий випадок Д.Р.8, відображення розв’язку, частина 2

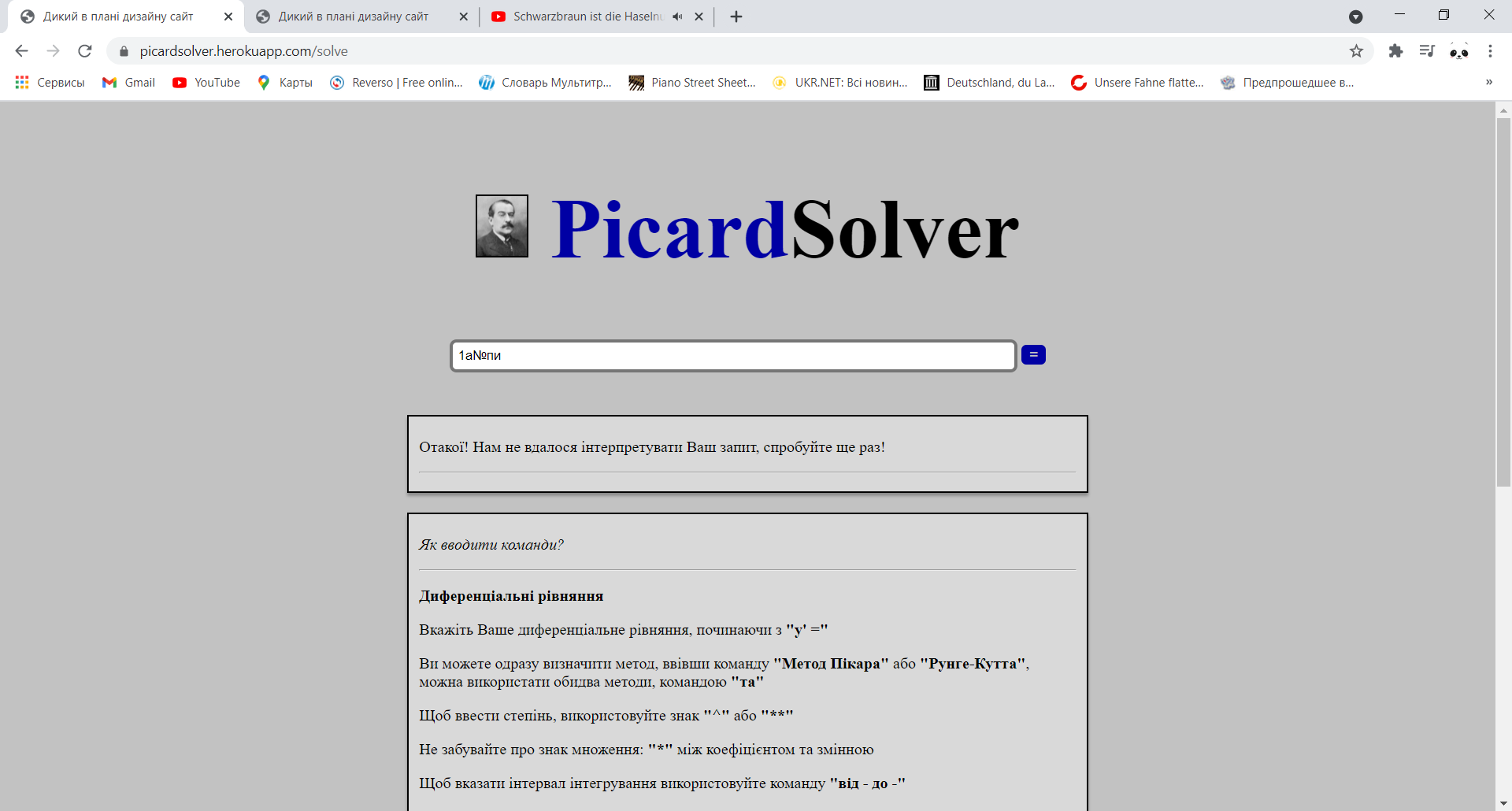


Рисунок 5.10 - Тестовий випадок Д.Р.9, некоректні дані у полі

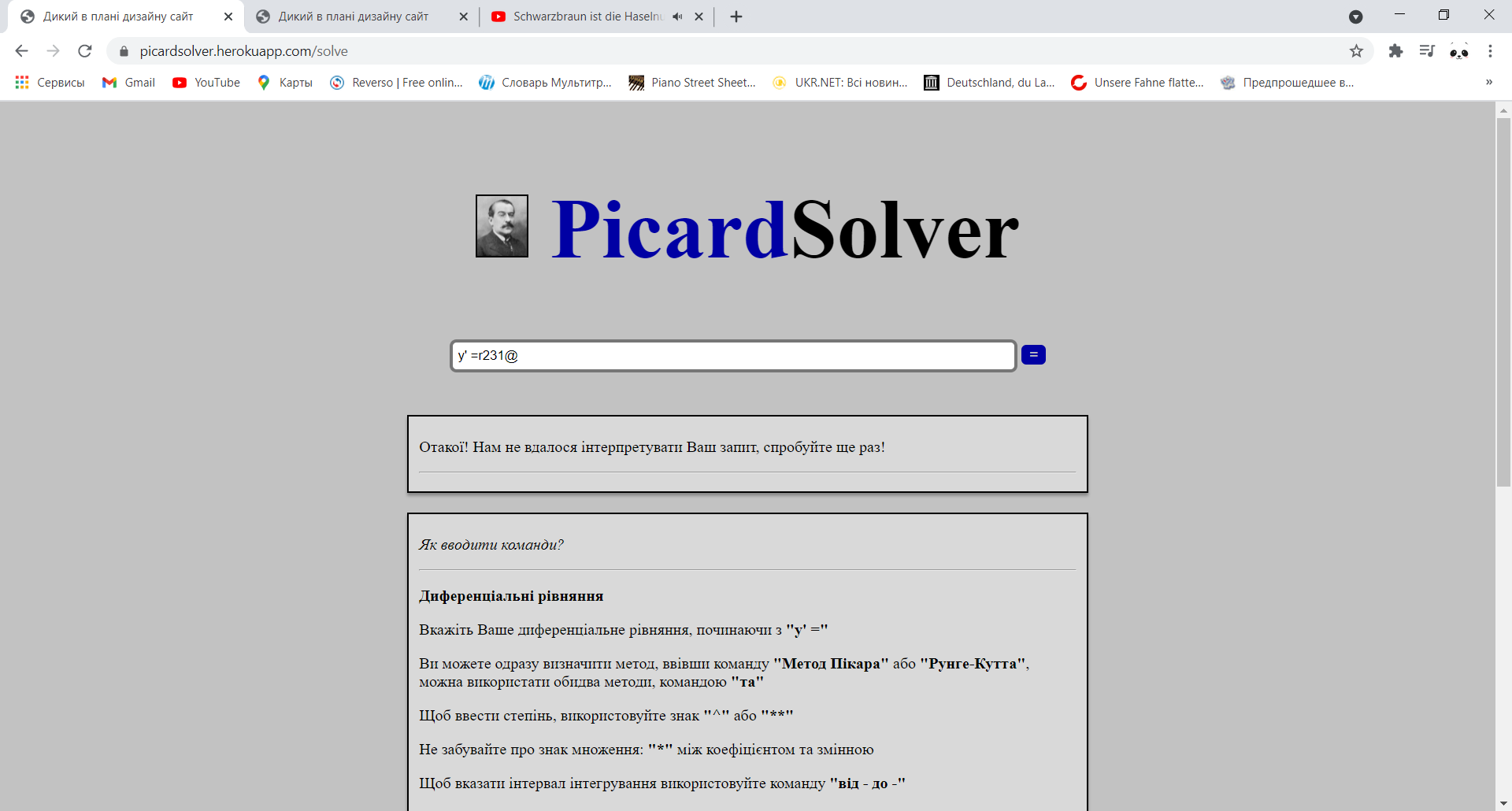


Рисунок 5.11 - Тестовий випадок Д.Р.9, некоректні дані замість диф. рівняння

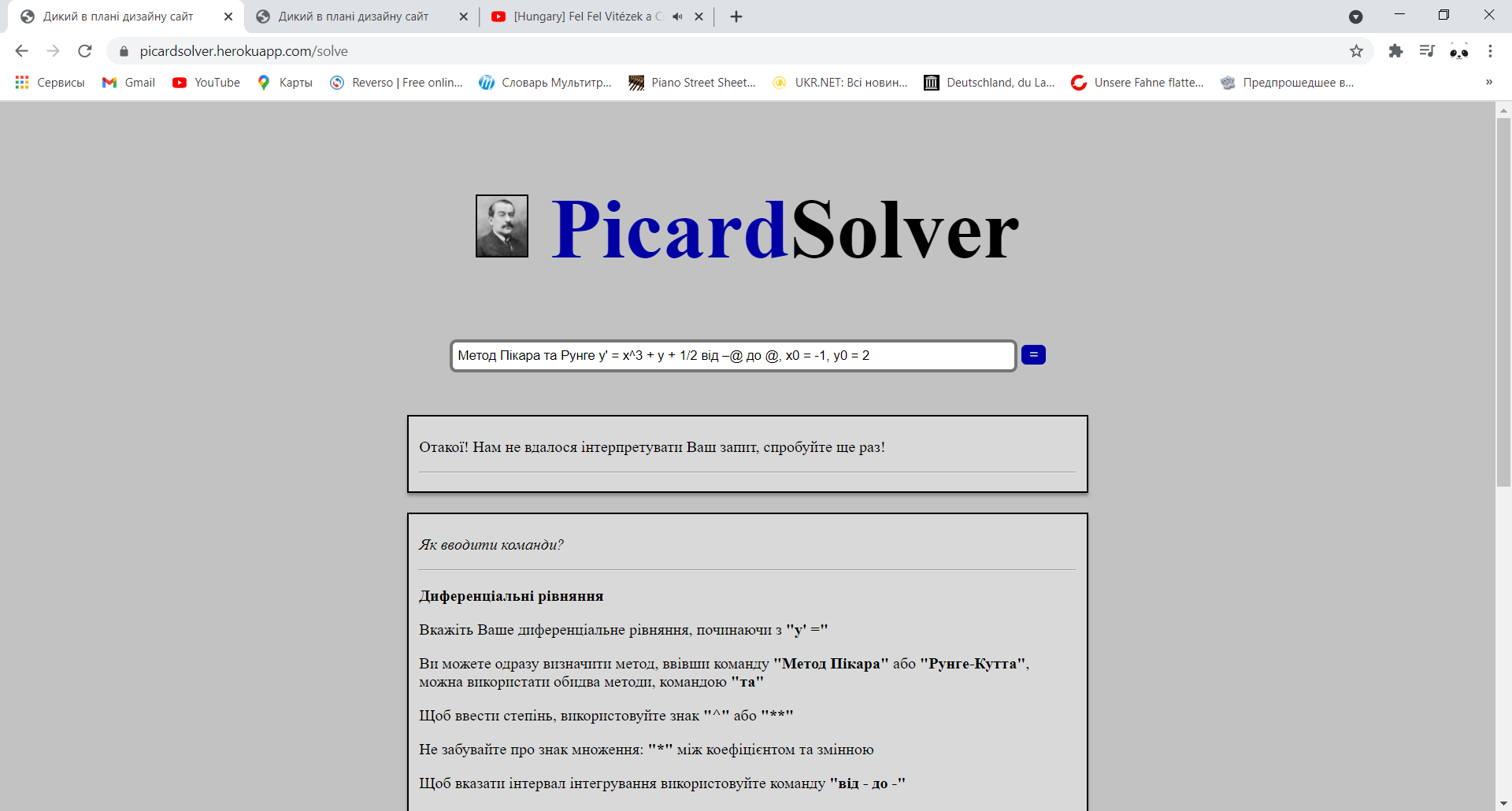


Рисунок 5.12 - Тестовий випадок Д.Р.9, некоректні дані замість інтервалу інтегрування

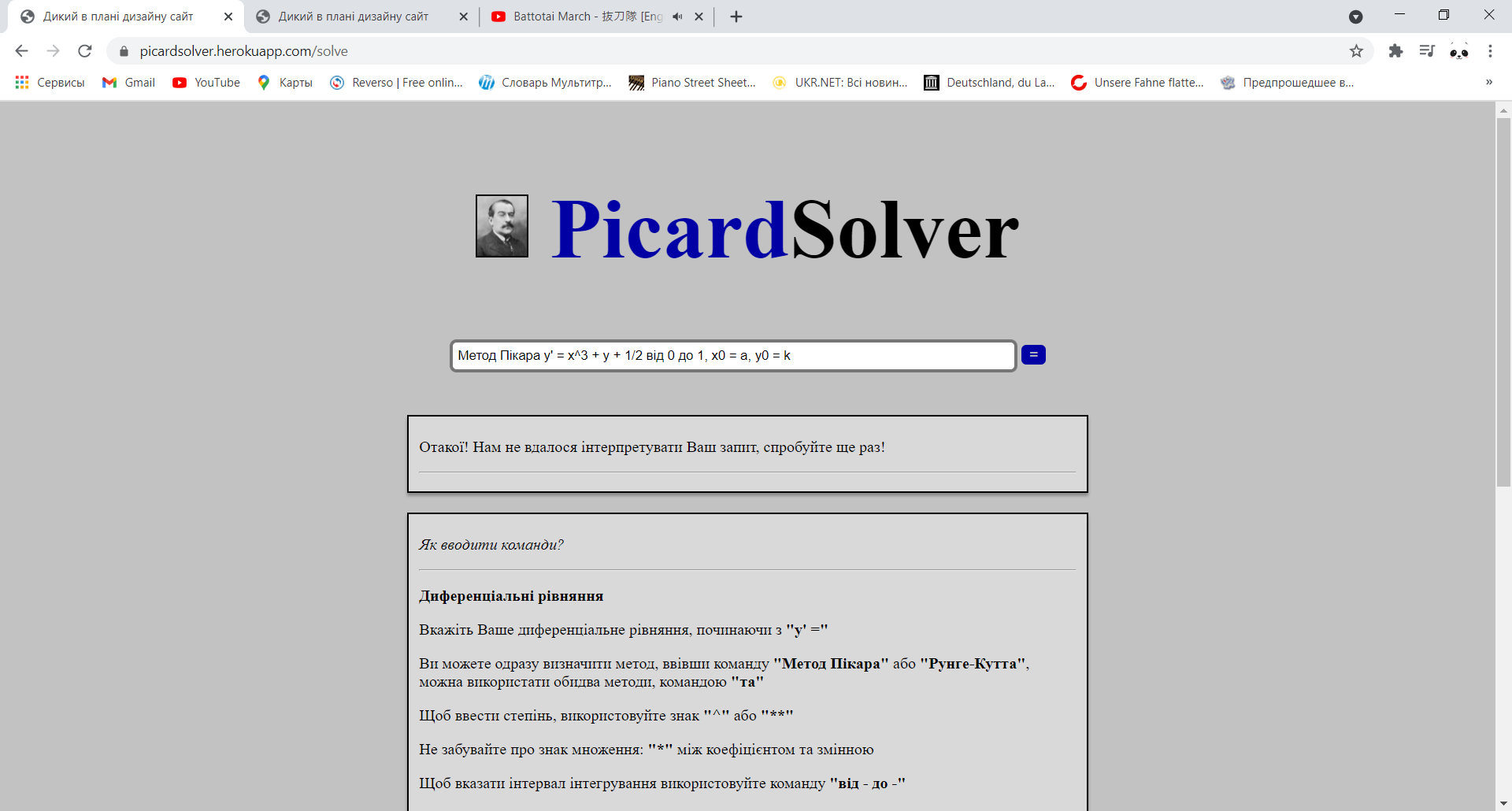


Рисунок 5.13 - Тестовий випадок Д.Р.9, некоректні дані замість початкової умови



Рисунок 5.14 - Тестовий випадок Д.Р.10, не виконана умова Ліпшиця

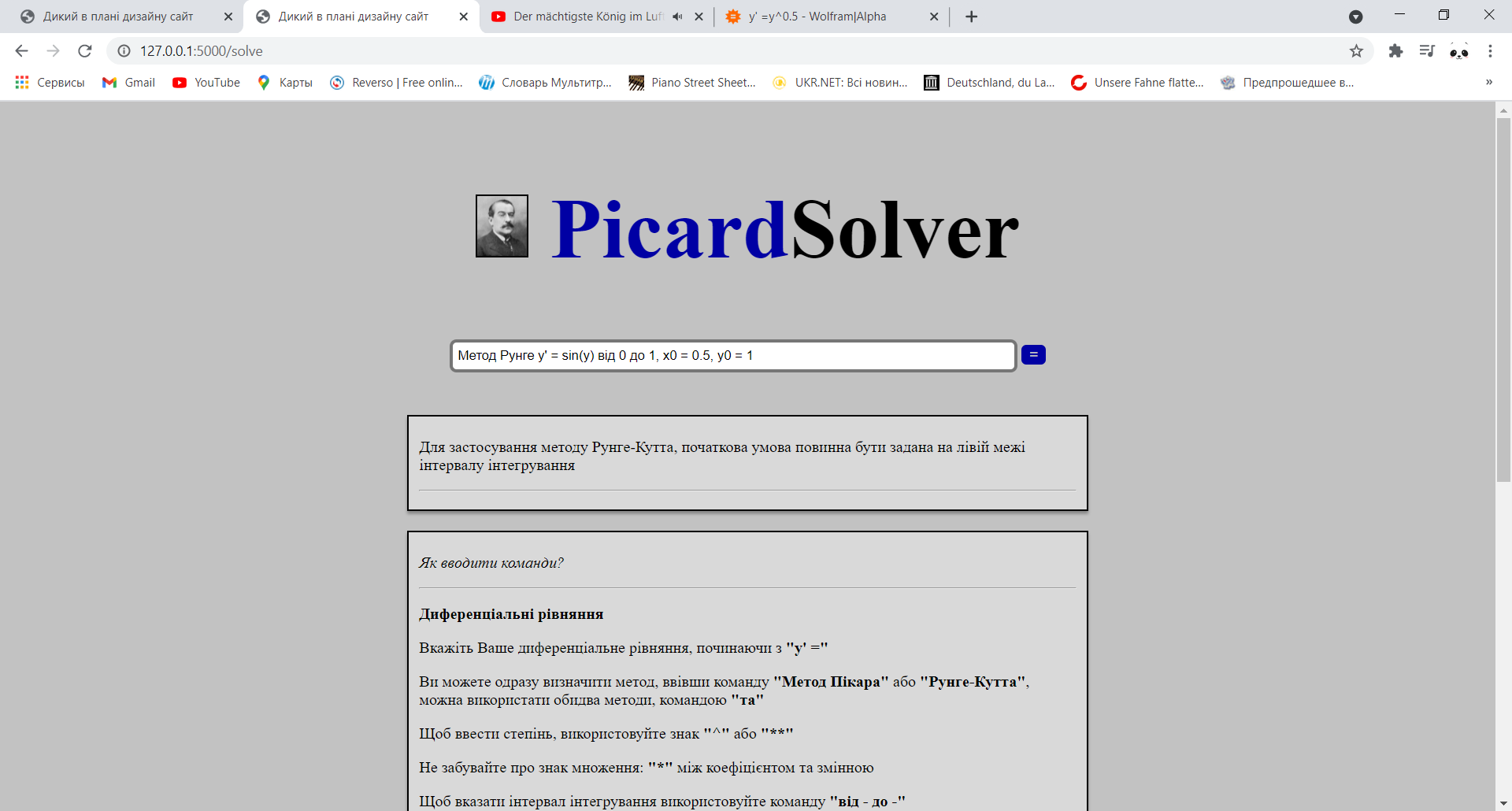


Рисунок 5.15 - Тестовий випадок Д.Р.11, точка x0 лежить всередині інтервалу інтегрування при застосуванні методу Рунге-Кутта

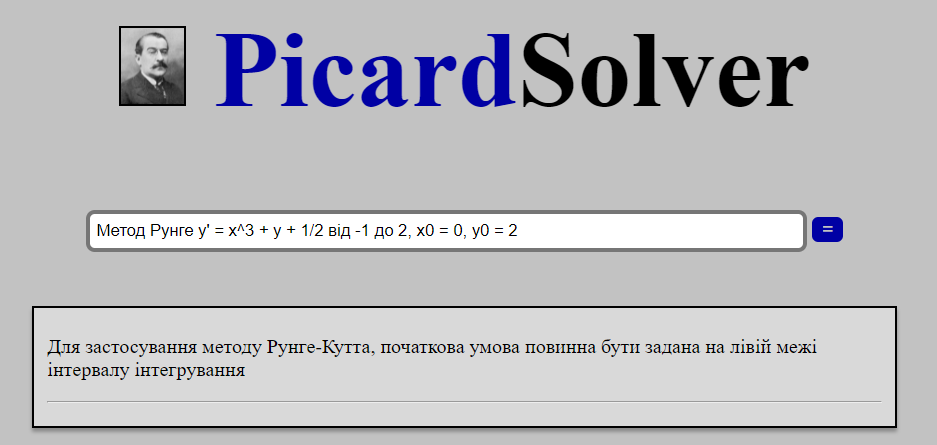


Рисунок 5.16 - Тестовий випадок Д.Р.11, точка x0 лежить всередині інтервалу інтегрування при застосуванні методу Рунге-Кутта та Пікара

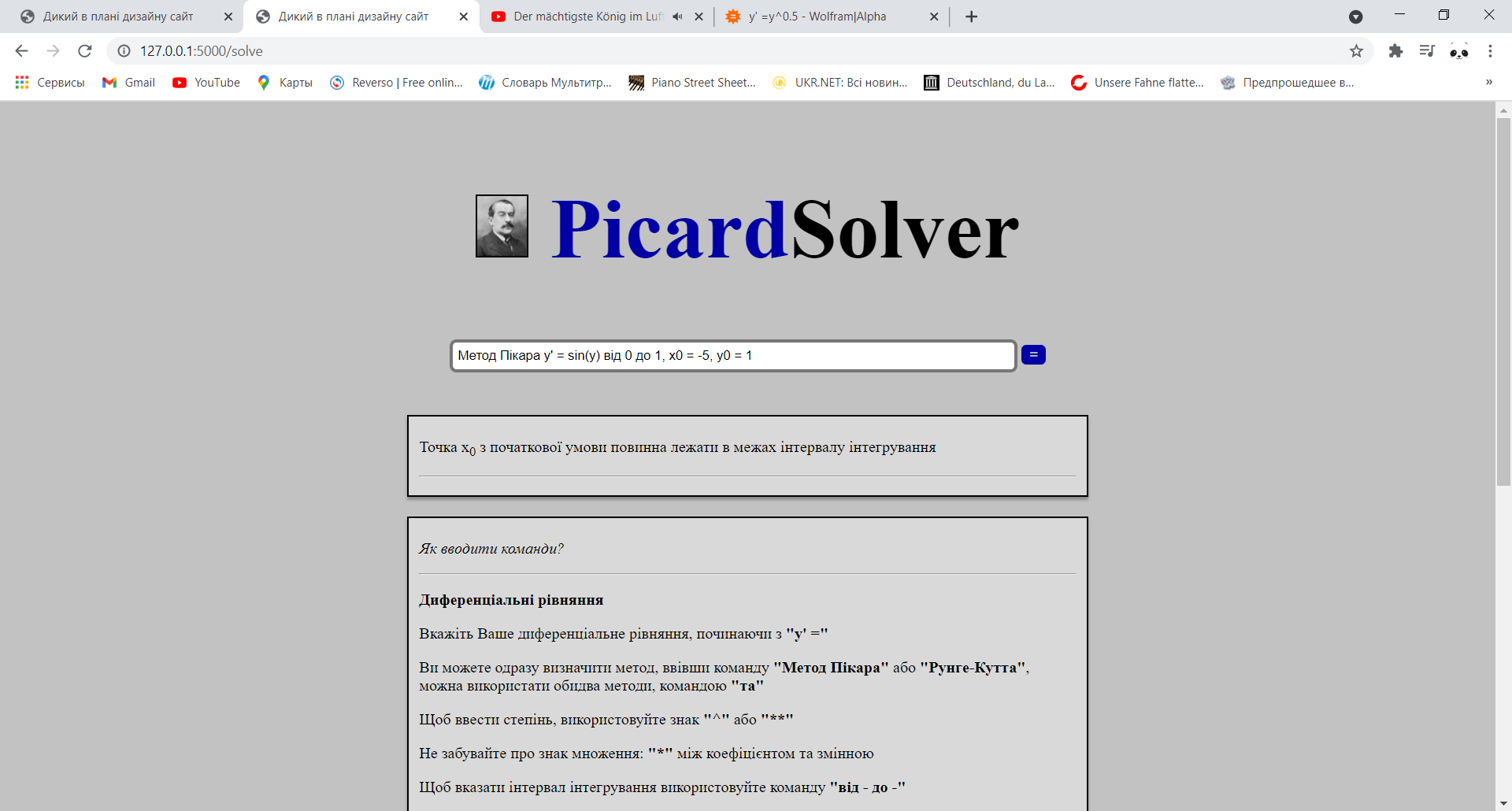


Рисунок 5.17 - Тестовий випадок Д.Р.12, точка x0 лежить за межами інтервалу інтегрування

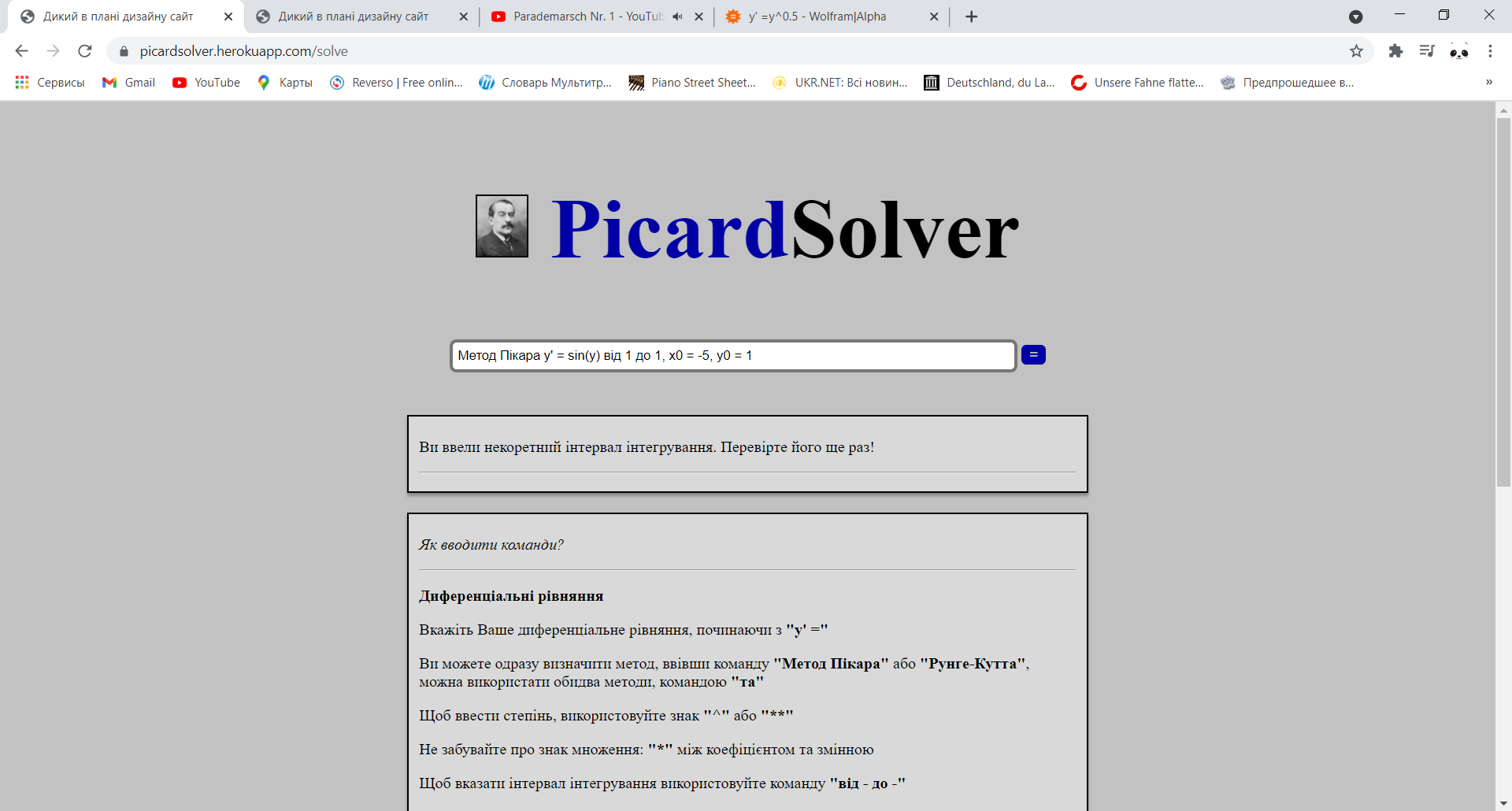


Рисунок 5.18 - Тестовий випадок Д.Р.13, довжина інтервалу інтегрування рівна нулю

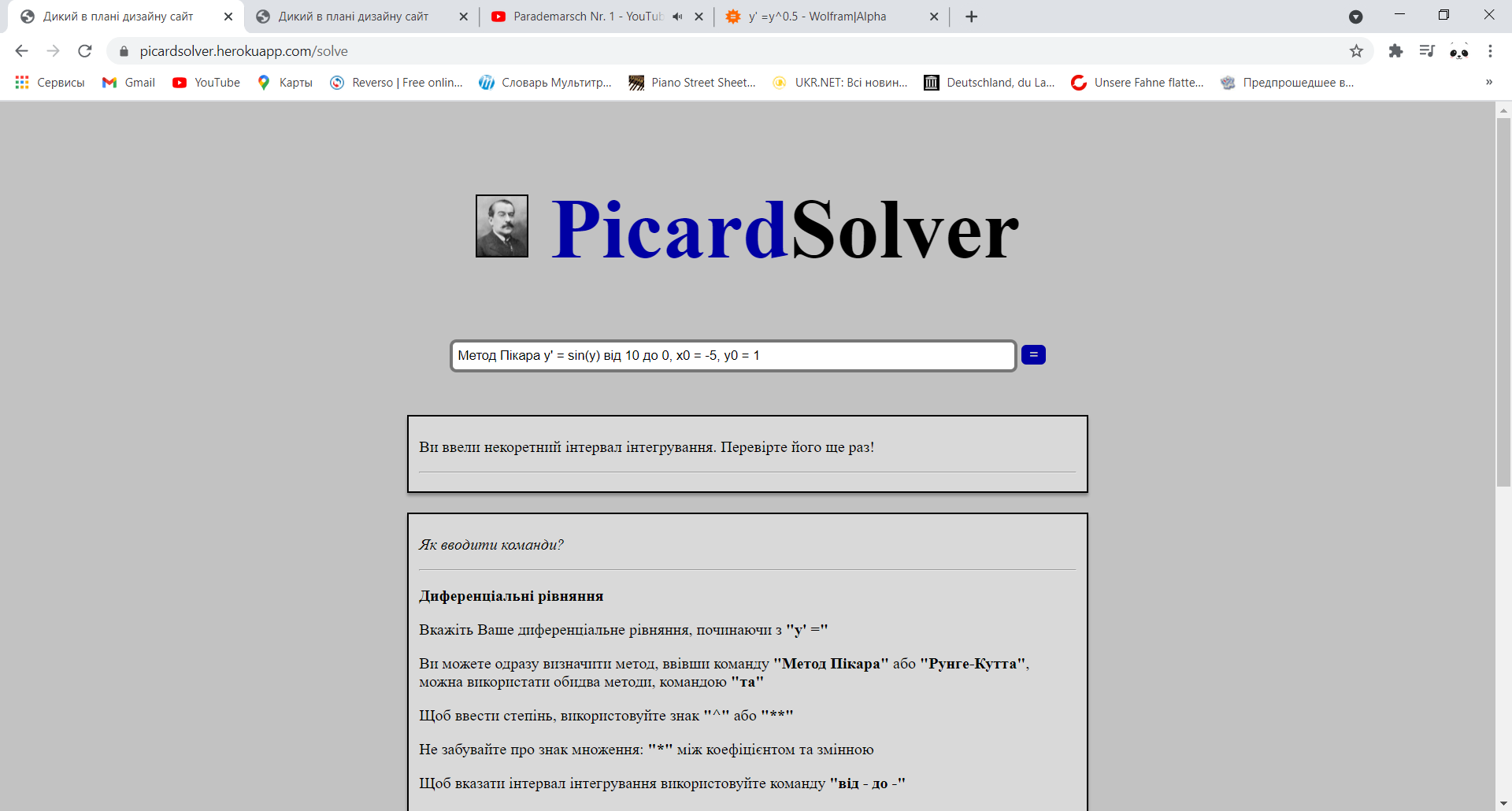


Рисунок 5.19 - Тестовий випадок Д.Р.13, інтервал задано некоректно

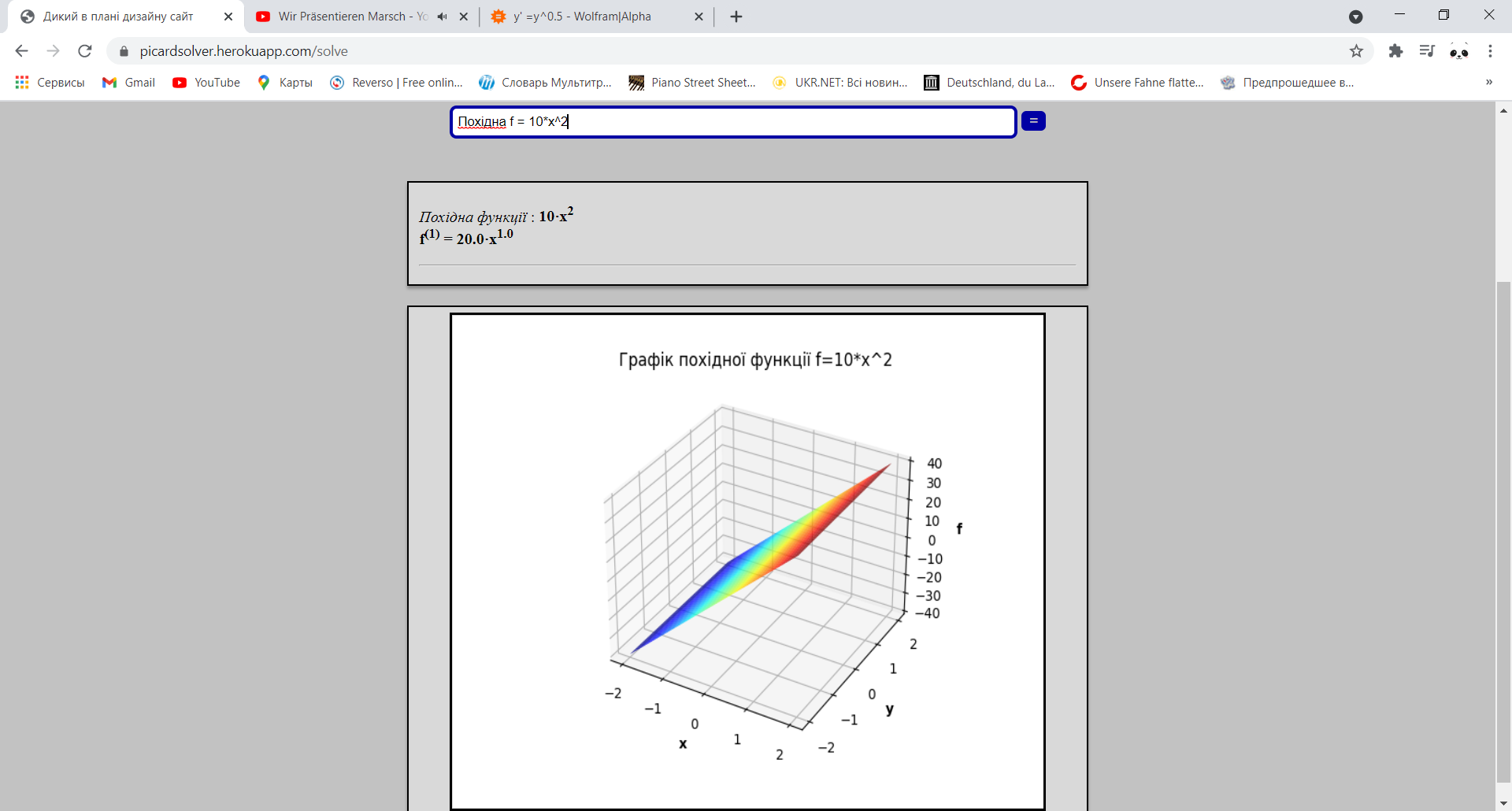


Рисунок 5.20 - Тестовий випадок П.1, частина 1

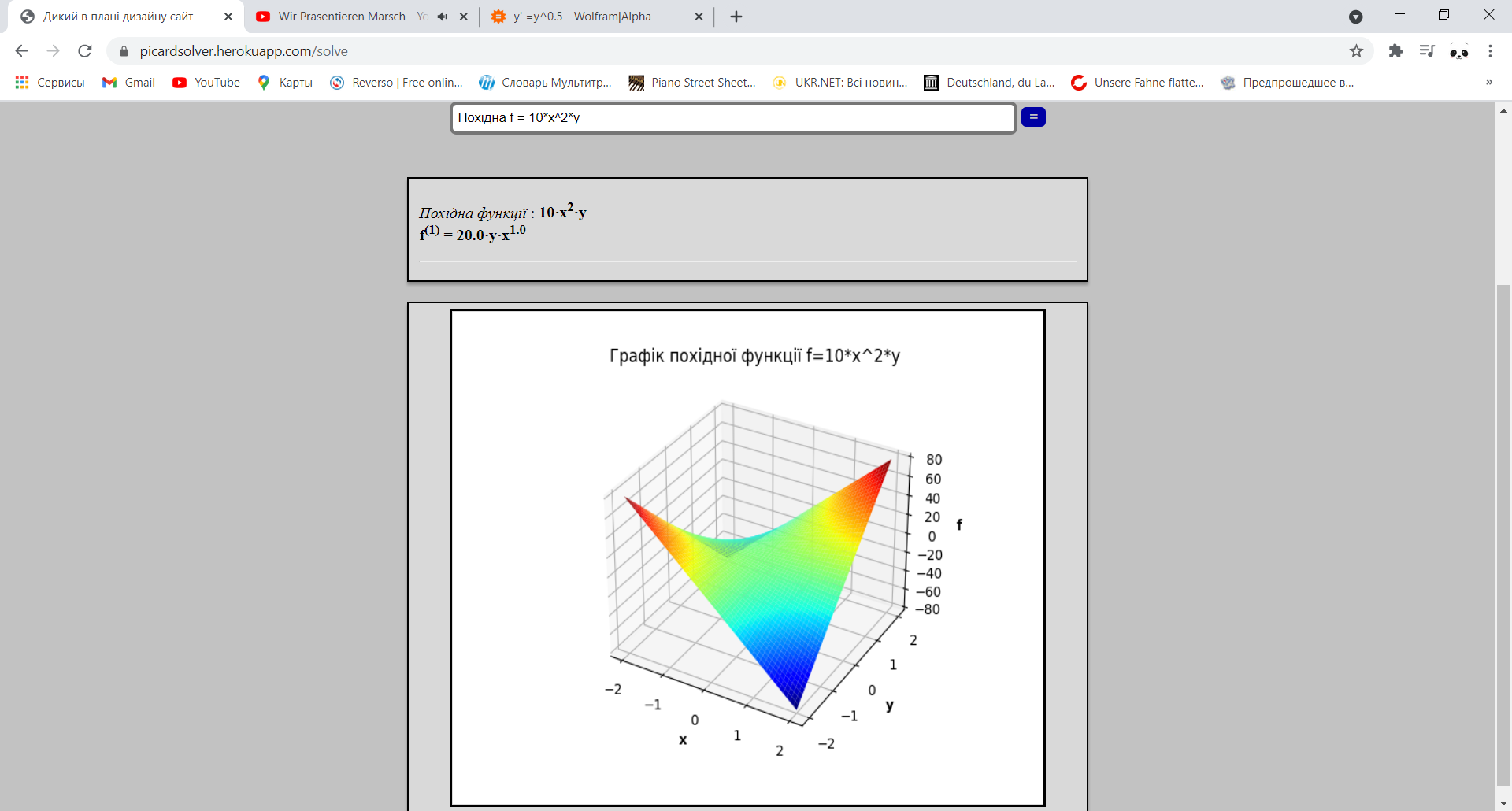


Рисунок 5.21 - Тестовий випадок П.1, частина 2

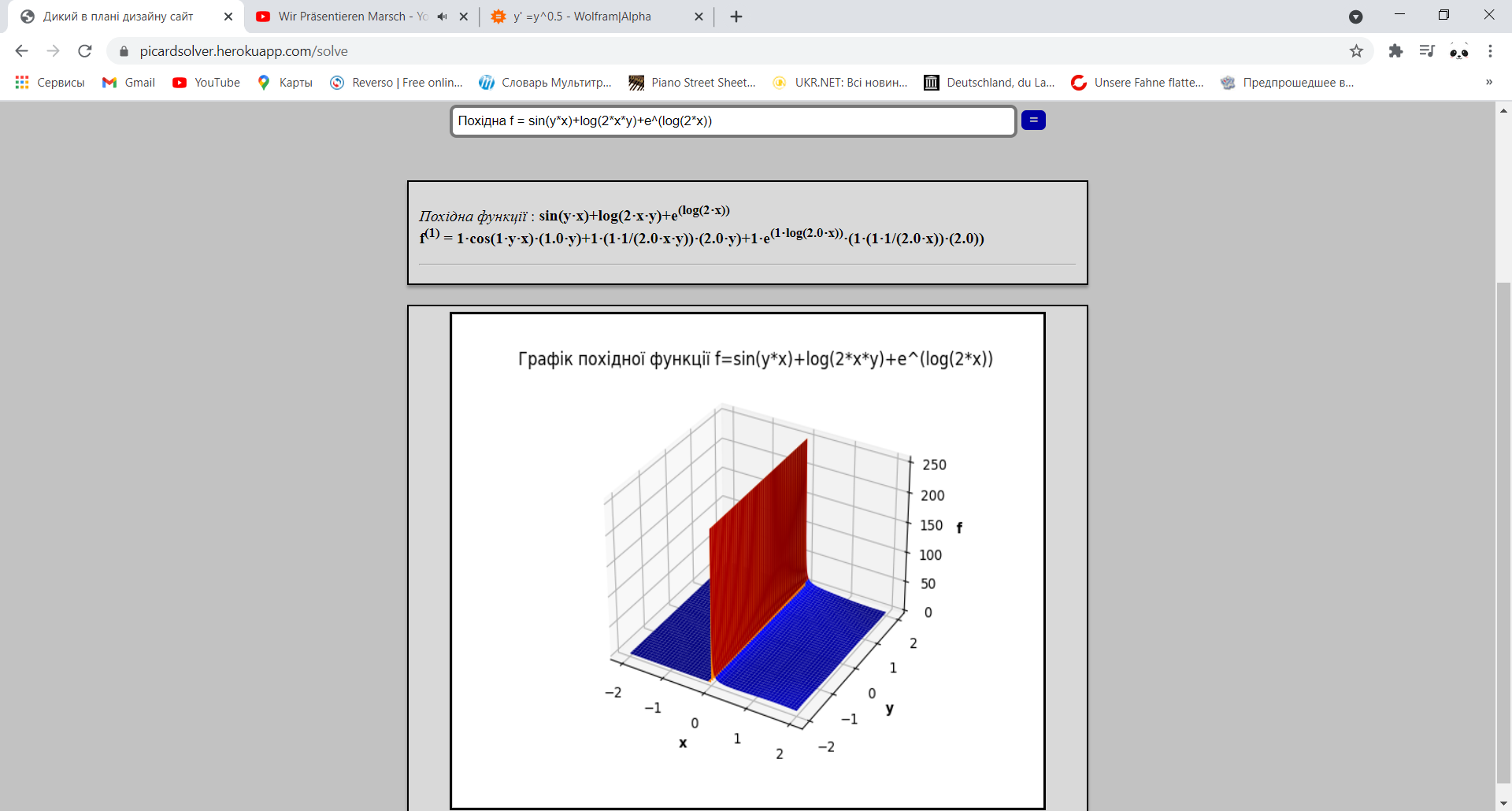


Рисунок 5.22 - Тестовий випадок П.1, частина 3

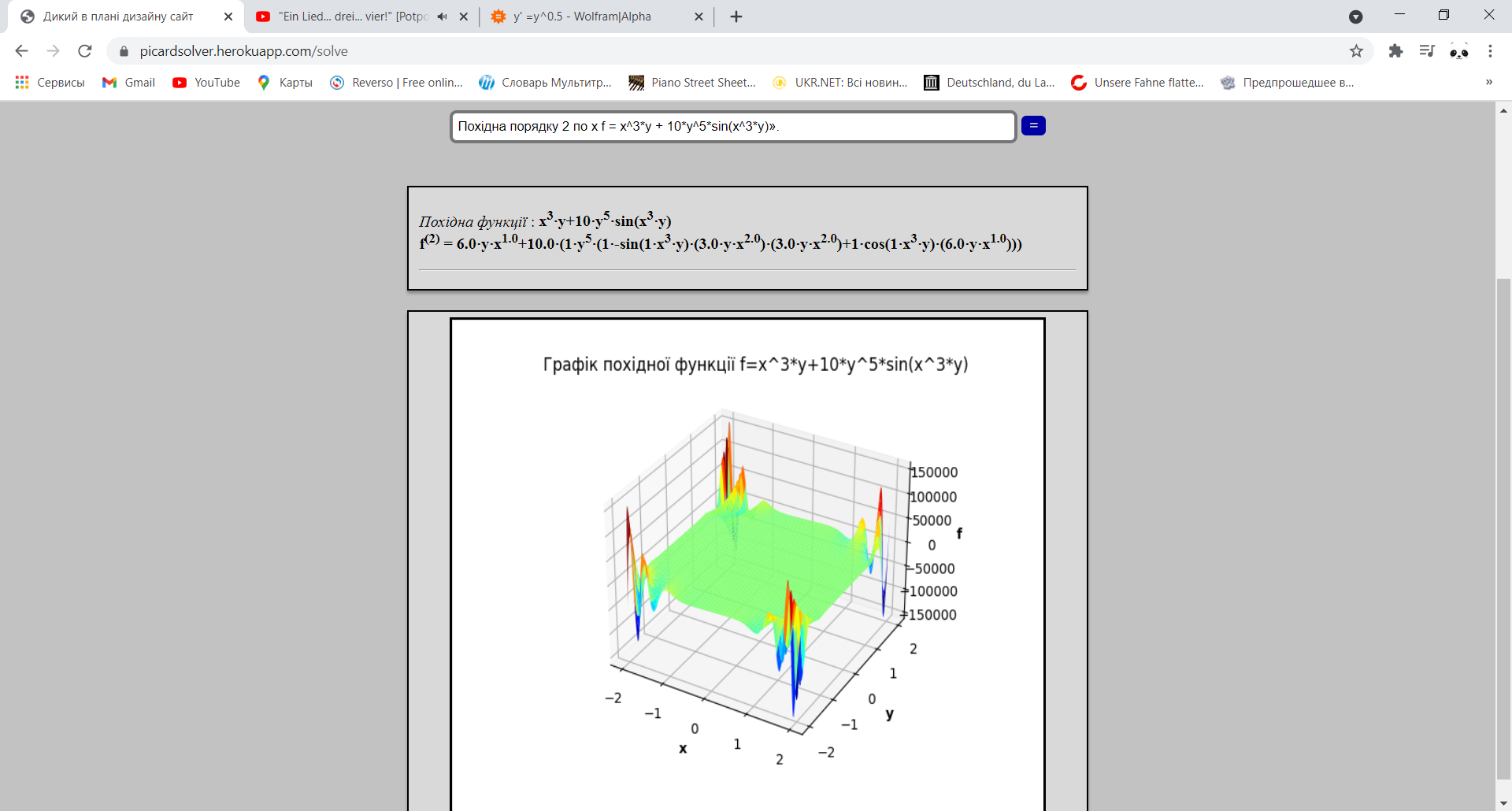


Рисунок 5.23 - Тестовий випадок П.1, частина 4

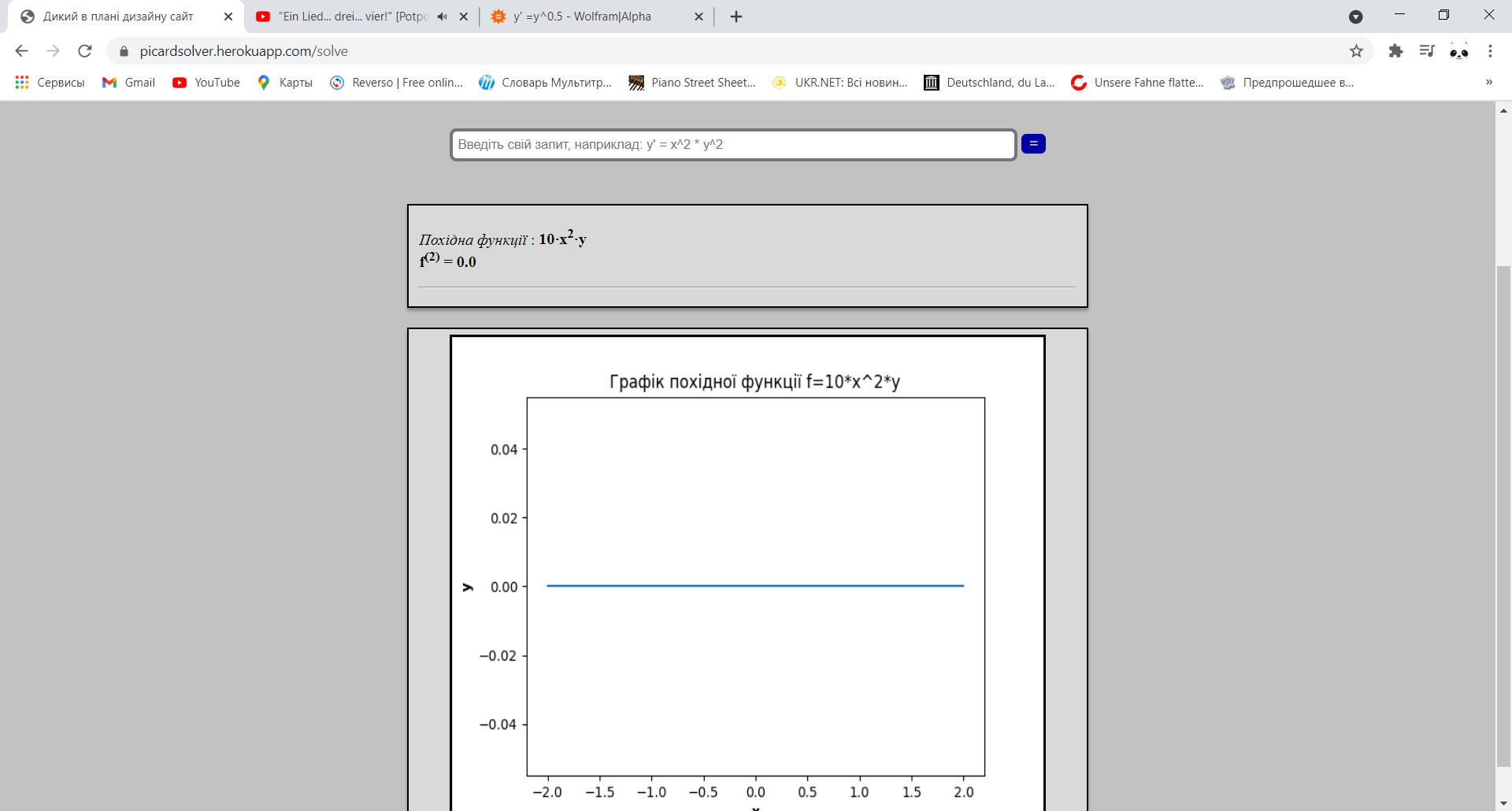


Рисунок 5.24 - Тестовий випадок П.2, похідна по у порядку 2

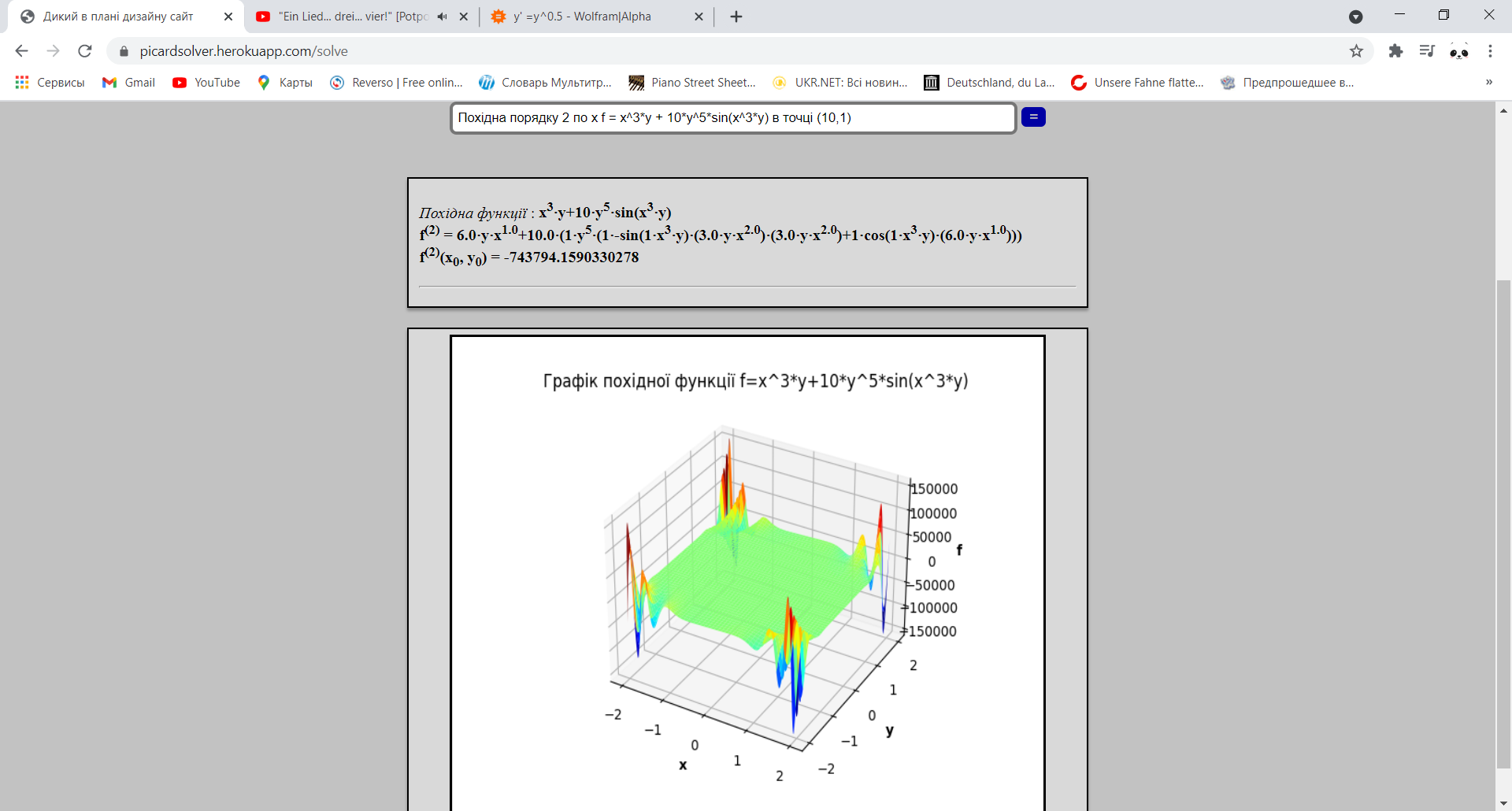


Рисунок 5.25 - Тестовий випадок П.3, похідна функції у точці

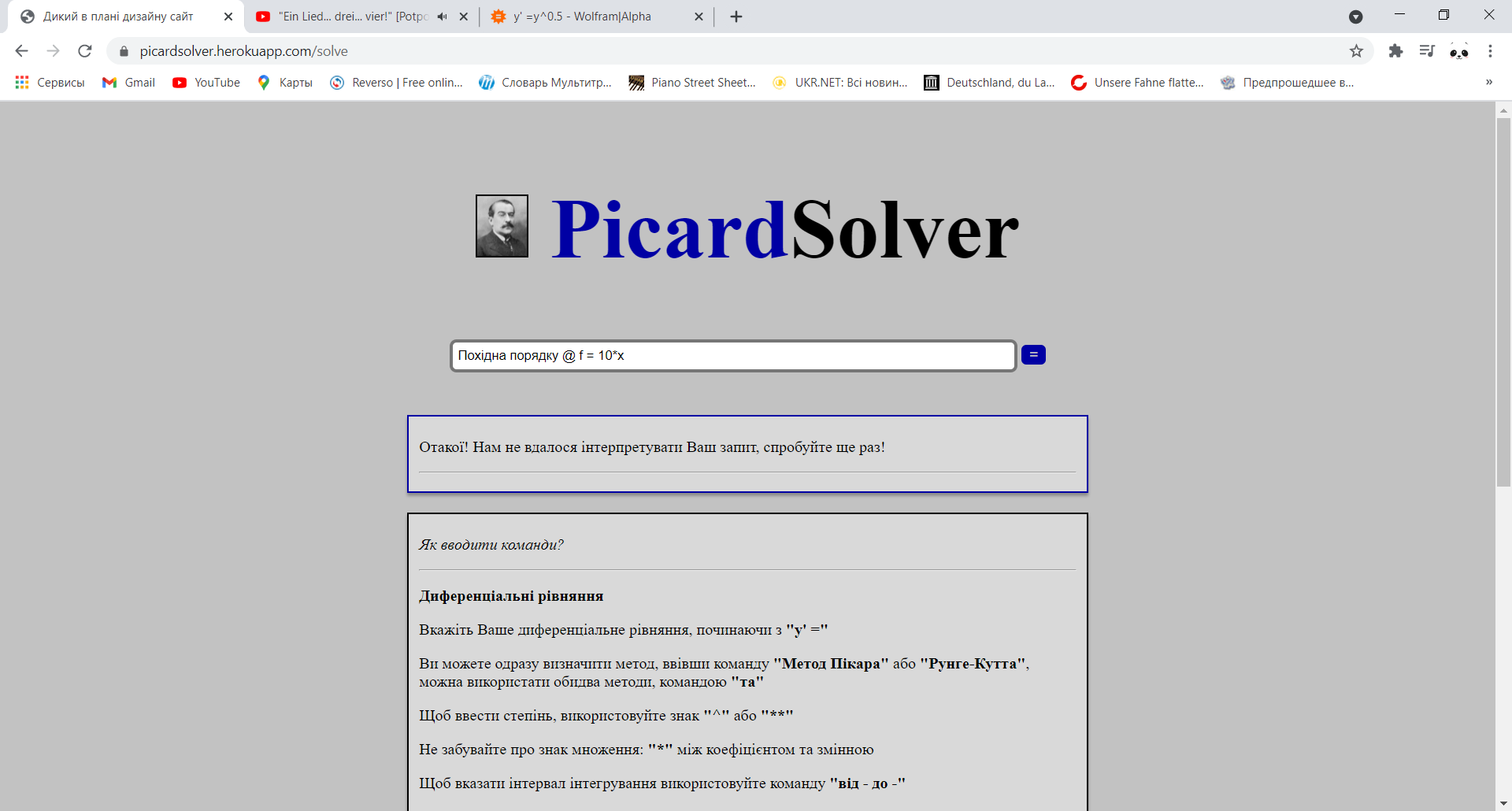


Рисунок 5.26 - Тестовий випадок П.4, некоректні дані замість порядку похідної

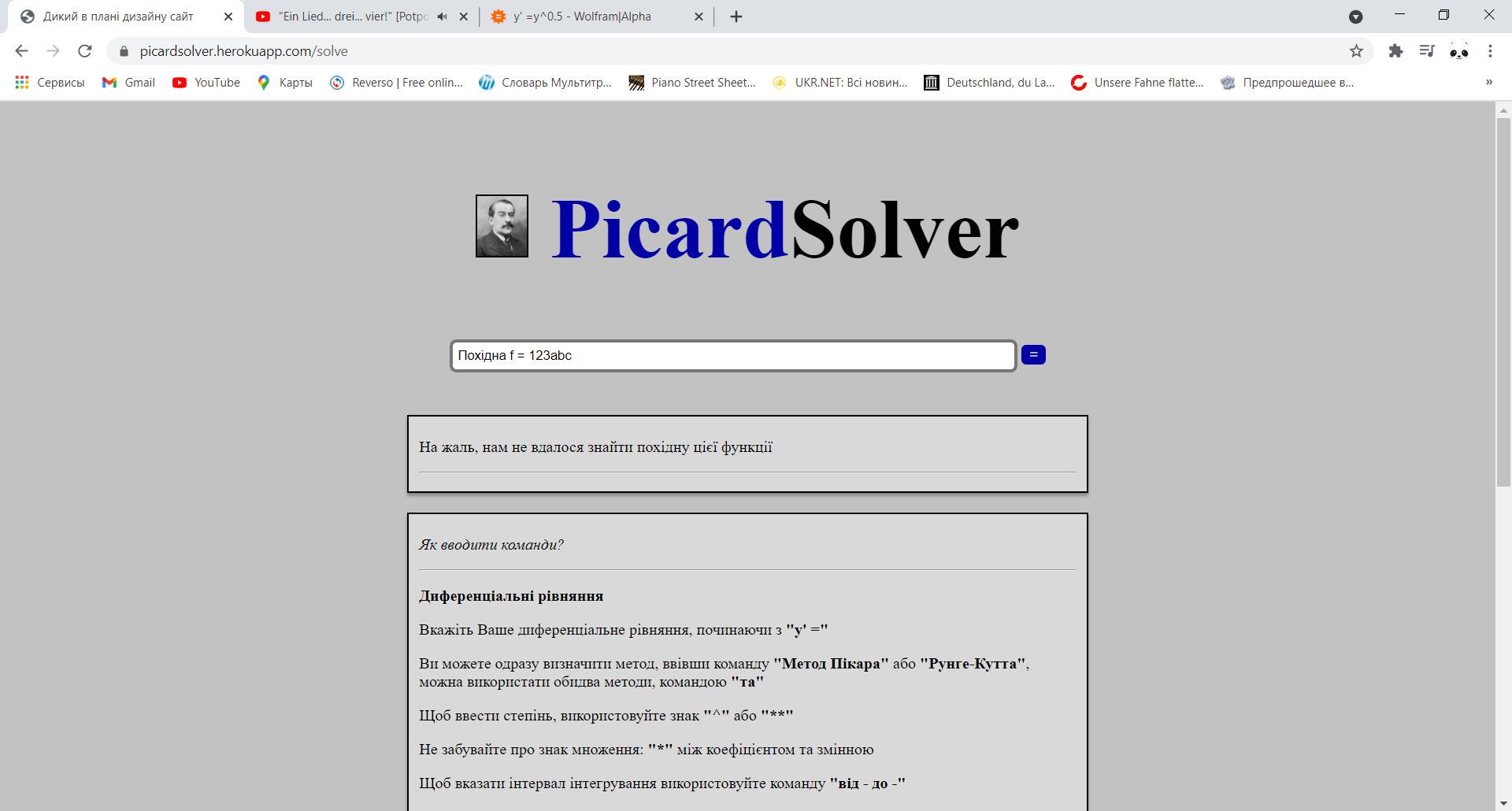


Рисунок 5.27 - Тестовий випадок П.4, некоректні дані замість функції

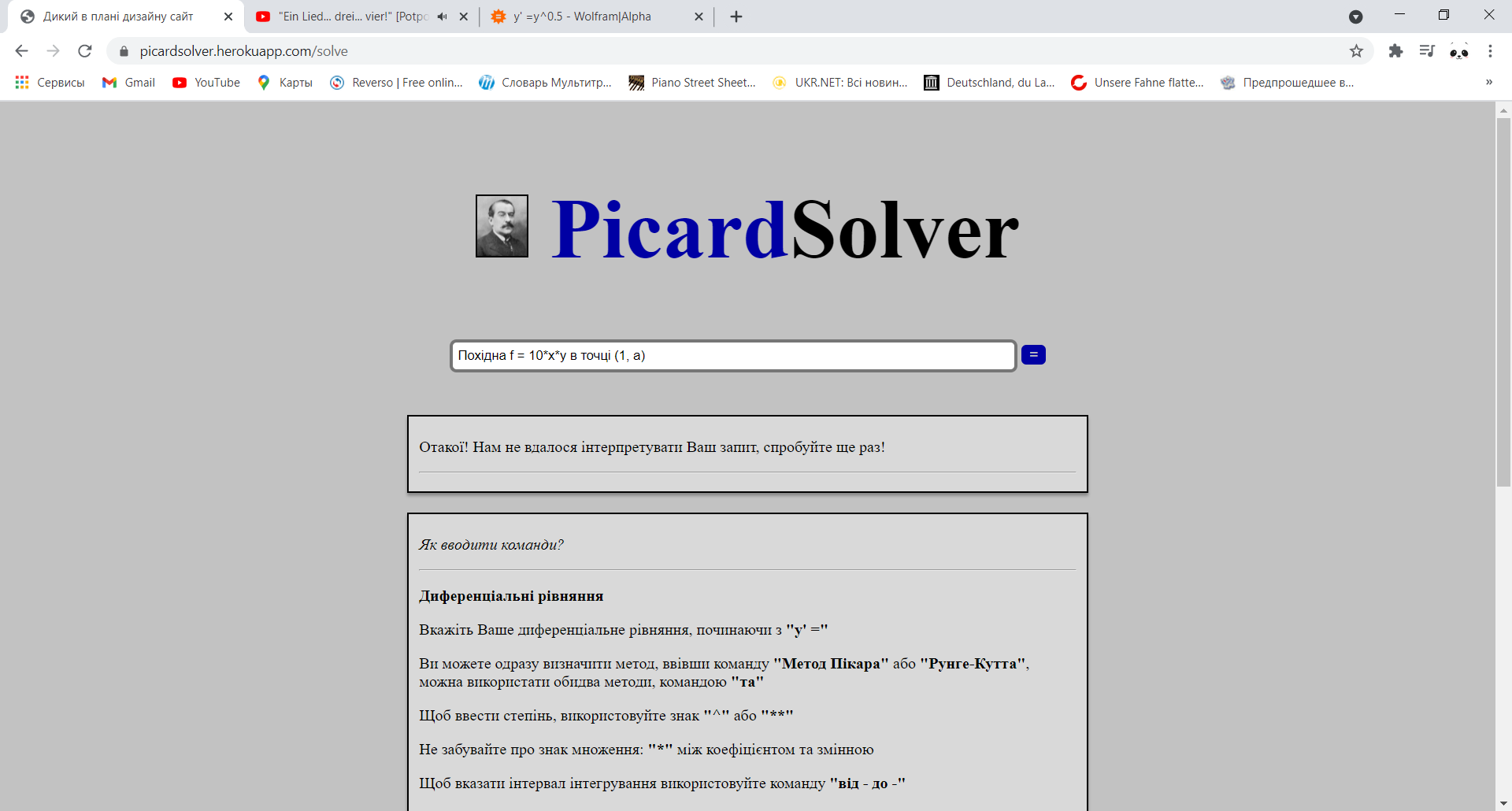


Рисунок 5.28 - Тестовий випадок П.4, некоректні дані замість точки, у якій потрібно знайти похідну

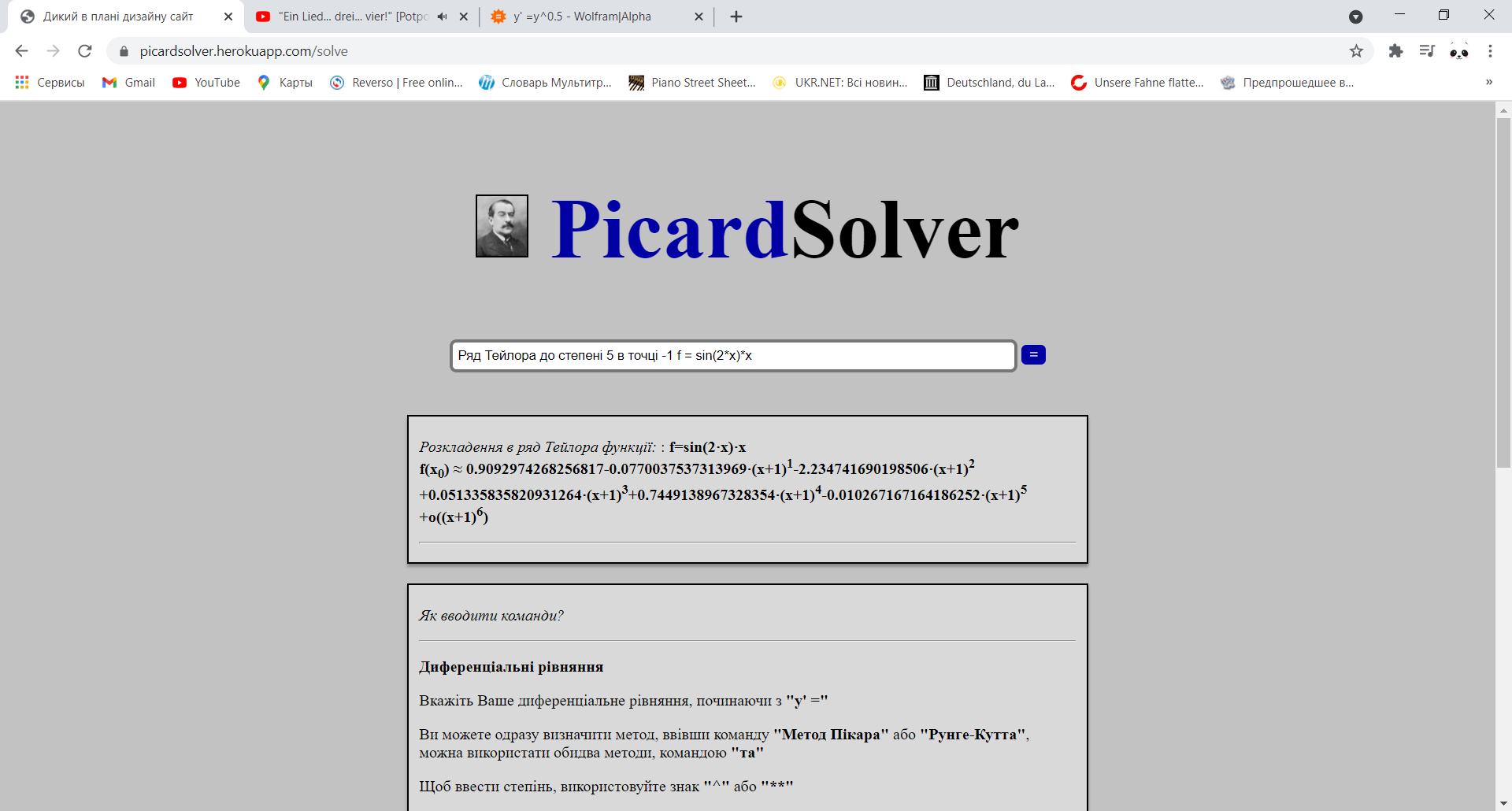


Рисунок 5.29 - Тестовий випадок Р.Т.1, ряд Тейлора функції до заданого порядку у заданій точці

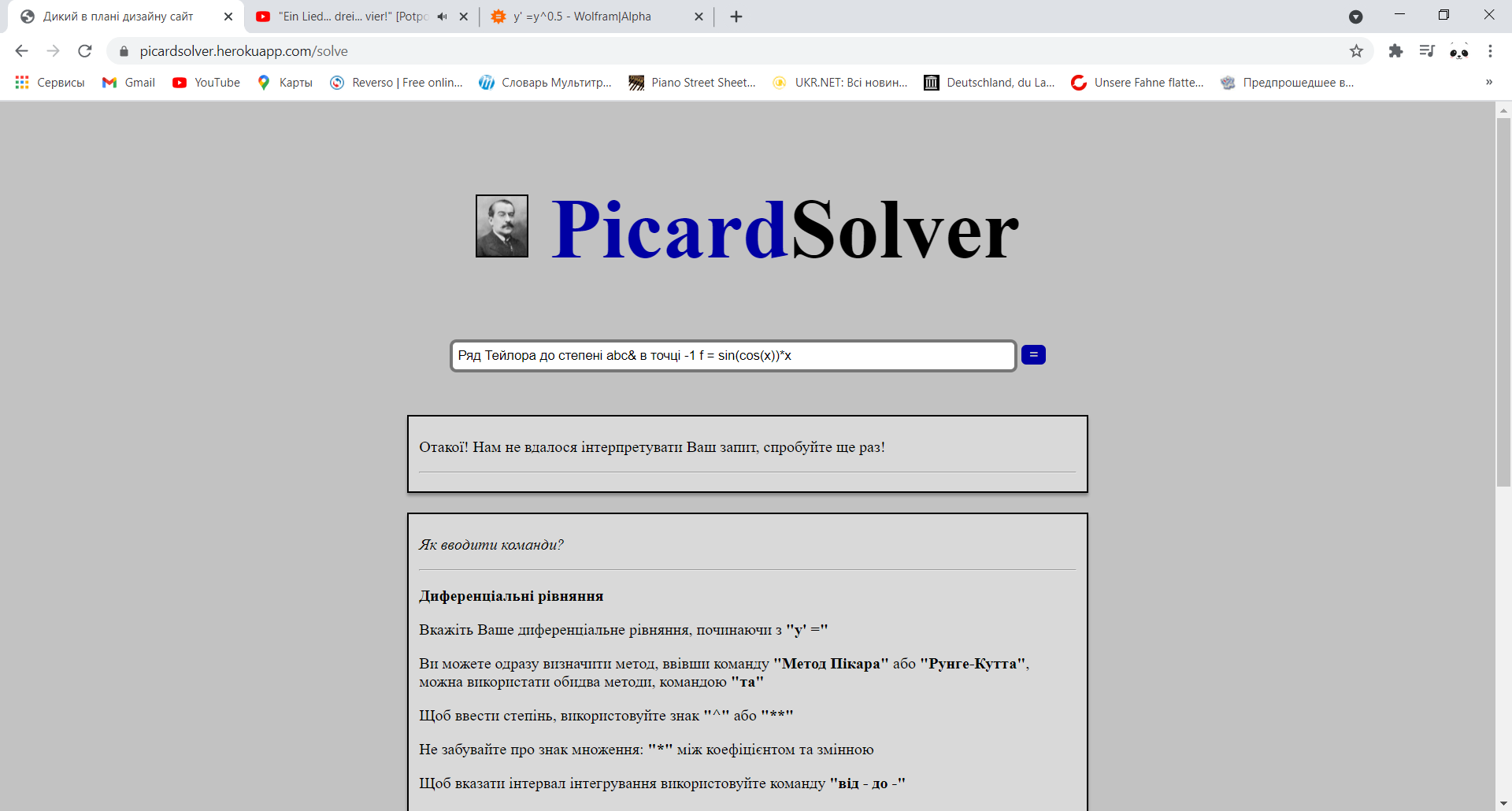


Рисунок 5.30 - Тестовий випадок Р.Т.1, некоректні дані замість степені розкладання

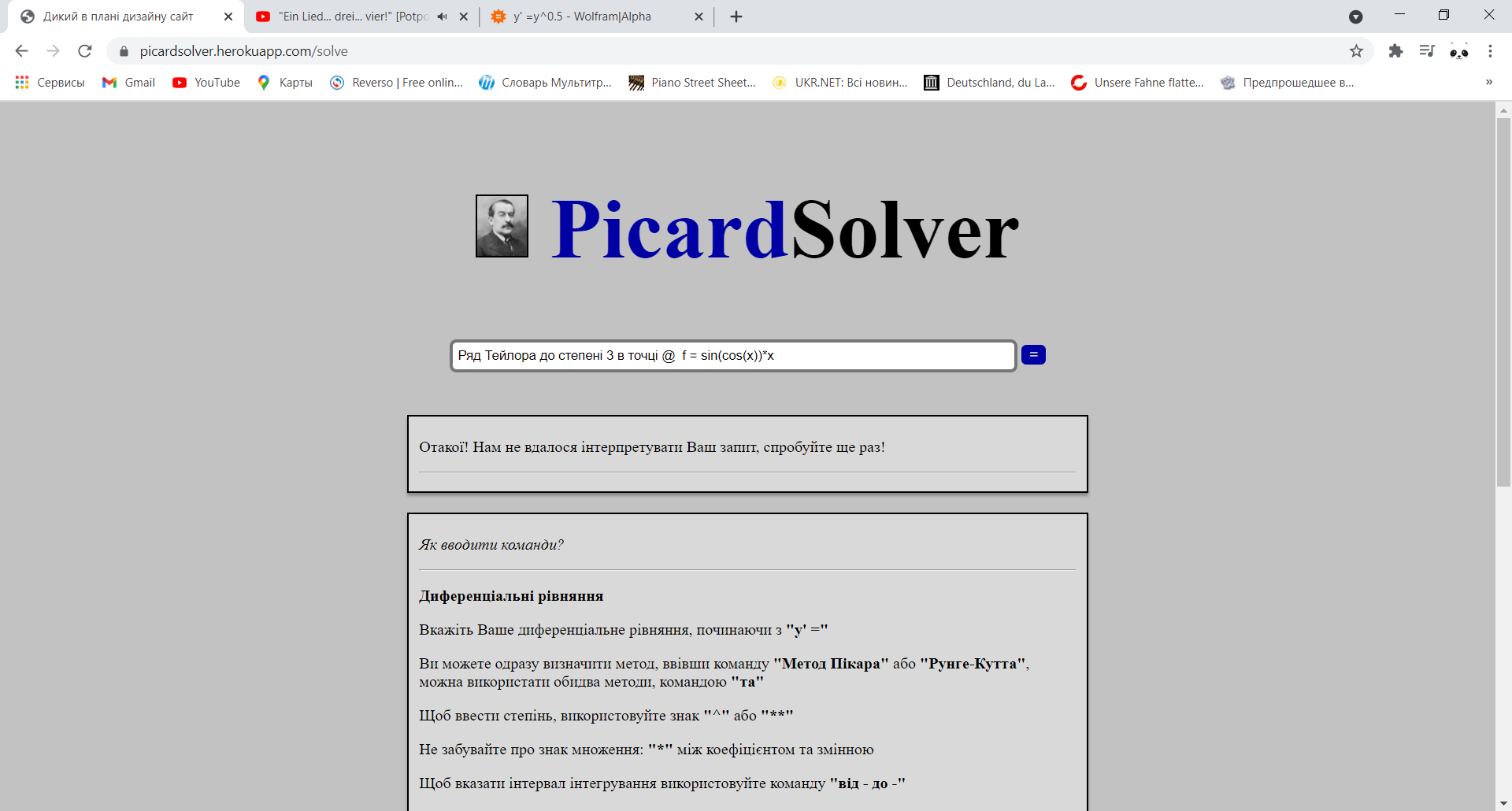


Рисунок 5.31 - Тестовий випадок Р.Т.2, некоректні дані замість точки розкладання

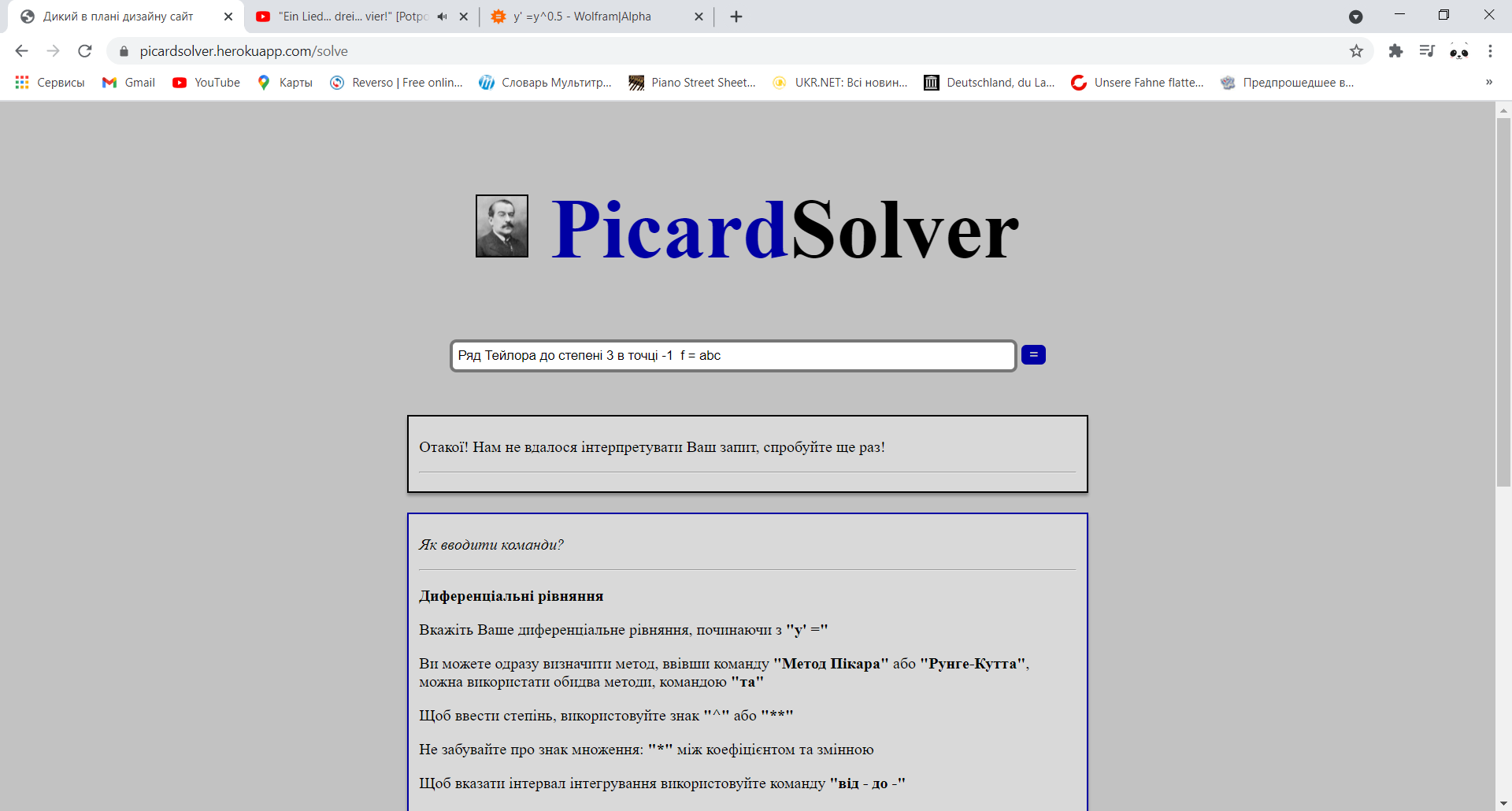


Рисунок 5.32 - Тестовий випадок Р.Т.2, некоректні дані замість функції

## 5.3 Висновки до розділу

У даному розділі було продемонстровано приклади роботи розробленого програмного забезпечення, а також проведено виконання тестових випадків.

Зокрема, було отримано наближене рішення задачі Коші за методом Пікара та порівняно його з точним аналітичним та чисельним за методом Рунге-Кутта розв’язками. Отримані результати можна вважати задовільними і навіть досить точними, адже аналітичний запис наближення, фактично, повторює перші члени ряду Тейлора точного розв’язку.

У ході тестування, відхилень від очікуваних результатів не було отримано. Це, звісно, не гарантує відсутності помилок, але, загалом, робота системи стабільна.

# ВИСНОВКИ

За результатами виконання дипломної роботи було:

а) проведено порівняльний аналіз існуючих чисельних та наближених методів розв’язання задачі Коші для звичайних диференціальних рівнянь. Зокрема, було описано переваги та недоліки таких методів, як методи Ейлера, Рунге-Кутта, багатокрокових чисельних методів Адамса і Мілна, а також наближених методів Чаплигіна та Пікара.

б) описано математичну модель для реалізації методу послідовних наближень для диференціальних рівнянь, права частина яких, задана поліномом: у цьому випадку, ітеративна процедура реалізовується досить просто. Для рівнянь, права частина яких, задана більш загальними функціями, розроблено модифікований алгоритм, який полягає в розбитті інтервалу інтегрування на декілька частин та послідовному знаходженні наближення розв’язку на кожному з них, за допомогою розкладення у ряд Тейлора. Доведено збіжність методу, за такого наближення і оцінено його похибку.

в) програмно реалізовано описану веб-орієнтовану навчальну систему, що розв’язує, методом Пікара, звичайні диференціальні рівняння, а також імплементує метод Рунге-Кутта 4-го порядку для порівняння результатів. Разом з тим, додано функціонал обчислення похідної від скалярної функції і частинних похідних функцій двох змінних, та функціонал розкладу функції у ряд Тейлора. Ці, додаткові функції, є підзадачами, вирішеними при реалізації методу Пікара, але вони також можуть стати у нагоді при розв’язанні диференціальних рівнянь.

г) Проведено тестування розробленої системи, у якому зокрема порівнюються інтегральні криві, отримані за методом Рунге-Кутта та за методом Пікара – вони співпадають досить точно.