Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота №8

з дисципліни «Спеціальні розділи математики»

на тему

«Розв’язання задачі Коші»

Виконав:

студент гр. ІС-02

Плостак Ілля

Викладач:

доц. Рибачук Л.В.

Київ – 2021

**1. Постановка задачі**

Методами Рунге-Кутта та Адамса розв'язати задачу Коші. На початку інтервалу у

необхідній кількості точок значення для методу Адамса визначити методом Рунге-Кутта

четвертого порядку.

Для фіксованого h потрібно навести:

− значення наближеного розв'язку y(x) у тих самих точках, одержані обома методами;

− значення функції помилки ε(x) для обох методів;

графіки:

− обох наближених - на одному малюнку;

− обох помилок - на другому малюнку.

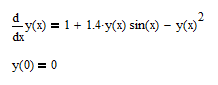
Розв’язати задане рівняння за допомогою Matchad, порівняти із власними

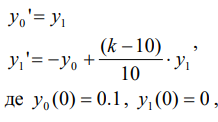
результатами.

Розв’язати за допомогою Matchad систему рівнянь, побудувати графік 0 y та

фазовий портрет системи, зробити висновки щодо стійкості системи.

**2. Вихідне рівняння та система:**



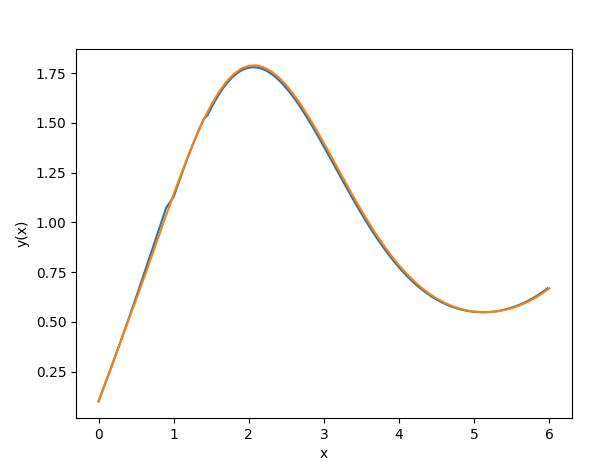


k = 1

**3. Значення наближеного розв'язку y(x) у тих самих точках, одержані обома методами:**



**4. Спільний графік значень обох наближених методів:**

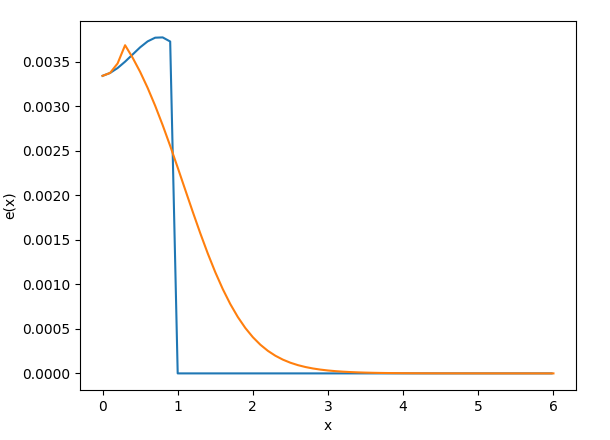


Синій – Рунге-Кутта

Помаранчевий – Адамса

**5. Значення функції помилки ε для обох методів у всіх точках xi:**

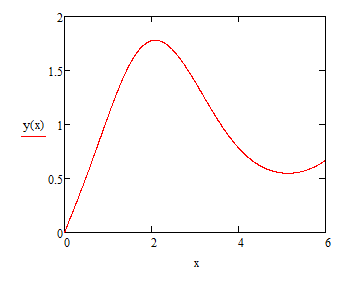


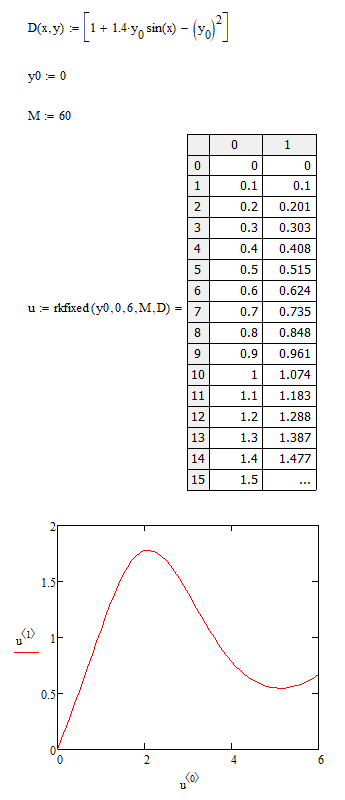
**6.** **Спільний графік помилок ε для обох методів**

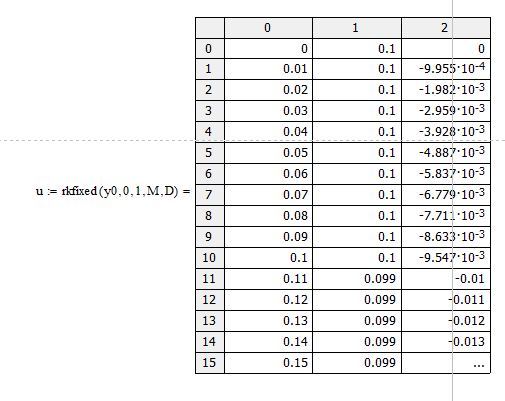
Синій – Рунге-Кутта

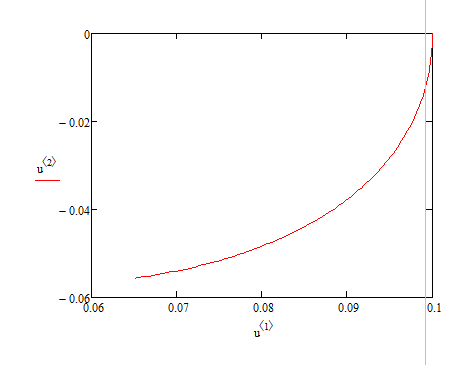
Помаранчевий – Адамса

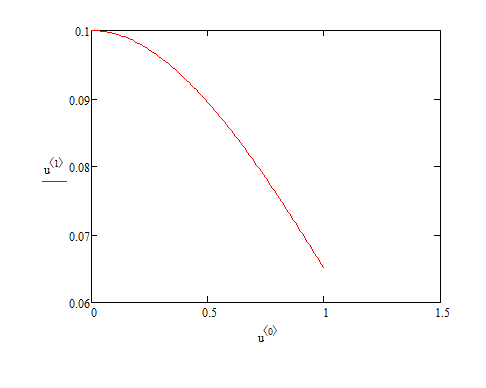
**7. Графік y0 та фазовий портрет системи**

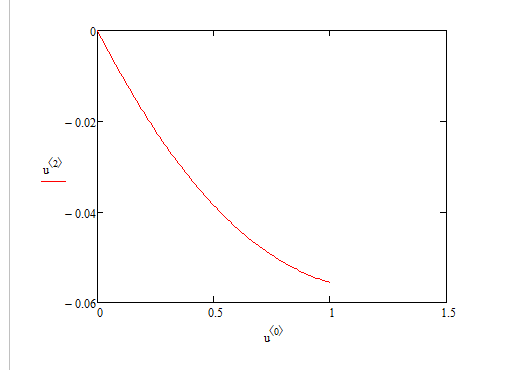












**8. Лістинг програми:**

[main.py](https://github.com/feedblackg44/kpilabs2/blob/master/CHM/Lab8/main.py):

from functions import RungeKutta, Adams, beautifulPrint, calcualteE  
from math import sin  
from matplotlib.pyplot import plot, show, xlabel, ylabel  
  
func = lambda x, y: 1+1.4\*y\*sin(x)-y\*\*2  
a = 0  
b = 6  
y0 = 0  
h = 0.1  
  
rungeX1, rungeY1 = RungeKutta(func, y0, a, b, h)  
rungeX2, rungeY2 = RungeKutta(func, y0, a, b, h/2)  
rungeE = calcualteE(rungeX1, rungeX2, rungeY1, rungeY2)  
adamsX1, adamsY1 = Adams(func, y0, a, b, h)  
adamsX2, adamsY2 = Adams(func, y0, a, b, h/2)  
adamsE = calcualteE(adamsX1, adamsX2, adamsY1, adamsY2)  
strOut = ""  
strOut += "Перший метод (Рунге-Кутта):\n"  
strOut += beautifulPrint(rungeY1, 5)  
strOut += "Другий метод (Адамса):\n"  
strOut += beautifulPrint(adamsY1, 5)  
strOut += "Похибки першого методу (Рунге-Кутта):\n"  
strOut += beautifulPrint(rungeE, 5)  
strOut += "Похибки другого методу (Адамса):\n"  
strOut += beautifulPrint(adamsE, 5)  
file = open("output.txt", "w")  
file.write(strOut)  
file.close()  
plot(rungeX1, rungeY1)  
plot(adamsX1, adamsY1)  
ylabel("y(x)")  
xlabel("x")  
show()  
plot(rungeX1, rungeE)  
plot(adamsX1, adamsE)  
ylabel("e(x)")  
xlabel("x")  
show()

[functions.py](https://github.com/feedblackg44/kpilabs2/blob/master/CHM/Lab8/functions.py):

from math import ceil  
  
def RungeKutta(func, y0, a, b, h0):  
 x0 = a  
 x = x0  
 y = y0  
 arrX = []  
 arrY = []  
 h = h0  
 while x <= b:  
 k1 = func(x0, y0)  
 k2 = func(x0 + h / 2, y0 + h \* k1 / 2)  
 k3 = func(x0 + h / 2, y0 + h \* k2 / 2)  
 k4 = func(x0 + h, y0 + h \* k3)  
 t = abs((k2 - k3) / (k1 - k2))  
 while t > 0.04:  
 h /= 2  
 k1 = func(x0, y0)  
 k2 = func(x0+h/2, y0+h\*k1/2)  
 k3 = func(x0+h/2, y0+h\*k2/2)  
 k4 = func(x0+h, y0+h\*k3)  
 t = abs((k2 - k3) / (k1 - k2))  
 y = y0 + h/6\*(k1+2\*k2+2\*k3+k4)  
 arrX.append(round(x, 6))  
 arrY.append(y)  
 x += h  
 y0 = y  
 x0 = x  
  
 return arrX, arrY  
  
def Adams(func, y0, a, b, h0):  
 h = h0  
 arrX, arrY = RungeKutta(func, y0, a, a+h\*3, h)  
 i = 3  
 while arrX[i] <= b:  
 y\_ex = arrY[i] + (h / 24) \* (55 \* func(arrX[i], arrY[i]) -  
 59 \* func(arrX[i - 1], arrY[i - 1]) +  
 37 \* func(arrX[i - 2], arrY[i - 2]) -  
 9 \* func(arrX[i - 3], arrY[i - 3]))  
 y\_cor = arrY[i] + (h / 24) \* (9 \* func(arrX[i] + h, y\_ex) +  
 19 \* func(arrX[i], arrY[i]) -  
 5 \* func(arrX[i - 1], arrY[i - 1]) +  
 func(arrX[i - 2], arrY[i - 2]))  
 while abs(y\_cor - y\_ex) > 0.04:  
 h /= 2  
 y\_ex = arrY[i] + (h/24)\*(55\*func(arrX[i], arrY[i]) -  
 59\*func(arrX[i-1], arrY[i-1]) +  
 37\*func(arrX[i-2], arrY[i-2]) -  
 9\*func(arrX[i-3], arrY[i-3]))  
 y\_cor = arrY[i] + (h/24)\*(9\*func(arrX[i]+h, y\_ex) +  
 19\*func(arrX[i], arrY[i]) -  
 5\*func(arrX[i-1], arrY[i-1]) +  
 func(arrX[i-2], arrY[i-2]))  
  
 arrX.append(round(arrX[i] + h, 6))  
 arrY.append(y\_cor)  
 i += 1  
 arrX.pop(len(arrX)-1)  
 arrY.pop(len(arrY)-1)  
 return arrX, arrY  
  
def calcualteE(arrX1, arrX2, arrY1, arrY2):  
 arrE = []  
 for i in range(len(arrX1)):  
 tempIndex = arrX2.index(arrX1[i])  
 arrE.append(abs(arrY1[i] - arrY2[tempIndex])/(2\*\*4 - 1))  
 return arrE  
  
def beautifulPrint(arr, num):  
 strOut = ""  
 newArr = [[None for i in range(num)] for i in range (ceil(len(arr) / num))]  
 i = 0  
 j = 0  
 k = 0  
 while i < len(arr):  
 if j < len(newArr):  
 if k < len(newArr[j]):  
 newArr[j][k] = arr[i]  
 i += 1  
 k += 1  
 else:  
 k = 0  
 j += 1  
 else:  
 break  
 for g in range(len(newArr[len(newArr) - 1]) - 1, -1, -1):  
 if newArr[len(newArr) - 1][g] == None:  
 newArr[len(newArr) - 1].pop(g)  
 for g in range(len(newArr)):  
 for u in range(len(newArr[g])):  
 strOut += str(newArr[g][u]) + " "  
 strOut += "\n"  
 return strOut