МIНIСТЕРСТВО ОСВIТИ I НАУКИ УКРАЇНИ

НАЦIОНАЛЬНИЙ ТЕХНIЧНИЙ УНIВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛIТЕХНIЧНИЙ IНСТИТУТ»

Кафедра прикладної математики

Звіт

з лабораторної роботи №3

з дисципліни «Алгоритми і системи комп’ютерної математики-2.

Програмні засоби»

на тему:

«Розв’язання звичайних диференційних рівнянь вищих порядків та

систем диференційних рівнянь»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-63 | *Старший викладач Бай Ю.П.* |
| *Вовченко І.В..* |  |

KM-6304.docx

KM-6304.py

KM-6304.m

Київ — 2020

# **ЗМІСТ**

[1 ВСТУП 2](#_Toc36047690)

[2 ОСНОВНА ЧАСТИНА 3](#_Toc36047691)

[2.1 Постановка задачі 3](#_Toc36047692)

[2.2 Описання методу 3](#_Toc36047693)

[2.3 Порядок виконання роботи 5](#_Toc36047694)

[2.5 Контрольні запитання 9](#_Toc36047695)

[2.6 Опис програмних засобів 10](#_Toc36047696)

[3 ВИСНОВКИ 12](#_Toc36047697)

[4 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 13](#_Toc36047698)

[ДОДАТКИ 14](#_Toc36047699)

[Додаток А (код програми GNU Octave) 14](#_Toc36047700)

[Додаток Б (код програми Python) 15](#_Toc36047701)

# **1 ВСТУП**

Темою для даної лабораторної роботи є «Розв’язання звичайних диференційних рівнянь вищих порядків та систем диференційних рівнянь», а саме метод Ейлера.

Проаналізувати отримані розв’язки за допомогою відповідних вбудованих функцій бібліотек Python та GNUOctave. Побудувати порівняльні таблиці для чотирьох груп роз в’язків.

Побудувати графіки роз в’язків, отриманих за допомогою бібліотечних функцій та власноруч розроблених програм.

# **2 ОСНОВНА ЧАСТИНА**

# **2.1 Постановка задачі**

За допомогою Python та GNUOctave розв’язати задачу Коші для звичайного диференційного рівняння другого порядку з точністю ε=0,001.

Варіант №5:

Диференційне рівняння:

Початкові умови:

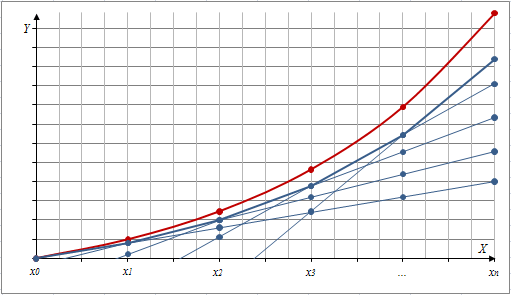
Інтервал:

Крок: 0.1

# **2.2 Описання методу**

Для використання методу Ейлера на рівняннях вищого порядку, для початку потрібно понизити степінь самого рівняння за допомогою методу пониження степені.

Метод Ейлера — один з найпростіших чисельних алгоритмів розв'язку звичайних диференціальних рівнянь першого порядку з заданим початковим значенням тобто задачі Коші. Він є явним, однокроковим методом першого порядку точності, основна ідея якого полягає в тому, що інтегральна крива апроксимується кусочно-лінійною функцією, так званою ламаною Ейлера.



*Геометрична інтерпретація методу Ейлера*

Розглянемо даний процес більш детально. Для цього запишемо диференціальне рівняння наступного вигляду:

з початковою умовою і припустимо, що потрібно занйти його ровз'язок на деякому інтервалі [a, b]. Для цього розіб'ємо заданий інтервал на *n* частин з кроком . В результаті отримаємо систему рівновіддалених точок:

де .

Припустим, що шуканим розв'язком задачі Коші є функція . Побудуємо дотичну до графіка даної функції в точці і запишемо її рівняння:

Знайдемо точку перетину даної дотичної з прямою . В результаті отримаємо . Беручи тепер за нову вихідну точку, аналогічним чином будуємо до неї дотичну:

і знаходимо точку перетину даної дотичної з прямою . Продовжуючи даний поцес далі, отримаємо рекурентну послідовність:

яку називають послідовністю Ейлера. З'єднюючи всі точки, які були знайдені з допомогою даної послідовності, отримаємо ламану лінію (ламану Ейлера), графік якої і будемо приймати в якості наближеного розв'язку задачі Коші.

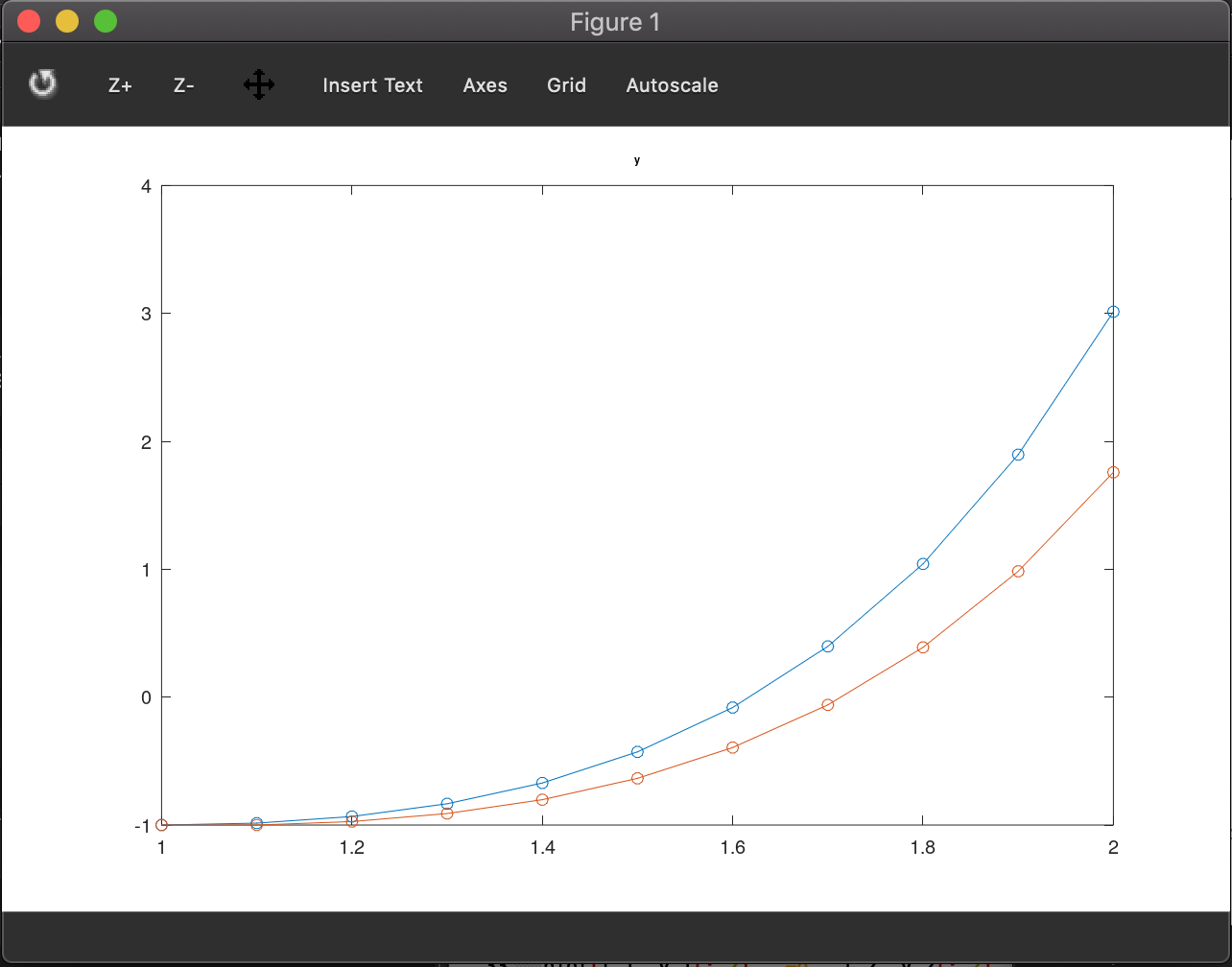
# **2.3 Порядок виконання роботи**

1. Записуємо вхідні дані відповідно умови варіанту, а саме: рівняння, крок, точність, початкові умови та інтервал.
2. За допомогою формул наведених вище обчислюємо результат.
3. Аналізуємо отримані розв’язки за допомогою відповідних вбудованих функцій бібліотек Python та GNUOctave. Будуємо порівняльні таблиці для чотирьох груп роз в’язків.
4. Будуємо графіки роз в’язків, отриманих за допомогою бібліотечних функцій та власноруч розроблених програм.

Представлену вище послідовність дій необхідно реалізувати у вигляді програмного забезпечення двома способами: використовуючи мову Python, а також за допомогою програмного пакету математичних обчислень GNU Octave.

**2.4 Результат виконання**

GNUOctave:



Графік залежності X від Y

A close up of a map

Description automatically generated

Графік залежності X від Z

Python:

A close up of a device

Description automatically generated

Графік залежності X від Y

A close up of a map

Description automatically generated

Графік залежності X від Z

|  |  |  |
| --- | --- | --- |
| X | Y(octave) | Y(python) |
| 1.0 | -1.00000 | -1 |
| 1.1 | -0.98497 | -0.98496664 |
| 1.2 | -0.93338 | -0.93337602 |
| 1.3 | -0.83364 | -0.83363948 |
| 1.4 | -0.67124 | -0.67123526 |
| 1.5 | -0.42802 | -0.42802004 |
| 1.6 | -0.08138 | -0.08138467 |
| 1.7 | 0.39678 | 0.39678183 |
| 1.8 | 1.04136 | 1.04136261 |
| 1.9 | 1.89557 | 1.89557003 |
| 2.0 | 3.01285 | 3.01285349 |

Порівняльна таблиця y(octave) та у(python)

|  |  |  |
| --- | --- | --- |
| X | Z(octave) | Z(python) |
| 1.0 | -1.00000 | -1.0 |
| 1.1 | -1.00000 | -1.0 |
| 1.2 | -0.97282 | -0.97281718 |
| 1.3 | -0.91016 | -0.91015614 |
| 1.4 | -0.80176 | -0.80176172 |
| 1.5 | -0.63500 | -0.63499545 |
| 1.6 | -0.39432 | -0.39432392 |
| 1.7 | -0.06070 | -0.06070121 |
| 1.8 | 0.38918 | 0.38917638 |
| 1.9 | 0.98377 | 0.98376896 |
| 2.0 | 1.75778 | 1.75777653 |

Порівняльна таблиця z(octave) та z(python)

# **2.5 Контрольні запитання**

1. Для чого потрібен метод Ейлера?

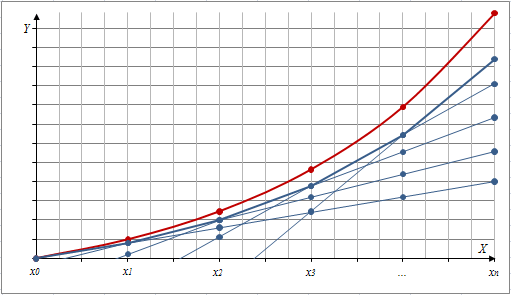
Для розв'язку звичайних диференціальних рівнянь першого порядку з заданим початковим значенням

1. Як виглядає послідовність Ейлера?

1. Як можна використати даний метод для рівнянь вищого порядку?

Для використання методу Ейлера на рівняннях вищого порядку, для початку потрібно понизити степінь самого рівняння за допомогою методу пониження степені.

1. Як виглядає геометрична інтерпретація методу Ейлера?



1. Яка бібліотека в Python дозволяє взаємодіяти з графіками?

matplotlib.pyplot

# **2.6 Опис програмних засобів**

В процесі реалізації поставленої задачі були використані наступні вбудовані функції бібліотек numpy (np), scipy, matplotlib мови Python:

1. - функція arange() повертає одновимірний масив з рівномірно рознесеними значеннями всередині заданого інтервалу.
2. – додає елемент до масиву;
3. *–* функція odeint () має багато опцій, керуючих її роботою. Опції rtol (відносна похибка) і atol (абсолютна похибка) визначають похибка обчислень e.
4. - повертає значення експоненти в степені
5. – виконує побудову графіку функції, що передається параметром *func()* на відрізку, що заданий параметром *np.arange()*;

В процесі реалізації поставленої задачі були використані наступні вбудовані функції GNU Octave:

1. – реалізований метод Ейлера;
2. *–* повертає рішення системи;
3. – повертає значення експоненти в степені
4. – дозволяє виконати побудову графіку функції.

# **3 ВИСНОВКИ**

В даній лабораторній роботі було розроблено програмне забезпечення, що реалізує метод Ейлера для диф. рівнянь вищого порядку. Проаналізувано отримані розв’язки за допомогою відповідних вбудованих функцій бібліотек Python та GNUOctave.

Побудувати графіки роз в’язків, отриманих за допомогою бібліотечних функцій та власноруч розроблених програм.

# **4 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. [Линник Ю. В](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%BD%D0%B8%D0%BA,_%D0%AE%D1%80%D0%B8%D0%B9_%D0%92%D0%BB%D0%B0%D0%B4%D0%B8%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B8%D1%87_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA)). Метод наименьших квадратов и основы математико-статистической теории обработки наблюдений. — 2-е изд. — М., 1962.
2. Лоран, П. Ж. Аппроксимация и оптимизация. — М.: Мир, 1975. — 496 с.
3. Е. Р. Алексеев, О. В. Чеснокова. Введение в Octave для инженеров и математиков — С.326-335.

# **ДОДАТКИ**

# **Додаток А (код програми GNU Octave)**

h = 0.1

a = 1

b = 2

y0 = -1

z0 = 0

function output = myode45(t, y)

output = [y(2); 2\*y(2) + exp(t)];

endfunction

function[y,t]=eiler(a,b,h,y0,z0)

n = (b-a)/h;

y(1,:) = [y0, z0];

for i = 1:n+1

t(i) = a + (i-1)\*h;

end

for i = 2:n+1

temporary = myode45(t(i-1),y(i-1,:));

y(i,:) = [y(i-1, 1)+h\*temporary(1) y(i-1,2)+h\*temporary(2)];

end

end

par=odeset ('InitialStep' ,h , 'MaxStep' ,h) ;

[t\_1, y\_1] = ode45("@myode45", [a, b], [y0, z0], par)

[y\_2,t\_2] = eiler(a, b, h, y0, z0)

plot(t\_1, y\_1(:,1), "-o", t\_2, y\_2(:,1), "-o"), title('y');

figure;

plot(t\_1, y\_1(:,2), "-o", t\_2, y\_2(:,2), "-o"), title('y\_dot');

# **Додаток Б (код програми Python)**

**from** scipy.integrate **import** odeint  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** math **import** exp  
  
h = 0.1  
a = 0  
b = 1  
y0 = -1  
z0 = 0  
  
  
**def** funct(u, x):  
 y = u[0]  
 z = u[1]  
 **return** [z, 2\*z + exp(x)]  
  
  
x = np.arange(a, b+h, h)  
beg = [y0, z0]  
solution = odeint(funct, beg, x)  
  
n = int((b - a) / h)  
y = [[y0, z0]]  
  
t = []  
**for** i **in** range(0, n + 1):  
 t.append(a + (i) \* h)  
  
**for** i **in** range(1, n + 1):  
 temp = funct(y[i - 1], t[i - 1])  
 y.append([y[i - 1][0] + h \* temp[0], y[i - 1][1] + h \* temp[1]])  
  
  
  
plt.plot(x, solution[:, 0], **'b'**, label=**'theta(t)'**)  
plt.plot(x, np.array(y)[:, 0], **'g'**, label=**'omega(t)'**)  
print(**'dependencies x of y'**)  
print(x, solution[:, 0])  
print(x, np.array(y)[:, 0])  
plt.legend(loc=**'best'**)  
plt.xlabel(**'t'**)  
plt.grid()  
plt.show()  
  
plt.plot(x, solution[:, 1], **'b'**, label=**'theta(t)'**)  
plt.plot(x, np.array(y)[:, 1], **'g'**, label=**'omega(t)'**)  
print(**'dependencies x of z'**)  
print(x, solution[:, 1])  
print(x, np.array(y)[:, 1])  
plt.legend(loc=**'best'**)  
plt.xlabel(**'tн'**)  
plt.grid()  
plt.show()

**Додаток В (результат виконання Octave)**

h = 0.10000

a = 1

b = 2

y0 = -1

z0 = 0

t\_1 =

1.0000

1.1000

1.2000

1.3000

1.4000

1.5000

1.6000

1.7000

1.8000

1.9000

2.0000

y\_1 =

-1.00000 0.00000

-0.98497 0.31595

-0.93338 0.73508

-0.83364 1.28374

-0.67124 1.99445

-0.42802 2.90737

-0.08138 4.07198

0.39678 5.54923

1.04136 7.41409

1.89557 9.75875

3.01285 12.69648

y\_2 =

-1.00000 0.00000

-1.00000 0.27183

-0.97282 0.62661

-0.91016 1.08394

-0.80176 1.66766

-0.63500 2.40672

-0.39432 3.33623

-0.06070 4.49878

0.38918 5.94593

0.98377 7.74008

1.75778 9.95668

t\_2 =

1.0000 1.1000 1.2000 1.3000 1.4000 1.5000 1.6000 1.7000 1.8000 1.9000 2.0000

**Додаток Г (результат виконання Python)**

dependencies x of y

[0. 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ] [-1. -0.99446954 -0.97549041 -0.93879941 -0.87905422 -0.78958034

-0.66206032 -0.48615269 -0.24902467 0.06522067 0.47624629]

[0. 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ] [-1. -1. -0.99 -0.96694829 -0.92707221 -0.86572233

-0.77718422 -0.65445128 -0.48895057 -0.27021218 0.01452929]

dependencies x of z

[0. 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ] [0. 0.11623183 0.27042193 0.47226 0.73371625 1.06956059

1.49799817 2.04144732 2.72749159 3.59004446 4.6707744 ]

[0. 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ] [0. 0.1 0.23051709 0.39876079 0.61349882 0.88538106

1.2273294 1.65500716 2.18738386 2.84741472 3.66285798]