Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки
Кафедра Обчислювальної Техніки

Лабораторна робота № 8

з дисципліни «Чисельні методи»

на тему

«**Розв’язання задачі Коші**»

Виконав:
студент гр. ІП-93
Домінський Валентин

Викладач:
доц. Рибачук Л.В.

Київ – 2021

Зміст

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 у та фазовий портрет системи $u^{\langle 2\rangle}$ ($u^{\langle 1\rangle}$) , зробити висновки щодо стійкості системи.

2 Розв’язок

Вивід програми:

```
My variant: y' = e^(-ax)*(y^(2)+b), with y(0) = 0 , intervals = [ 0 , 4 ] and h = 0.1

Runge Kutta
iterations      x       y                       fault
0               0       0       0.000000000000000
1               0.1     0.23050646574222378     0.000000437561509
2               0.2     0.4142588412688023      0.000000662387270
3               0.3     0.562936868021617       0.000000803684329
4               0.4     0.6835931344030579      0.000000903386957
5               0.5     0.781219973981651       0.000000978628499
6               0.6     0.781219973981651       0.002761728925931
7               0.7     0.8597742584224248      0.002211180353480
8               0.8     0.9225809661832618      0.001759174842239
9               0.9     0.9724862079907614      0.001391452561454
10              1.0     1.0119188236949914      0.001095035972226
11              1.1     1.0429267651951084      0.000858085598867
12              1.2     1.0672120535180694      0.000670027520824
13              1.3     1.1009293204890005      0.000001221776649
14              1.4     1.1123962450184135      0.000001231068098
15              1.5     1.121290166185456       0.000001238329682
16              1.6     1.1281793701387028      0.000001243989076
17              1.7     1.1335102433556257      0.000001248389793
18              1.8     1.1376319731750961      0.000001251805537
19              1.9     1.1408168300090313      0.000001254452923
20              2.0     1.143276575295172       0.000001256502442
21              2.1     1.145175586272003       0.000001258087686
22              2.2     1.146641264801965       0.000001259312965
23              2.3     1.147772238389576       0.000001260259499
24              2.4     1.148644789418553       0.000001260990389
25              2.5     1.1493178760938665      0.000001261554578
26              2.6     1.149837042046218       0.000001261989975
27              2.7     1.1502374537088746      0.000001262325915
28              2.8     1.1505462559622812      0.000001262585076
29              2.9     1.1507843966187736      0.000001262784982
30              3.0     1.150968038061982       0.000001262939168
31              3.1     1.1511096485946526      0.000001263058082
32              3.2     1.1512188456521175      0.000001263149787
33              3.3     1.1513030469922174      0.000001263220507
34              3.4     1.1513679734086815      0.000001263275041
35              3.5     1.1514180367134519      0.000001263317094
36              3.6     1.151456639107846       0.000001263349521
```

```
36              3.6     1.151456639107846       0.000001263349521
37              3.7     1.1514864041421093      0.000001263374524
38              3.8     1.1515093548738349      0.000001263393805
39              3.9     1.1515270512829032      0.000001263408671
40              4.0     1.1515406962526111      0.000001263420134

Adams
iterations      x       y                       fault
0               0.0     0       0.000000000000000
1               0.1     0.23050646574222378     0.000000437561509
2               0.2     0.4142588412688023      0.000000662387270
3               0.3     0.562936868021617       0.000000803684329
4               0.4     0.6836172599384118      0.000000845607467
5               0.5     0.7812563593953459      0.000000884746372
6               0.6     0.8598166843052849      0.000000920282543
7               0.7     0.9226266126424083      0.000000951127685
8               0.8     0.9725339410623992      0.000000977186805
9               0.9     1.0119682971175308      0.000000998748837
10              1.0     1.0429779196488338      0.000001016306655
11              1.1     1.0672648741016337      0.000001030428419
12              1.2     1.0862241396935408      0.000001041678542
13              1.3     1.1009852313971573      0.000001050574877
14              1.4     1.112453478495946       0.000001057569677
15              1.5     1.1213485407900263      0.000001063045027
16              1.6     1.1282387047413862      0.000001067316296
17              1.7     1.1335703697451587      0.000001070639437
18              1.8     1.137692742650205       0.000001073219611
19              1.9     1.1408781156203653      0.000001075219753
20              2.0     1.143338271368118       0.000001076768357
21              2.1     1.145237606464519       0.000001077966228
22              2.2     1.1467035395407736      0.000001078892124
23              2.3     1.1478347121959187      0.000001079607395
24              2.4     1.1487074184011021      0.000001080159714
25              2.5     1.149380625736534       0.000001080586061
26              2.6     1.1498998853296931      0.000001080915083
27              2.7     1.1503003695563405      0.000001081168947
28              2.8     1.150609227976539       0.000001081364790
29              2.9     1.1508474120694991      0.000001081515856
30              3.0     1.1510310870813407      0.000001081632371
31              3.1     1.1511727235429543      0.000001081722232
32              3.2     1.15128194062024        0.000001081791532
33              3.3     1.1513661574128848      0.000001081844973
34              3.4     1.1514310957537164      0.000001081886183
35              3.5     1.1514811682585353      0.000001081917961
36              3.6     1.1515197777500594      0.000001081942466
37              3.7     1.1515495482586091      0.000001081961360
38              3.8     1.1515725032125008      0.000001081975930
39              3.9     1.151590202877796       0.000001081987164
40              4.0     1.1516038503586503      0.000001081995826
Press any key to continue . . .
```

3 Розв’язок у Mathcad

Нижче наведено розв’язок у Mathcad

Початкові значення

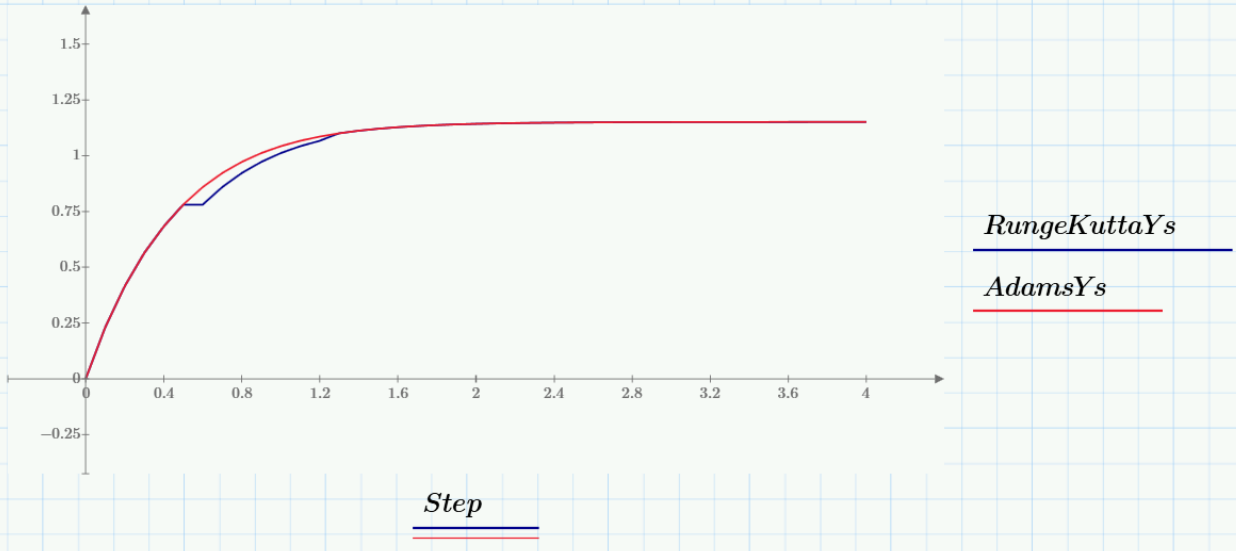
$$variant := 9 \qquad n := variant - 5 = 4 \quad a := 1 + 0.4 \quad n = 2.6 \qquad interval := \begin{bmatrix} 0 \\ 4 \end{bmatrix}$$

$$k := variant - 5 = 4 \qquad b := 1 + 0.4 \quad k = 2.6 \qquad h := 0.1$$

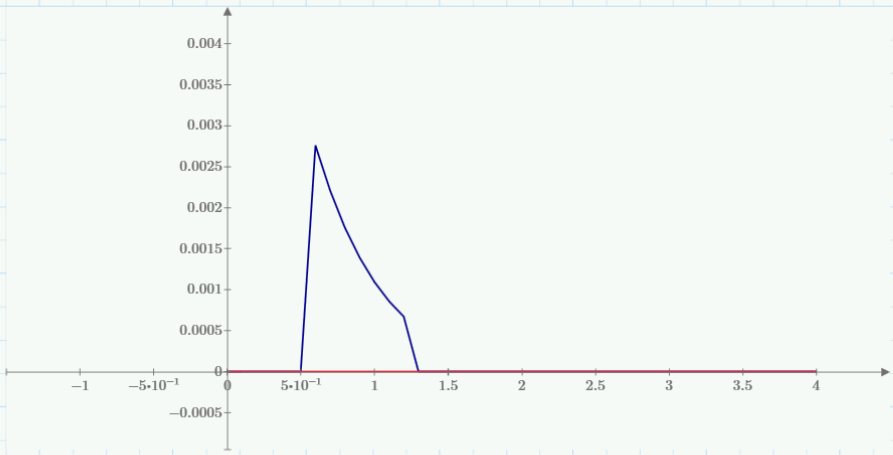
$$RungeKuttaYs := \begin{bmatrix} 0 \\ 0.23050646574222378 \\ 0.4142588412688023 \\ 0.562936868021617 \\ 0.6835931344030579 \\ 0.781219973981651 \\ 0.781219973981651 \\ 0.8597742584224248 \\ 0.9225809661832618 \\ 0.9724862079907614 \\ 1.0119188236949914 \\ \vdots \end{bmatrix} \qquad RungeKuttaErrors := \begin{bmatrix} 0 \\ 0.000000437561509 \\ 0.000000662387270 \\ 0.000000803684329 \\ 0.000000903386957 \\ 0.000000978628499 \\ 0.002761728925931 \\ 0.002211180353480 \\ 0.001759174842239 \\ 0.001391452561454 \\ 0.001095035972226 \\ \vdots \end{bmatrix}$$

$$AdamsYs := \begin{bmatrix} 0 \\ 0.23050646574222378 \\ 0.4142588412688023 \\ 0.562936868021617 \\ 0.6836172599384118 \\ 0.7812563593953459 \\ 0.8598166843052849 \\ 0.9226266126424083 \\ 0.9725339410623992 \\ 1.0119682971175308 \\ 1.0429779196488338 \\ 1.0672648741016337 \\ \vdots \end{bmatrix} \qquad AdamsErrors := \begin{bmatrix} 0 \\ 0.000000437561509 \\ 0.000000662387270 \\ 0.000000803684329 \\ 0.000000845607467 \\ 0.000000884746372 \\ 0.000000920282543 \\ 0.000000951127685 \\ 0.000000977186805 \\ 0.000000998748837 \\ 0.000001016306655 \\ 0.000001030428419 \\ \vdots \end{bmatrix} \qquad Step := \begin{bmatrix} 0 \\ 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \\ 0.5 \\ 0.6 \\ 0.7 \\ 0.8 \\ 0.9 \\ 1 \\ 1.1 \\ \vdots \end{bmatrix}$$

Графік Значень



Графік Похибок



RungeKuttaErrors

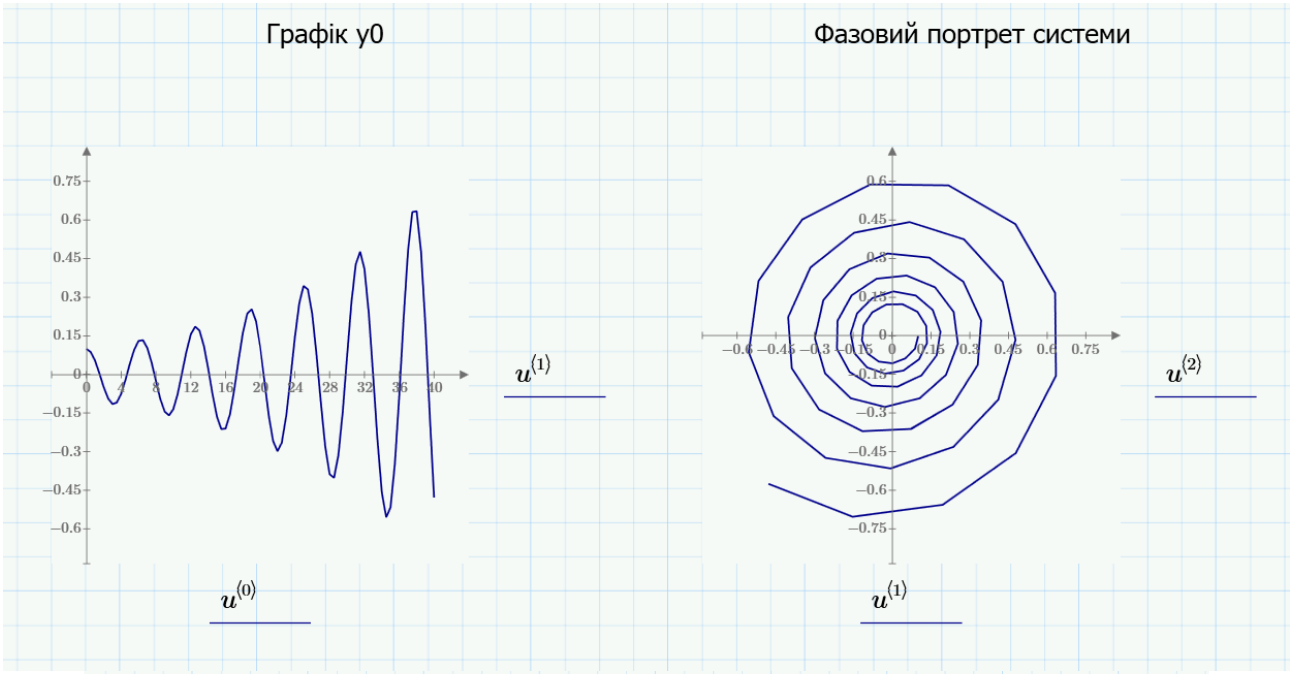
AdamsErrors

Step

ЗДР

$$\text{ORIGIN} := 0 \qquad y0 := \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} \qquad D(x, y) := \begin{bmatrix} y_1 \\ -y_0 - \dfrac{(variant - 10)}{10}\, y_1 \end{bmatrix} \qquad M := 80$$

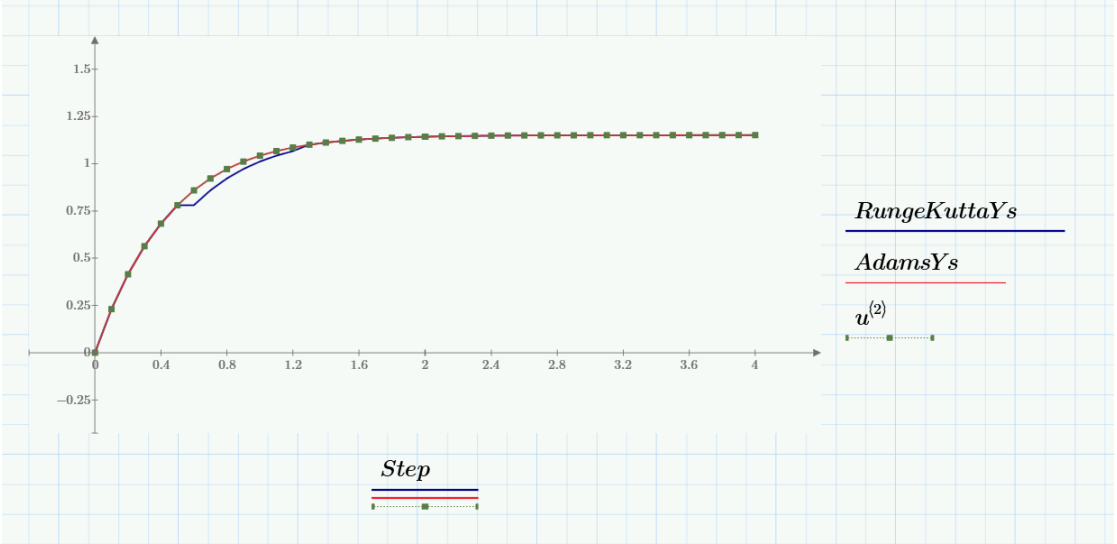
$$u := \text{rkfixed}(y0, 0, 40, M, D) = \begin{bmatrix} 0 & 0.1 & 0 \\ 0.5 & 0.08755 & -0.04914 \\ 1 & 0.05251 & -0.08845 \\ 1.5 & 0.00251 & -0.10758 \\ 2 & -0.05067 & -0.10071 \\ 2.5 & -0.09384 & -0.06822 \\ 3 & -0.11568 & -0.01697 \\ 3.5 & -0.10961 & 0.04115 \\ 4 & -0.07575 & 0.09191 \\ 4.5 & -0.02116 & 0.1222 \\ 5 & 0.04152 & 0.12339 \\ 5.5 & 0.09698 & 0.09369 \\ & & \vdots \end{bmatrix}$$



Наше рівняння

$$y0 := \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} \quad D(x, y) := \begin{bmatrix} y_1 \\ e^{-a \cdot x}\left(y_1^{\,2} + b\right) \end{bmatrix} \quad M := 40$$

$$u := \text{rkfixed}(y0, 0, 4, M, D) \quad u^{\langle 2 \rangle} = \begin{bmatrix} 0 \\ 0.23051 \\ 0.41426 \\ 0.56294 \\ 0.68359 \\ 0.78122 \\ 0.85977 \\ 0.92258 \\ 0.97249 \\ 1.01192 \\ 1.04293 \\ \vdots \end{bmatrix} \quad RungeKuttaYs = \begin{bmatrix} 0 \\ 0.23051 \\ 0.41426 \\ 0.56294 \\ 0.68359 \\ 0.78122 \\ 0.78122 \\ 0.85977 \\ 0.92258 \\ 0.97249 \\ 1.01192 \\ \vdots \end{bmatrix} \quad AdamsYs = \begin{bmatrix} 0 \\ 0.23051 \\ 0.41426 \\ 0.56294 \\ 0.68362 \\ 0.78126 \\ 0.85982 \\ 0.92263 \\ 0.97253 \\ 1.01197 \\ 1.04298 \\ \vdots \end{bmatrix}$$



У технічних розрахунках точність вимірювань характеризують відносною похибкою. Результат вважають гарним, якщо відносна похибка не перевищує 0,1 %. Отже Наш результат є гарним. Також саме у цій лабораторній є умова, що τ не повинно перевищувати декількох сотих, інакше крок потрібно зменшити, яка дотримується

4 Лістинг програми

Lab8.py

```
# region Starting Values
import math

epsilonValue = 0.1
variant = 9
N = K = variant - 5
A = B = 1 + 0.4 * N
h = 0.1
rungeConstants = [0.5, 1, 6, 2]
adamsConstants = [24, 55, 59, 37, 9, 19, 5]
leftBorder = 0
rightBorder = 4
yZero = 0
yFirstAdams = []
ySecondAdams = []
square = half = 2
header = "iterations\t x\t y\t\t\t fault"
rounding = 1
rungeValue = 15
adamsStartValue = 3
numOfIter = 41

# endregion Starting Values

# region Prime Function

def MyPrimeFunction(X, Y):
    result = (Y ** square + B) * math.e ** (-A * X)
    return result

# endregion Prime Function

# region Runge Kutta

def RungeKutta(xInnerZero, yInnerZero, x, h):
    y = yInnerZero
    countToIterate = (int)(x / h)
    for i in range(0, countToIterate):
        K1 = h * MyPrimeFunction(xInnerZero, y)
        K2 = h * MyPrimeFunction(rungeConstants[0] * h + xInnerZero, rungeConstants[0] * K1 + y)
        K3 = h * MyPrimeFunction(rungeConstants[0] * h + xInnerZero, rungeConstants[0] * K2 + y)
        K4 = h * MyPrimeFunction(h + xInnerZero, K3 + y)

        y = (K1 + rungeConstants[3] * K2 + rungeConstants[3] * K3 + K4) * (rungeConstants[1] /
rungeConstants[2]) + y

        errorValue = abs((K2 - K3) / (K1 - K2))
        if epsilonValue < errorValue :
            h = h / half

        xInnerZero = xInnerZero + h
    return y

def RungeKuttaFull():
    tempValueForLeftBorder = leftBorder
    iterations = 0
    yFirstRunge = []
    ySecondRunge = []
    print(header)
```

```
    # do the Runge's rule
    while tempValueForLeftBorder <= rightBorder + 0.1:
        tempOne = RungeKutta(leftBorder, yZero, tempValueForLeftBorder, h)
        yFirstRunge.append(tempOne)
        tempTwo = RungeKutta(leftBorder, yZero, tempValueForLeftBorder, h / half)
        ySecondRunge.append(tempTwo)

        faultValue = abs((yFirstRunge[iterations] - ySecondRunge[iterations]) / (rungeValue))

        print(iterations, "\t\t", round(tempValueForLeftBorder, rounding), "\t", tempOne, "\t", "%-
.15f"%(faultValue))
        tempValueForLeftBorder = tempValueForLeftBorder + 0.1

        # For Adams Method
        if iterations <= adamsStartValue:
            yFirstAdams.append(yFirstRunge[iterations])
            ySecondAdams.append(ySecondRunge[iterations])

        iterations = iterations + 1

# endregion Runge Kutta

# region Adams

def Adams(firstValuesFromRunge, h):
    iterations = adamsStartValue
    while iterations < ((rightBorder - leftBorder) / h) + 4:
        K4 = MyPrimeFunction(iterations * h - 0.3, firstValuesFromRunge[iterations-3])
        K3 = MyPrimeFunction(iterations * h - 0.2, firstValuesFromRunge[iterations-2])
        K2 = MyPrimeFunction(iterations * h - 0.1, firstValuesFromRunge[iterations-1])
        K1 = MyPrimeFunction(iterations * h, firstValuesFromRunge[iterations])

        firstAdditionalY = h / adamsConstants[0] * (adamsConstants[1] * K1 - adamsConstants[2] * K2 +
adamsConstants[3] * K3 - adamsConstants[4] * K4) + firstValuesFromRunge[iterations]
        additionalX = h + h * iterations
        secondAdditionalY = firstValuesFromRunge[iterations] + h / adamsConstants[0] * (adamsConstants[4] *
MyPrimeFunction(additionalX, firstAdditionalY) + adamsConstants[5] * K1 - adamsConstants[6] * K2 + K3)

        errorValue = abs(firstAdditionalY - secondAdditionalY)

        if epsilonValue < errorValue :
            h = h / half

        if firstAdditionalY != secondAdditionalY:
            firstValuesFromRunge.append(secondAdditionalY)
        else:
            firstValuesFromRunge.append(firstAdditionalY)

        iterations = iterations + 1
    return firstValuesFromRunge

def AdamsFull():
    print(header)

    yFirstAdamsErrors = Adams(yFirstAdams, h)
    ySecondAdamsErrors = Adams(ySecondAdams, h)
    # do the Runge's rule
    for x in range(numOfIter):
        faultValue = abs((yFirstAdamsErrors[x] - ySecondAdamsErrors[x]) / (rungeValue))
        print(x, "\t\t", round(x * 0.1, rounding), "\t", yFirstAdamsErrors[x], "\t", "%-.15f"%(faultValue))

# endregion Adams
```

```
def RunAll():
   print("My variant: y' = e^(-ax)*(y^(2)+b), with y(0) =", yZero, ", intervals = [",leftBorder,",",rightBorder,"] and h =", h)
   print("\nRunge Kutta")
   RungeKuttaFull()
   print("\nAdams")
   AdamsFull()

RunAll()
```

Висновок:

Я навчився використовувати різні методи розв'язання задачі Коші (методи Рунге-Кута та Адамса). Метод Рунге-Кутта має різні зручні властивості, які стосуються обчислень, але має також один суттєвий недолік. При побудові цього методу застосовується інформація на відрізку прямої довжиною в один крок, тому подібна інформація має бути отримана знову, що передбачає велику трудоємкість відповідних обчислювальних правил. Якщо відмовитись від умови однокроковості, можна обчислювальні методи будувати таким чином, щоби частина отриманої інформації використовувалась повторно на декількох наступних кроках обчислювального процесу. Такі методи називаються багатокроковими. До них відноситься зокрема метода Адамса (Адамса-Башфорта).

З фазового портрету та графіку y0 можна зрозуміти, що Наша система має нестійкий фокус, а тип коренів – комплексні з додатною дійсною частиною