МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Лабораторна робота №5

з дисципліни: «Чисельні методи»

16 Варіант

СПЕЦІАЛЬНОСТІ 121 – Інженерія програмного забезпечення

Виконав: Левак О. О.

Група: ІТ-91

Перевірила: Тимофєєва Ю.С.

Київ 2020

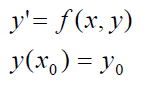
**Лабораторна робота №5. Розв’язування диференціальних рівнянь**

**Мета:** ознайомитися з методами розв’язку диференціальних рівнянь, такими як Ейлера, Рунге-Кутти і Адамса.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

**Задача Коші для одного звичайного диференціального рівняння**

Розглянемо задачу Коші для одного диференціального рівняння першого порядку розв’язаного відносно похідної



Потрібно знайти розв’язок на відрізку [a, b], де x0 = a.

Уведемо різницеву сітку на відрізку [a, b]:





Точки х називають вузлами різницевої сітки, відстані між вузлами – кроком різницевої сітки (h), а сукупність значень якої-небудь велчини заданих у вузлах сітки називається сітковою функцією



Наближений розв’язок задачі Коші будемо шукати чисельно у вигляді сіткової функції. Для оцінки похибки наближеного чисельного розв’язку будемо розглядати його як елемент (N+1)-мірного лінійного векторного простору з якою-небудь нормою. Як похибку розв’язку приймемо норму елемента цього простору



де [y] – точний розв’язок задачі у вузлах розрахункової сітки. У такий спосіб:



## Однокрокові методи: Метод Ейлера (явний)

Метод Ейлера відіграє важливу роль у теорії чисельних методів розв’язування ЗДР, хоча й не часто використовується в практичних розрахунках через не високу точність. Вивести розрахункові співвідношення для цього методу можна декількома способами: за допомогою геометричної інтерпретації, використовуючи розкладання в ряд Тейлора, різницевим методом (за допомогою різницевої апроксимації похідної), квадратним способом (з використанням еквівалентного інтегрального рівняння).

Розглянемо виведення співвідношень методу Ейлера геометричним способом. Розв’язок у вузлі х0 відомий із початкових умов, розглянемо процедуру отримання розв’язку у вузлі х1.

Графік функції , що є розв’язком задачі Коші, являє собою гладку криву, що проходить через точку (х0, у0) відповідно до умови у(х0)=у0, і має в цій точці дотичну. Тангенс кута нахилу дотичній до осі Ох дорівнює значенню похідної від розв’язку в точці х0 й дорівнює значенню правої частини диференціального рівняння в точці (х0, у0) відповідно до виразу y’(x0)=f(x0,y0). У разі невеликого кроку різницевої сітки h графік функції й графік дотичної не встигають сильно розійтися один від одного й можна як значення розв’язку у вузлі х1 прийняти значення дотичної у1, замість значення невідомого точному розв’язку. При цьому допускається похибка



Із прямокутного трикутника АВС знаходимо СВ=ВА tg(CAB) або . Замінюючи похідну на праву частину диференціального рівняння отримаємо співвідношення



Вважаючи тепер точку (х1, у1) початковою та повторюючи всі попередні міркування, отримаємо значення у2 у вузлі х2.

Перехід до довільних індексів дає формулу метода Ейлера:



## Похибка методу Ейлера

На кожному кроці методу Ейлера допускається локально похибка відносно точного розв’язку, графік якого проходить через крайню ліву точку відрізка. Крім того, на кожному кроці, починаючи із другого, накопичується глобальна похибка, що являє собою різницю між чисельним розв’язком і точним розв’язком вихідного початкового завдання (а не локального).

Локальна помилка на кожному кроці визначається співвідношенням

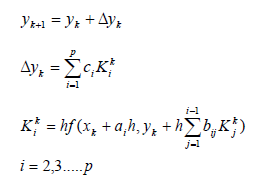


Глобальна похибка методу Ейлера в околиці h=0 поводиться як лінійна функція, і, отже, метод Ейлера має перший порядок точності щодо кроку h.

## Методи Рунге-Кутти

Всі розглянуті вище методи є варіантами методів Рунге-Кутти.

Родина явних методів Рунге-Кутти р-го порядку записується як сукупність формул:

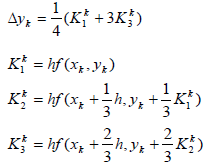


Параметри a, b, c підбираються так, щоб значення yk+1, розраховане за співвідношенням, збігається зі значенням розкладання в точці хk+1 точного розв’язку в ряд Тейлора з похибкою .

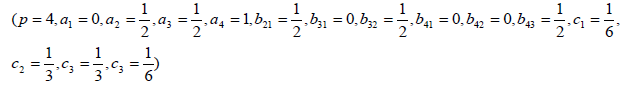
## Метод Рунге-Кутти третього порядку точності

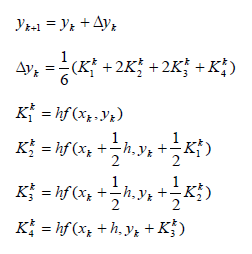
Один з методів Рунге-Кутти третього порядку має вигляд:





## Метод Рунге-Кутти четвертого порядку точності

Метод Рунге-Кутти четвертого порядку точності  є одним із найчастіше використовуваних методів для розв’язування задачі Коші:



## Контроль точності на кожному кроці h

Основним способом контролю точності отриманого чисельного розв’язку при розв’язуванні задачі Коші є методи, що спираються на принцип Рунге-Ромберга-Ричардсона.

Кількість обчислень для такого контролю точності є досить велика, особливо для багатостадійних методів. Тому можна використовувати менш точний спосіб контролю правильному вибору кроку h. У випадку методу Рунге-Кутти четвертого порядку точності треба на кожному кроці h розраховувати параметр

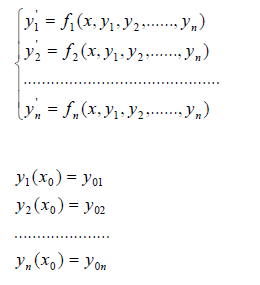


Якщо величина має порядок декількох сотих одиниць, то розрахунок триває з тим же кроком, якщо більше від однієї десятої, то крок варто зменшити, якщо менше від однієї сотої, то крок можна збільшити.

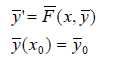
У такий спосіб можна організувати алгоритм вибору кроку h для явного методу Рунге-Кутти.

## Розв’язування задачі Коші для системи звичайних диференціальних рівнянь

Розглянемо задачу Коші для системи диференціальних рівнянь першого порядку, розв’язаних відносно похідної

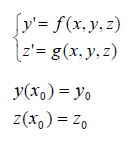


Система у компактнішому вигляді записується у векторній формі

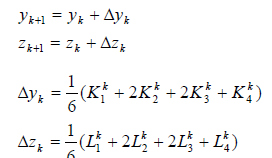


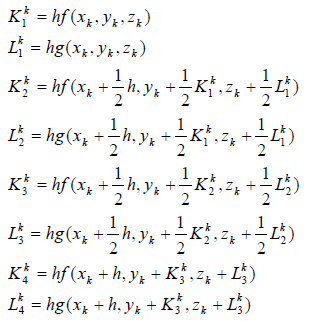
Тут – вектор стовпець невідомих функцій, а – вектор-функція правих частин.

Розглянемо задачу Коші для системи двох ЗДР першого порядку, де рівняння записано в розгорнутому вигляді

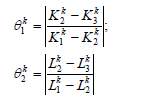


Формули методу Рунге-Кутти 4-го порядку точності для розв’язування такі:





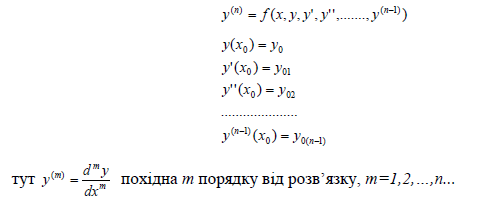
Контролювати правильність вибору кроку h, коли використовується метод Рунге-Кутти четвертого порядку точності для системи, можна обчислюючи на кожному кроці h такі параметри:



Якщо величини мають порядок кількох сотих одиниць, то розрахунок триває з тим же кроком, якщо більше від однієї десятої, то крок варто зменшити, якщо є менше від однієї сотої, то крок можна збільшити.

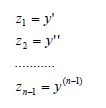
## Розв’язування задачі Коші для ЗДР другого та більш високого порядків

Задача Коші для ЗДР n-го порядку визначається в такий спосіб:

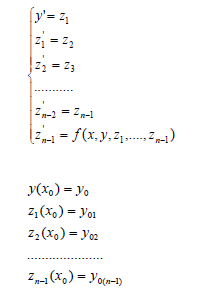


Основний прийом, що використовується полягає у введенні нових змінних і зведені задачі для ЗДР високого порядку до розв’язування системи ЗДР першого порядку.

Введемо нові змінні



Тоді



Отримана система розв’язується будь-яким с описаних методів.

## Багатокрокові методи

Багатокрокові методи розв’язування задачі Коші характеризуються тим, що розв’язок у поточному вузлі залежить від даних не в одному попередньому вузлі, а від декількох попередніх. Розв’язок диференціального рівняння задовольняє інтегрально співвідношення:



## Метод Адамса-Башфорта-Моултона

Цей метод типу предиктор-коректор дозволяє підвищити точність обчислень методу Адамса внаслідок подвійного обчислення значення функції при визначенні yk+1 на кожному новому кроці по х.

**Етап предиктор**

Аналогічно методу Адамса за значенням у вузлах  розраховується «попереднє» значення розв’язку у вузлі xk+1



За допомогою отриманого значення розраховується попереднє значення функції  в новій точці.

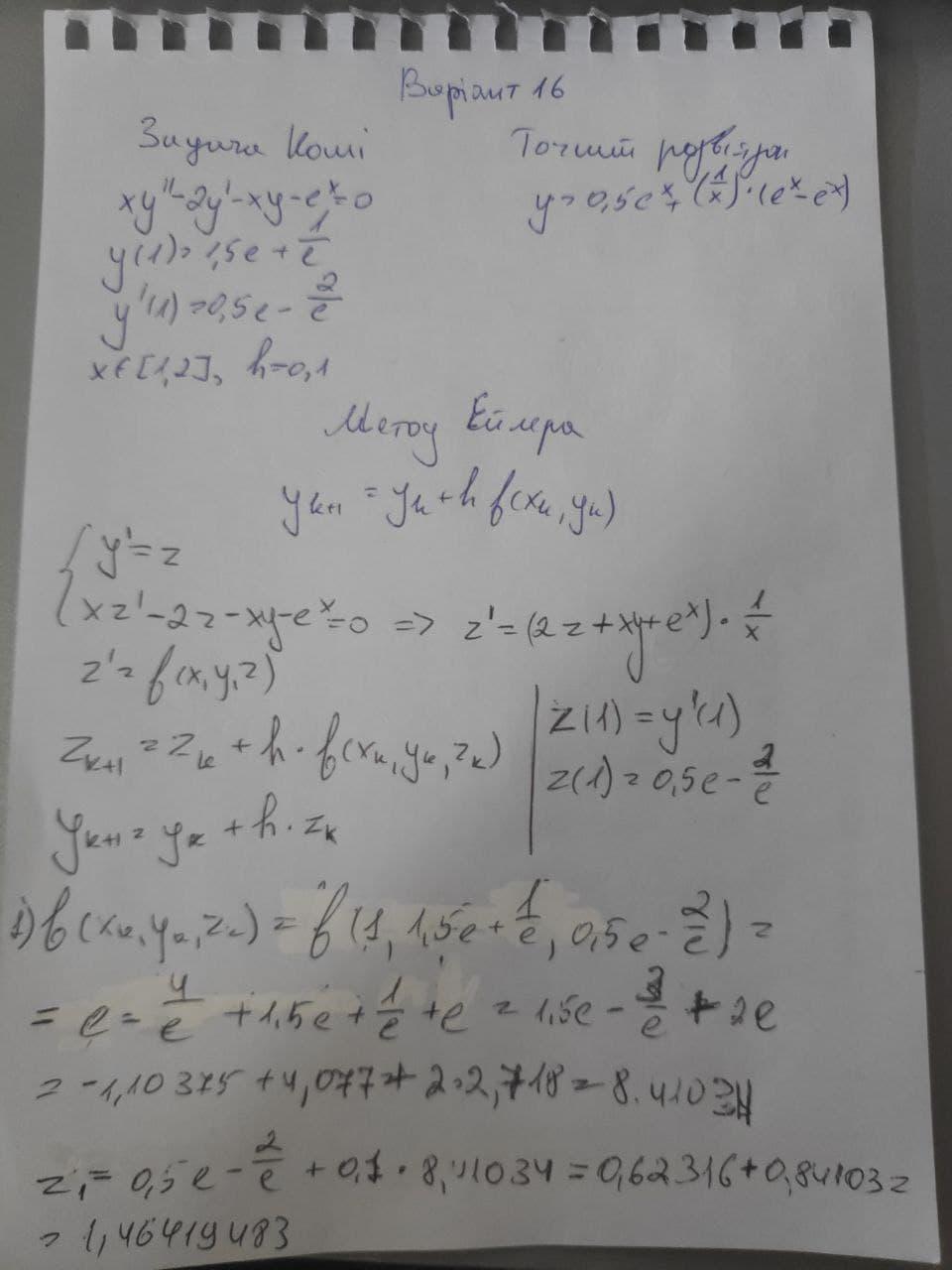
**Етап коректор**

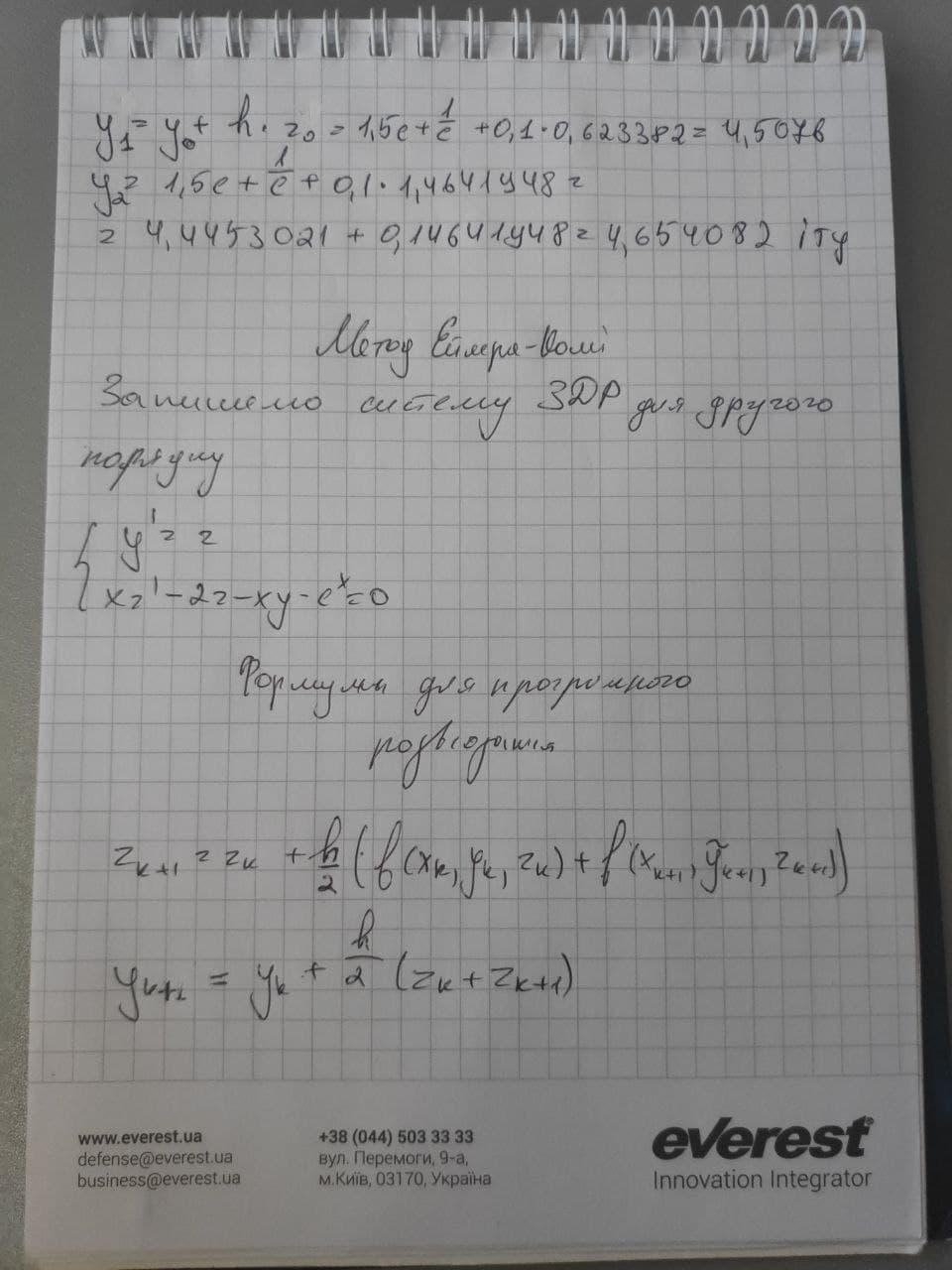
На коригувальному етапі за методом Адамса 4-го порядку за значенням у вузлах  розраховується «остаточне» значення розв’язку у вузлі.



елементарних гармонік).

**РОЗВ’ЯЗОК ЗАДАЧІ В АНАЛІТИЧНІЙ ФОРМІ**

****

****

**ЛІСТИНГ ПРОГРАМИ РОЗВ’ЯЗУВАННЯ ЗАДАЧІ**

**import math**

**def GetPoints(rangeValues, h):**

**return [round(rangeValues[0] + h \* i, 2) for i in range(int((rangeValues[1] - rangeValues[0]) / h) + 1)]**

**def Euler(rangeValues, h, y0, z0, zFunc, isKoshi):**

**y = [y0]**

**z = [z0]**

**x = GetPoints(rangeValues, h)**

**for i in range(len(x)):**

**if isKoshi:**

**if len(x) - 1 == i: break**

**zBeta = z[-1] + (h \* zFunc(x[-1], y[-1], z[-1]))**

**yBeta = y[-1] + (h \* z[-1])**

**z.append(z[-1] + 0.5 \* h \* (zFunc(x[i - 1], y[i - 1], z[i - 1]) + zFunc(x[i], yBeta, zBeta)))**

**y.append(y[-1] + 0.5 \* h \* (z[-1] + z[-2]))**

**else:**

**z.append(z[-1] + h \* zFunc(x[i], y[i], z[i]))**

**y.append(y[i] + h \* z[-2])**

**return y**

**def GetAccuracy(func, rangeValues, h, methodResults):**

**result = []**

**x = GetPoints(rangeValues, h)**

**for i in range(len(x)):**

**result.append(abs(methodResults[i] - func(x[i])))**

**return result**

**def GetFuncResults(func, rangeValues, h):**

**result = []**

**x = GetPoints(rangeValues, h)**

**for i in range(len(x)):**

**result.append(func(x[i]))**

**return result**

**def GetAccuracyInPercent(funcResults, methodResults):**

**result = []**

**for i in range(len(funcResults)):**

**result.append(round(abs(100 - (methodResults[i] \* 100 / funcResults[i])), 2))**

**return result**

**def PrintResults(funcResults, methodResults, accuracy, accuracyInPercent):**

**print("{0:20} | {1:20} | {2:20} | {3} |".format("FuncResults", "MethodResults", "Accuracy", "Accuracy (%)"))**

**print("---------------------|----------------------|----------------------|--------------|")**

**for i in range(len(funcResults)):**

**print("{0:<20} | {1:<20} | {2:<20} | \t{3}%\t |".format(funcResults[i], methodResults[i], accuracy[i],**

**accuracyInPercent[i]))**

**rangeValues = [1, 2]**

**h = 0.1**

**y0 = 1.5 \* math.e + (1 / math.e)**

**z0 = 0.5 \* math.e - (2 / math.e)**

**def func(x):**

**return 0.5 \* (math.e \*\* x) + (1 / x) \* (math.e \*\* x - math.e \*\* (-x))**

**def zFunc(x, y, z):**

**return (2 \* z + x \* y + math.e \*\* x) / x**

**funcResults = GetFuncResults(func, rangeValues, h)**

**eulerResults = Euler(rangeValues, h, y0, z0, zFunc, 0)**

**eulerKoshiResults = Euler(rangeValues, h, y0, z0, zFunc, 1)**

**eulerAccuracy = GetAccuracy(func, rangeValues, h, eulerResults)**

**eulerKoshiAccuracy = GetAccuracy(func, rangeValues, h, eulerKoshiResults)**

**eulerAccuracyInPercent = GetAccuracyInPercent(funcResults, eulerResults)**

**eulerKoshiAccuracyInPercent = GetAccuracyInPercent(funcResults, eulerKoshiResults)**

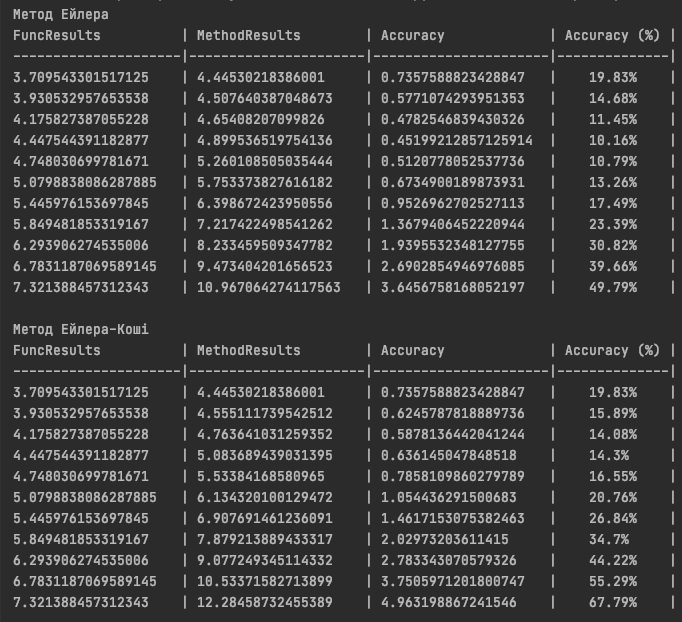
**print("Метод Ейлера")**

**PrintResults(funcResults, eulerResults, eulerAccuracy, eulerAccuracyInPercent)**

**print("Метод Ейлера-Коші")**

**PrintResults(funcResults, eulerKoshiResults, eulerKoshiAccuracy, eulerKoshiAccuracyInPercent)**

**РЕЗУЛЬТАТИ ВИКОНАННЯ ПРОГРАМ**

****

**Висновки :**

В ході лабораторної роботи ми ознайомилися з методами розв’язку диференціальних рівнянь, такими як Ейлера, Рунге-Кутти і Адамса. Розв’язок, отриманий за методом Адамса-Башфорта-Мултона, трохи точніший, ніж розв’язок, отриманий за методом Адамса. А поданий у методичці розв\*язок є не таким точним, тому він не збігається з розв\*язками за методами.