**Московский авиационный институт**

**(Национальный исследовательский университет)**

**Факультет: «Информационные технологии и прикладная математика»**

**Кафедра: 806 «Вычислительная математика и программирование»**

**Дисциплина: «Численные методы»**

**Лабораторные работы по курсу**

**“Численные методы”**

Студент: Аксенов А. Е.

Группа: М8О-406Б-19

Преподаватель: Ревизников Д. Л.

Дата: 28.11.2022

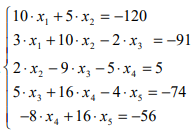
Оценка:

Москва, 2022

# Лабораторная работа 1.2

Задание: Реализовать метод прогонки в виде программы, задавая в качестве входных данных ненулевые элементы матрицы системы и вектор правых частей. Используя разработанное программное обеспечение, решить СЛАУ с трехдиагональной матрицей.

Условие:



Исходный код:

import numpy as np

import copy

def metod\_progonki(matrix):

p = []

q = []

p.append(0)

q.append(0)

n = len(matrix)

m = len(matrix[0])

for i in range(n):

if i == 0:

a = 0

else:

a = matrix[i][i - 1]

if i == (n - 1):

c = 0

else:

c = matrix[i][i + 1]

b = matrix[i][i]

d = matrix[i][m - 1]

p\_i = float(((-1) \* c) / (b + a \* p[i]))

q\_i = float((d - a \* q[i]) / (b + a \* p[i]))

p.append(p\_i)

q.append(q\_i)

print(f'P-{p}')

print(f'Q-{q}')

x = []

x.append(q[n])

for i in range(n - 1, 0, -1):

x\_i = q[i] + p[i] \* x[n - i - 1]

x.append(x\_i)

x.reverse()

return x

def check\_slau(matrix, res):

eps = 0.000001

n = len(matrix)

for i in range(n):

elem = 0

for j in range(n):

elem += matrix[i][j] \* res[j]

if (elem - matrix[i][n] >= eps):

return 'ERR'

return 'OK'

matrix1 = [[10, 5, 0, 0, 0, -120],

[3, 10, -2, 0, 0, -91],

[0, 2, -9, -5, 0, 5],

[0, 0, 5, 16, -4, -74],

[0, 0, 0, -8, 16, -56]]

print('По методу прогонки:')

progonka = np.array(metod\_progonki(matrix1.copy()))

print('x = ', progonka)

print('Проверка прогонки')

print(check\_slau(matrix1, progonka))

Результат:  
По методу прогонки:

P-[0, -0.5, 0.23529411764705882, -0.5862068965517242, 0.30606860158311344, 0.0]

Q-[0, -12.0, -6.470588235294118, -2.103448275862069, -4.857519788918205, -6.999999999999999]

x = [-9. -6. 2. -7. -7.]

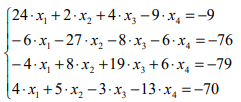
Проверка прогонки

OK

# Лабораторная работа 1.3

Задание: Реализовать метод простых итераций и метод Зейделя в виде программ, задавая в качестве входных данных матрицу системы, вектор правых частей и точность вычислений. Используя разработанное программное обеспечение, решить СЛАУ. Проанализировать количество итераций, необходимое для достижения заданной точности.

Условие:



Исходный код:

N = 4

epsilon\_adj = 0.01

matrix = [] #инициализация списка строк матрицы

matrix = [[24, 2, 4, -9, -9],

[-6, -27, -8, -6, -76],

[-4, 8, 19, 6, -79],

[4, 5, -3, -13, -70]]

print('Матрица')

for i in range(N):

for j in range(N):

print(int(matrix[i][j]), end='\t')

print('', end='\t')

print(int(matrix[i][N]))

for i in range(N): #преобразование исходной системы в матричном виде

divisor = matrix[i][i]

for j in range(N+1):

matrix[i][j] /= -divisor

matrix[i][i] = 0

matrix[i][N] \*= -1

print("\n Преобразованная матрица B, вектор beta:")

for i in range(N): #печать преобразованной системы

for j in range(N):

print(matrix[i][j], end='\t')

print('', end='\t')

print(matrix[i][N])

print("\n Норма матрицы B и вектора beta")

b2 = [] #ициализация вспомогательного вектороа для вычисления 2 нормы матрицы B

for i in range(N): #вычисление нормы. Находится сумма модулей элементов в столбце и заносится в соотв. вектор

b\_i2 = 0

for j in range(N):

b\_i2 += abs(matrix[j][i])

b2.append(b\_i2)

norm\_B2 = max(b2) #норма определяется максимальным элементом в соответствующем векторе.

norm\_B = norm\_B2

norm\_beta = abs(matrix[0][N]) #инициализация переменной для нормы вектора beta

for i in range(1, N):

cur = abs(matrix[i][N])

norm\_beta = (norm\_beta + cur)

print(f'||B||2 = {norm\_B2}\n||beta||2 = {norm\_beta}') #вывод норм

results = [[]] #таблицы для обоих методов

results\_z = [[]]

epsilon = (norm\_B / (1-norm\_B)) \* norm\_beta

L = 0

for i in range(N): #добавление в таблицы начального приближения

results[0].append(matrix[i][N])

results\_z[0].append(matrix[i][N])

while epsilon\_adj < epsilon: #итерационный процесс

L += 1

for k in range(1, L):

results.append([])

for i in range(N):

x\_i = 0

for j in range(N): #вычисление значения x\_i по формуле (на основе значений предыдущего шага)

x\_i += matrix[i][j]\*results[k-1][j]

x\_i += results[0][i]

results[k].append(x\_i)

results[k].append(epsilon)

diff = []

for i in range(N): #рассчёт параметра delta как нормы разности векторов значений на текущем и предыдущем шагах

diff.append(abs(results[k][i] - results[k - 1][i]))

delta = max(diff)

results[k].append(delta)

results\_z.append([])

for i in range(N):

x\_i = 0

for j in range(0, i):

x\_i +=matrix[i][j]\*results\_z[k][j]

for j in range(i, N):

x\_i += matrix[i][j]\*results\_z[k-1][j]

x\_i += results\_z[0][i]

results\_z[k].append(x\_i)

results\_z[k].append(epsilon)

diff = []

for i in range(N):

diff.append(abs(results\_z[k][i] - results\_z[k - 1][i]))

delta = max(diff)

results\_z[k].append(delta)

epsilon \*= norm\_B

print('\nРезультаты:')

print("Метод простых итераций")

print('i', end=' ')

for i in range(1, N+1):

print(f'x\_{i} ', end='\t\t\t')

print(" epsilon", end='\t')

print(" delta")

print(0, end=' ')

for j in range(N):

print(results[0][j], end=' \t')

print()

for i in range(1, L):

print(i, end=' ')

for j in range(N):

print(results[i][j], end='\t')

print(results[i][N], end=' ')

print(results[i][N+1])

print("Метод Зейделя")

print('№', end=' ')

for i in range(1, N+1):

print(f'x\_{i} ', end='\t\t\t')

print(" epsilon", end='\t')

print(" delta")

print(0, end=' ')

for j in range(N):

print(results\_z[0][j], end=' \t')

print()

for i in range(1, L):

print(i, end=' ')

for j in range(N):

print(results\_z[i][j], end='\t')

print(results\_z[i][N], end=' ')

print(results\_z[i][N+1])

print("\nПроверка\nМетод простых итераций.")

for k in range(1, L):

print(f'Шаг {k}:')

for i in range(N):

sum = 0

for j in range(N): #вычисление значения переменной x\_i путём подстановки значений с k-ого приближения

sum += matrix[i][j]\*results[k][j]

sum += results[0][i]

difference = abs(sum - results[k][i])

if difference <= results[k][N]:

print(f'x\_{i+1}: {difference} = |{sum} - {results[k][i]}| <= epsilon = {results[k][N]}. OK.')

else:

print(f'x\_{i+1}: {difference} = |{sum} - {results[k][i]}| > epsilon = {results[k][N]}. ERR.')

print("\nМетод Зейделя.")

for k in range(1, L):

print(f'Шаг {k}:')

for i in range(N):

sum = 0

for j in range(N):

sum += matrix[i][j]\*results\_z[k][j]

sum += results\_z[0][i]

difference = abs(sum - results\_z[k][i])

if difference <= results\_z[k][N]:

print(f'x\_{i+1}: {difference} = |{sum} - {results\_z[k][i]}| <= epsilon = {results\_z[k][N]}. OK.')

else:

print(f'x\_{i+1}: {difference} = |{sum} - {results\_z[k][i]}| > epsilon = {results\_z[k][N]}. ERR.')

Результат:

Матрица

24 2 4 -9 -9

-6 -27 -8 -6 -76

-4 8 19 6 -79

4 5 -3 -13 -70

Преобразованная матрица B, вектор beta:

0 -0.08333333333333333 -0.16666666666666666 0.375 -0.375

-0.2222222222222222 0 -0.2962962962962963 -0.2222222222222222 2.814814814814815

0.21052631578947367 -0.42105263157894735 0 -0.3157894736842105 -4.157894736842105

0.3076923076923077 0.38461538461538464 -0.23076923076923078 0 5.384615384615385

Норма матрицы B и вектора beta

||B||2 = 0.9130116959064327

||beta||2 = 12.732324936272306

Результаты:

Метод простых итераций

i x\_1 x\_2 x\_3 x\_4 epsilon delta

0 -0.375 2.814814814814815 -4.157894736842105 5.384615384615385

1 2.102645324136552 2.933535762483131 -7.122432148747938 7.311366021892338 122.01115308436363 2.964537411905833

2 3.3093729694606884 2.8331625597707473 -7.259257943930151 8.803504350400438 111.39760979705422 1.4921383285080996

3 3.9000935754076274 2.2739555419360906 -7.434149194880308 9.167775269883611 101.70732064073151 0.590720605946939

4 4.112444296858397 2.113554832862779 -7.1893642975347305 9.174815353534761 92.8599733042936 0.2447848973455775

5 4.087653571092758 1.992272462145144 -7.079344926140905 9.1219726341809 84.78224170837917 0.12128237071763492

6 4.059607853662559 1.9769260065735295 -7.016810590414501 9.042308567193807 77.40717828491637 0.0796640669870925

7 4.020590310552301 1.982332822154734 -6.991096212689518 9.013345632212415 70.67365912124309 0.0390175431102584

8 4.004992912348348 1.9898205201825134 -6.992440796226482 8.99748569161934 64.52587737019928 0.015859940593074384

9 3.9986455570464567 1.9972094350372493 -6.993868771672776 8.99587666453734 58.912880727616155 0.007388914854735917

10 3.997664424893861 1.9994006238473867 -6.997808065228221 8.9970950552608 53.78814914385422 0.003939293555444756

11 3.9985952712735546 2.0005150978110295 -6.999321980145708 8.998545001115158 49.10920926949848 0.0015139149174867583

12 3.999298447291549 2.0004346002530884 -7.000053142320262 8.999609424198962 44.837282439768714 0.0010644230838039448

13 3.99982617444023 2.0002584410970012 -7.000207345160646 8.999963555184033 40.936963280168946 0.0005277271486807678

14 3.9999993539627035 2.0001081623533183 -7.000133903269436 9.00009376451753 37.37592626968641 0.0001731795224735322

15 4.000048465376203 2.0000189820471883 -7.000075287898995 9.000072302878902 34.12465782956018 8.918030613003225e-05

16 4.000038079725488 1.9999954701356049 -7.0000206216919 9.000039587341366 31.156211717193464 5.466620709526637e-05

17 4.000018659690362 1.9999888507827435 -7.000002577170056 9.000014733435052 28.445985697934674 2.485390631434825e-05

18 4.0000068836679255 1.999993342911406 -6.999996029900569 9.000002048014256 25.971517643801466 1.2685420795577329e-05

19 4.00000066107949 1.9999968388559801 -6.999996394563425 8.999998641456187 23.712299369231015 6.222588435633725e-06

20 3.999999153068643 1.9999990867145683 -6.999998100803527 8.999998155560625 21.649606660942645 2.2478585881557933e-06

21 3.9999990679096085 2.000000035357504 -6.999999211305565 8.999998949866 19.766344093214446 1.1105020378820996e-06

22 3.9999994718042187 2.000000206807069 -6.999999879495663 8.999999544795589 18.04690334241582 6.681900979188526e-07

23 3.9999997919803674 2.000000182828387 -7.0000000545270105 8.99999988921071 16.477033826518536 3.444151204234913e-07

24 3.999999952306152 2.0000000870025785 -7.000000085787889 9.000000018895726 15.043724597457347 1.603257846483075e-07

25 4.000000014133663 2.0000000318182165 -7.000000052640546 9.000000038584705 13.735096507473848 6.182751111083462e-08

26 4.0000000205911705 2.000000003882006 -7.0000000226062795 9.000000028734414 12.54030375572722 3.003426662928632e-08

27 4.000000014219618 1.999999995736916 -7.000000006373571 9.000000013045659 11.449443999198316 1.6232708510699467e-08

28 4.000000006309641 1.999999995829515 -6.999999999331095 9.000000004206445 10.453476282893783 8.839213805345025e-09

29 4.000000001813473 1.999999997464898 -6.999999998244012 9.000000000183032 9.544146109162526 4.4961678824506635e-09

30 3.9999999999872307 1.9999999990360395 -6.999999998608605 8.999999999177724 8.713917025105259 1.8262422685211277e-09

31 3.9999999995400772 1.9999999997733005 -6.999999999337144 8.999999999304226 7.955908161079289 7.372609189815194e-10

32 3.9999999996474997 2.0000000000604192 -6.9999999997816555 8.999999999618327 7.26383720262283 4.445119827778399e-10

33 3.999999999815447 2.0000000000984555 -6.999999999979122 8.99999999986439 6.631968323154909 2.460627257505621e-10

34 3.9999999999374616 2.0000000000649614 -7.000000000037484 8.999999999976264 6.055064645921404 1.220143985847244e-10

35 3.9999999999919327 2.000000000030278 -7.0000000000330225 9.000000000014392 5.528344841195785 5.447109430178898e-11

36 4.000000000008377 2.000000000008379 -7.000000000018992 9.000000000016785 5.047443499015742 2.1898927116126288e-11

37 4.000000000008762 2.0000000000000355 -7.000000000007065 9.000000000010184 4.608374949028262 1.1927347998152982e-11

38 4.000000000004993 1.9999999999978828 -7.000000000001386 9.000000000004341 4.2075002275850135 5.842437644787424e-12

39 4.000000000002036 1.999999999998336 -6.999999999999428 9.000000000001043 3.841496918314095 3.298694650766265e-12

40 4.000000000000434 1.999999999999146 -6.9999999999992 8.999999999999854 3.507331616209287 1.6013856907193258e-12

41 3.9999999999998828 1.999999999999699 -6.999999999999503 8.999999999999622 3.2022347870214904 5.52891066263328e-13

42 3.9999999999998 1.999999999999963 -6.999999999999778 8.999999999999734 2.9236778135890655 2.753353101070388e-13

43 3.999999999999866 2.000000000000038 -6.999999999999942 8.999999999999874 2.669352038868964 1.6431300764452317e-13

44 3.9999999999999396 2.000000000000041 -7.000000000000004 8.999999999999961 2.4371496319790467 8.704148513061227e-14

45 3.9999999999999822 2.0000000000000235 -7.000000000000018 9.0 2.225146118670928 4.263256414560601e-14

46 4.000000000000001 2.0000000000000093 -7.000000000000014 9.000000000000007 2.03158443144736 1.865174681370263e-14

47 4.000000000000004 2.0000000000000027 -7.000000000000005 9.000000000000007 1.8548603471328602 8.881784197001252e-15

48 4.0000000000000036 1.9999999999999996 -7.000000000000003 9.000000000000004 1.6935091912053672 3.552713678800501e-15

49 4.000000000000002 1.9999999999999991 -7.0 9.000000000000002 1.5461936986955436 2.6645352591003757e-15

50 4.000000000000001 1.9999999999999991 -7.0 9.0 1.4116929310458581 1.7763568394002505e-15

51 4.0 1.9999999999999998 -7.0 9.0 1.2888921570733016 8.881784197001252e-16

52 4.0 2.0 -7.0 9.0 1.1767736141699954 2.220446049250313e-16

53 4.0 2.0 -7.0 9.0 1.0744080731712895 0.0

54 4.0 2.0 -7.0 9.0 0.9809471369816817 0.0

55 4.0 2.0 -7.0 9.0 0.8956162091302049 0.0

56 4.0 2.0 -7.0 9.0 0.8177080739792587 0.0

57 4.0 2.0 -7.0 9.0 0.7465770353801857 0.0

58 4.0 2.0 -7.0 9.0 0.6816335651972601 0.0

59 4.0 2.0 -7.0 9.0 0.6223394173474984 0.0

60 4.0 2.0 -7.0 9.0 0.5682031668618608 0.0

61 4.0 2.0 -7.0 9.0 0.5187761369959533 0.0

62 4.0 2.0 -7.0 9.0 0.47364868063446314 0.0

63 4.0 2.0 -7.0 9.0 0.4324467851699155 0.0

64 4.0 2.0 -7.0 9.0 0.39482897271726936 0.0

65 4.0 2.0 -7.0 9.0 0.36048346997358877 0.0

66 4.0 2.0 -7.0 9.0 0.3291256242668219 0.0

67 4.0 2.0 -7.0 9.0 0.30049554437811443 0.0

68 4.0 2.0 -7.0 9.0 0.27435594658498896 0.0

69 4.0 2.0 -7.0 9.0 0.2504901880735754 0.0

70 4.0 2.0 -7.0 9.0 0.22870047142097638 0.0

71 4.0 2.0 -7.0 9.0 0.2088062052666663 0.0

72 4.0 2.0 -7.0 9.0 0.1906425075863057 0.0

73 4.0 2.0 -7.0 9.0 0.17405883916322792 0.0

74 4.0 2.0 -7.0 9.0 0.15891775593192373 0.0

75 4.0 2.0 -7.0 9.0 0.14509376985305025 0.0

76 4.0 2.0 -7.0 9.0 0.13247230887899106 0.0

77 4.0 2.0 -7.0 9.0 0.12094876739024842 0.0

78 4.0 2.0 -7.0 9.0 0.11042763923276336 0.0

79 4.0 2.0 -7.0 9.0 0.100821726170849 0.0

80 4.0 2.0 -7.0 9.0 0.09205141519546081 0.0

81 4.0 2.0 -7.0 9.0 0.08404401869819485 0.0

82 4.0 2.0 -7.0 9.0 0.07673317204243083 0.0

83 4.0 2.0 -7.0 9.0 0.07005828353873984 0.0

84 4.0 2.0 -7.0 9.0 0.06396403226599859 0.0

85 4.0 2.0 -7.0 9.0 0.058399909576193156 0.0

86 4.0 2.0 -7.0 9.0 0.05331980048294244 0.0

87 4.0 2.0 -7.0 9.0 0.048681601464323904 0.0

88 4.0 2.0 -7.0 9.0 0.04444687151238345 0.0

89 4.0 2.0 -7.0 9.0 0.04058051353725652 0.0

90 4.0 2.0 -7.0 9.0 0.03705048348540453 0.0

91 4.0 2.0 -7.0 9.0 0.03382752476116247 0.0

92 4.0 2.0 -7.0 9.0 0.030884925750505794 0.0

93 4.0 2.0 -7.0 9.0 0.02819829843741355 0.0

94 4.0 2.0 -7.0 9.0 0.025745376278018657 0.0

95 4.0 2.0 -7.0 9.0 0.023505829657343057 0.0

96 4.0 2.0 -7.0 9.0 0.021461097399138505 0.0

97 4.0 2.0 -7.0 9.0 0.01959423293240058 0.0

98 4.0 2.0 -7.0 9.0 0.01788976383959673 0.0

99 4.0 2.0 -7.0 9.0 0.016333563622555784 0.0

100 4.0 2.0 -7.0 9.0 0.014912734623225272 0.0

101 4.0 2.0 -7.0 9.0 0.013615501128953482 0.0

102 4.0 2.0 -7.0 9.0 0.012431111776361768 0.0

103 4.0 2.0 -7.0 9.0 0.011349750444938486 0.0

104 4.0 2.0 -7.0 9.0 0.010362454901848075 0.0

Метод Зейделя

№ x\_1 x\_2 x\_3 x\_4 epsilon delta

0 -0.375 2.814814814814815 -4.157894736842105 5.384615384615385

1 2.102645324136552 2.382947912675008 -6.41898391118664 8.429405584113937 122.01115308436363 3.044790199498552

2 3.657278753184249 2.0308061209520005 -6.9049351294506645 8.884457769680692 111.39760979705422 1.5546334290476969

3 3.938260341792703 2.0112286061764797 -6.98123863685922 8.980992331433145 101.70732064073151 0.2809815886084541

4 3.98880951324926 2.0011517787696054 -6.996838429776772 8.996270172013492 92.8599733042936 0.050549171456556685

5 3.9979784045703877 2.0003413325448482 -6.999391477587273 8.99936882413597 84.78224170837917 0.009168891321127859

6 3.9996334442701302 2.0000414151948736 -6.999895288910226 8.999878978445045 77.40717828491637 0.0016550396997425665

7 3.9999337138023563 2.0000105984369405 -6.999980200155598 8.999979111373918 70.67365912124309 0.0003002695322260607

8 3.999987983588074 2.000001445610105 -6.999996542040634 8.999996060655748 64.52587737019928 5.426978571776431e-05

9 3.9999978259518363 2.000000333951095 -6.99999935430189 8.999999310497577 58.912880727616155 9.842363762224693e-06

10 3.9999996059909817 2.000000049462362 -6.999999886037918 8.9999998714915 53.78814914385422 1.7800391454159126e-06

11 3.999999928693769 2.0000000106367306 -6.999999978908829 8.99999997728348 49.10920926949848 3.2270278715884615e-07

12 3.9999999870797156 2.0000000016700543 -6.999999996249603 8.999999995801382 44.837282439768714 5.838594674401065e-08

13 3.9999999976612806 2.0000000003415126 -6.999999999310278 8.99999999925258 40.936963280168946 1.0581564957590217e-08

14 3.9999999995763043 2.0000000000558855 -6.999999999876702 8.999999999862673 37.37592626968641 1.9150236951759325e-09

15 3.999999999923296 2.0000000000110303 -6.999999999977426 8.999999999975433 34.12465782956018 3.469917686516055e-10

16 3.9999999999861053 2.000000000001859 -6.99999999999595 8.999999999995506 31.156211717193464 6.280931330593376e-11

17 3.9999999999974847 2.000000000000358 -6.999999999999261 8.999999999999194 28.445985697934674 1.1379341913198004e-11

18 3.9999999999995444 2.0000000000000617 -6.999999999999867 8.999999999999854 25.971517643801466 2.0596857552845904e-12

19 3.9999999999999183 2.000000000000011 -6.999999999999975 8.999999999999973 23.712299369231015 3.739231146937527e-13

20 3.999999999999985 2.0000000000000018 -6.999999999999996 8.999999999999996 21.649606660942645 6.661338147750939e-14

21 3.9999999999999982 2.0 -6.999999999999999 9.0 19.766344093214446 1.3322676295501878e-14

22 4.0 1.9999999999999996 -7.0 9.0 18.04690334241582 1.7763568394002505e-15

23 4.0 2.0 -7.0 9.0 16.477033826518536 4.440892098500626e-16

24 4.0 2.0 -7.0 9.0 15.043724597457347 0.0

25 4.0 2.0 -7.0 9.0 13.735096507473848 0.0

26 4.0 2.0 -7.0 9.0 12.54030375572722 0.0

27 4.0 2.0 -7.0 9.0 11.449443999198316 0.0

28 4.0 2.0 -7.0 9.0 10.453476282893783 0.0

29 4.0 2.0 -7.0 9.0 9.544146109162526 0.0

30 4.0 2.0 -7.0 9.0 8.713917025105259 0.0

31 4.0 2.0 -7.0 9.0 7.955908161079289 0.0

32 4.0 2.0 -7.0 9.0 7.26383720262283 0.0

33 4.0 2.0 -7.0 9.0 6.631968323154909 0.0

34 4.0 2.0 -7.0 9.0 6.055064645921404 0.0

35 4.0 2.0 -7.0 9.0 5.528344841195785 0.0

36 4.0 2.0 -7.0 9.0 5.047443499015742 0.0

37 4.0 2.0 -7.0 9.0 4.608374949028262 0.0

38 4.0 2.0 -7.0 9.0 4.2075002275850135 0.0

39 4.0 2.0 -7.0 9.0 3.841496918314095 0.0

40 4.0 2.0 -7.0 9.0 3.507331616209287 0.0

41 4.0 2.0 -7.0 9.0 3.2022347870214904 0.0

42 4.0 2.0 -7.0 9.0 2.9236778135890655 0.0

43 4.0 2.0 -7.0 9.0 2.669352038868964 0.0

44 4.0 2.0 -7.0 9.0 2.4371496319790467 0.0

45 4.0 2.0 -7.0 9.0 2.225146118670928 0.0

46 4.0 2.0 -7.0 9.0 2.03158443144736 0.0

47 4.0 2.0 -7.0 9.0 1.8548603471328602 0.0

48 4.0 2.0 -7.0 9.0 1.6935091912053672 0.0

49 4.0 2.0 -7.0 9.0 1.5461936986955436 0.0

50 4.0 2.0 -7.0 9.0 1.4116929310458581 0.0

51 4.0 2.0 -7.0 9.0 1.2888921570733016 0.0

52 4.0 2.0 -7.0 9.0 1.1767736141699954 0.0

53 4.0 2.0 -7.0 9.0 1.0744080731712895 0.0

54 4.0 2.0 -7.0 9.0 0.9809471369816817 0.0

55 4.0 2.0 -7.0 9.0 0.8956162091302049 0.0

56 4.0 2.0 -7.0 9.0 0.8177080739792587 0.0

57 4.0 2.0 -7.0 9.0 0.7465770353801857 0.0

58 4.0 2.0 -7.0 9.0 0.6816335651972601 0.0

59 4.0 2.0 -7.0 9.0 0.6223394173474984 0.0

60 4.0 2.0 -7.0 9.0 0.5682031668618608 0.0

61 4.0 2.0 -7.0 9.0 0.5187761369959533 0.0

62 4.0 2.0 -7.0 9.0 0.47364868063446314 0.0

63 4.0 2.0 -7.0 9.0 0.4324467851699155 0.0

64 4.0 2.0 -7.0 9.0 0.39482897271726936 0.0

65 4.0 2.0 -7.0 9.0 0.36048346997358877 0.0

66 4.0 2.0 -7.0 9.0 0.3291256242668219 0.0

67 4.0 2.0 -7.0 9.0 0.30049554437811443 0.0

68 4.0 2.0 -7.0 9.0 0.27435594658498896 0.0

69 4.0 2.0 -7.0 9.0 0.2504901880735754 0.0

70 4.0 2.0 -7.0 9.0 0.22870047142097638 0.0

71 4.0 2.0 -7.0 9.0 0.2088062052666663 0.0

72 4.0 2.0 -7.0 9.0 0.1906425075863057 0.0

73 4.0 2.0 -7.0 9.0 0.17405883916322792 0.0

74 4.0 2.0 -7.0 9.0 0.15891775593192373 0.0

75 4.0 2.0 -7.0 9.0 0.14509376985305025 0.0

76 4.0 2.0 -7.0 9.0 0.13247230887899106 0.0

77 4.0 2.0 -7.0 9.0 0.12094876739024842 0.0

78 4.0 2.0 -7.0 9.0 0.11042763923276336 0.0

79 4.0 2.0 -7.0 9.0 0.100821726170849 0.0

80 4.0 2.0 -7.0 9.0 0.09205141519546081 0.0

81 4.0 2.0 -7.0 9.0 0.08404401869819485 0.0

82 4.0 2.0 -7.0 9.0 0.07673317204243083 0.0

83 4.0 2.0 -7.0 9.0 0.07005828353873984 0.0

84 4.0 2.0 -7.0 9.0 0.06396403226599859 0.0

85 4.0 2.0 -7.0 9.0 0.058399909576193156 0.0

86 4.0 2.0 -7.0 9.0 0.05331980048294244 0.0

87 4.0 2.0 -7.0 9.0 0.048681601464323904 0.0

88 4.0 2.0 -7.0 9.0 0.04444687151238345 0.0

89 4.0 2.0 -7.0 9.0 0.04058051353725652 0.0

90 4.0 2.0 -7.0 9.0 0.03705048348540453 0.0

91 4.0 2.0 -7.0 9.0 0.03382752476116247 0.0

92 4.0 2.0 -7.0 9.0 0.030884925750505794 0.0

93 4.0 2.0 -7.0 9.0 0.02819829843741355 0.0

94 4.0 2.0 -7.0 9.0 0.025745376278018657 0.0

95 4.0 2.0 -7.0 9.0 0.023505829657343057 0.0

96 4.0 2.0 -7.0 9.0 0.021461097399138505 0.0

97 4.0 2.0 -7.0 9.0 0.01959423293240058 0.0

98 4.0 2.0 -7.0 9.0 0.01788976383959673 0.0

99 4.0 2.0 -7.0 9.0 0.016333563622555784 0.0

100 4.0 2.0 -7.0 9.0 0.014912734623225272 0.0

101 4.0 2.0 -7.0 9.0 0.013615501128953482 0.0

102 4.0 2.0 -7.0 9.0 0.012431111776361768 0.0

103 4.0 2.0 -7.0 9.0 0.011349750444938486 0.0

104 4.0 2.0 -7.0 9.0 0.010362454901848075 0.0

Проверка

Метод простых итераций.

Шаг 1:

x\_1: 1.2067276453241362 = |3.3093729694606884 - 2.102645324136552| <= epsilon = 122.01115308436363. OK.

x\_2: 0.10037320271238359 = |2.8331625597707473 - 2.933535762483131| <= epsilon = 122.01115308436363. OK.

x\_3: 0.1368257951822125 = |-7.259257943930151 - -7.122432148747938| <= epsilon = 122.01115308436363. OK.

x\_4: 1.4921383285080996 = |8.803504350400438 - 7.311366021892338| <= epsilon = 122.01115308436363. OK.

Шаг 2:

x\_1: 0.590720605946939 = |3.9000935754076274 - 3.3093729694606884| <= epsilon = 111.39760979705422. OK.

x\_2: 0.5592070178346567 = |2.2739555419360906 - 2.8331625597707473| <= epsilon = 111.39760979705422. OK.

x\_3: 0.17489125095015723 = |-7.434149194880308 - -7.259257943930151| <= epsilon = 111.39760979705422. OK.

x\_4: 0.36427091948317347 = |9.167775269883611 - 8.803504350400438| <= epsilon = 111.39760979705422. OK.

Шаг 3:

x\_1: 0.21235072145077005 = |4.112444296858397 - 3.9000935754076274| <= epsilon = 101.70732064073151. OK.

x\_2: 0.16040070907331172 = |2.113554832862779 - 2.2739555419360906| <= epsilon = 101.70732064073151. OK.

x\_3: 0.2447848973455775 = |-7.1893642975347305 - -7.434149194880308| <= epsilon = 101.70732064073151. OK.

x\_4: 0.007040083651149942 = |9.174815353534761 - 9.167775269883611| <= epsilon = 101.70732064073151. OK.

Шаг 4:

x\_1: 0.02479072576563901 = |4.087653571092758 - 4.112444296858397| <= epsilon = 92.8599733042936. OK.

x\_2: 0.12128237071763492 = |1.992272462145144 - 2.113554832862779| <= epsilon = 92.8599733042936. OK.

x\_3: 0.11001937139382534 = |-7.079344926140905 - -7.1893642975347305| <= epsilon = 92.8599733042936. OK.

x\_4: 0.05284271935386187 = |9.1219726341809 - 9.174815353534761| <= epsilon = 92.8599733042936. OK.

Шаг 5:

x\_1: 0.028045717430199346 = |4.059607853662559 - 4.087653571092758| <= epsilon = 84.78224170837917. OK.

x\_2: 0.015346455571614515 = |1.9769260065735295 - 1.992272462145144| <= epsilon = 84.78224170837917. OK.

x\_3: 0.06253433572640432 = |-7.016810590414501 - -7.079344926140905| <= epsilon = 84.78224170837917. OK.

x\_4: 0.0796640669870925 = |9.042308567193807 - 9.1219726341809| <= epsilon = 84.78224170837917. OK.

Шаг 6:

x\_1: 0.0390175431102584 = |4.020590310552301 - 4.059607853662559| <= epsilon = 77.40717828491637. OK.

x\_2: 0.005406815581204594 = |1.982332822154734 - 1.9769260065735295| <= epsilon = 77.40717828491637. OK.

x\_3: 0.02571437772498264 = |-6.991096212689518 - -7.016810590414501| <= epsilon = 77.40717828491637. OK.

x\_4: 0.028962934981391797 = |9.013345632212415 - 9.042308567193807| <= epsilon = 77.40717828491637. OK.

Шаг 7:

x\_1: 0.015597398203953006 = |4.004992912348348 - 4.020590310552301| <= epsilon = 70.67365912124309. OK.

x\_2: 0.007487698027779377 = |1.9898205201825134 - 1.982332822154734| <= epsilon = 70.67365912124309. OK.

x\_3: 0.0013445835369640946 = |-6.992440796226482 - -6.991096212689518| <= epsilon = 70.67365912124309. OK.

x\_4: 0.015859940593074384 = |8.99748569161934 - 9.013345632212415| <= epsilon = 70.67365912124309. OK.

Шаг 8:

x\_1: 0.0063473553018909 = |3.9986455570464567 - 4.004992912348348| <= epsilon = 64.52587737019928. OK.

x\_2: 0.007388914854735917 = |1.9972094350372493 - 1.9898205201825134| <= epsilon = 64.52587737019928. OK.

x\_3: 0.0014279754462940275 = |-6.993868771672776 - -6.992440796226482| <= epsilon = 64.52587737019928. OK.

x\_4: 0.0016090270820008357 = |8.99587666453734 - 8.99748569161934| <= epsilon = 64.52587737019928. OK.

Шаг 9:

x\_1: 0.000981132152595876 = |3.997664424893861 - 3.9986455570464567| <= epsilon = 58.912880727616155. OK.

x\_2: 0.0021911888101373567 = |1.9994006238473867 - 1.9972094350372493| <= epsilon = 58.912880727616155. OK.

x\_3: 0.003939293555444756 = |-6.997808065228221 - -6.993868771672776| <= epsilon = 58.912880727616155. OK.

x\_4: 0.0012183907234604163 = |8.9970950552608 - 8.99587666453734| <= epsilon = 58.912880727616155. OK.

Шаг 10:

x\_1: 0.0009308463796937616 = |3.9985952712735546 - 3.997664424893861| <= epsilon = 53.78814914385422. OK.

x\_2: 0.001114473963642837 = |2.0005150978110295 - 1.9994006238473867| <= epsilon = 53.78814914385422. OK.

x\_3: 0.0015139149174867583 = |-6.999321980145708 - -6.997808065228221| <= epsilon = 53.78814914385422. OK.

x\_4: 0.0014499458543575372 = |8.998545001115158 - 8.9970950552608| <= epsilon = 53.78814914385422. OK.

Шаг 11:

x\_1: 0.0007031760179945223 = |3.999298447291549 - 3.9985952712735546| <= epsilon = 49.10920926949848. OK.

x\_2: 8.049755794115043e-05 = |2.0004346002530884 - 2.0005150978110295| <= epsilon = 49.10920926949848. OK.

x\_3: 0.0007311621745538233 = |-7.000053142320262 - -6.999321980145708| <= epsilon = 49.10920926949848. OK.

x\_4: 0.0010644230838039448 = |8.999609424198962 - 8.998545001115158| <= epsilon = 49.10920926949848. OK.

Шаг 12:

x\_1: 0.0005277271486807678 = |3.99982617444023 - 3.999298447291549| <= epsilon = 44.837282439768714. OK.

x\_2: 0.00017615915608715227 = |2.0002584410970012 - 2.0004346002530884| <= epsilon = 44.837282439768714. OK.

x\_3: 0.00015420284038469845 = |-7.000207345160646 - -7.000053142320262| <= epsilon = 44.837282439768714. OK.

x\_4: 0.0003541309850714214 = |8.999963555184033 - 8.999609424198962| <= epsilon = 44.837282439768714. OK.

Шаг 13:

x\_1: 0.0001731795224735322 = |3.9999993539627035 - 3.99982617444023| <= epsilon = 40.936963280168946. OK.

x\_2: 0.00015027874368289673 = |2.0001081623533183 - 2.0002584410970012| <= epsilon = 40.936963280168946. OK.

x\_3: 7.344189121027966e-05 = |-7.000133903269436 - -7.000207345160646| <= epsilon = 40.936963280168946. OK.

x\_4: 0.00013020933349672248 = |9.00009376451753 - 8.999963555184033| <= epsilon = 40.936963280168946. OK.

Шаг 14:

x\_1: 4.911141349950299e-05 = |4.000048465376203 - 3.9999993539627035| <= epsilon = 37.37592626968641. OK.

x\_2: 8.918030613003225e-05 = |2.0000189820471883 - 2.0001081623533183| <= epsilon = 37.37592626968641. OK.

x\_3: 5.8615370440939785e-05 = |-7.000075287898995 - -7.000133903269436| <= epsilon = 37.37592626968641. OK.

x\_4: 2.1461638628039736e-05 = |9.000072302878902 - 9.00009376451753| <= epsilon = 37.37592626968641. OK.

Шаг 15:

x\_1: 1.0385650714539452e-05 = |4.000038079725488 - 4.000048465376203| <= epsilon = 34.12465782956018. OK.

x\_2: 2.351191158345145e-05 = |1.9999954701356049 - 2.0000189820471883| <= epsilon = 34.12465782956018. OK.

x\_3: 5.466620709526637e-05 = |-7.0000206216919 - -7.000075287898995| <= epsilon = 34.12465782956018. OK.

x\_4: 3.271553753592116e-05 = |9.000039587341366 - 9.000072302878902| <= epsilon = 34.12465782956018. OK.

Шаг 16:

x\_1: 1.9420035126671564e-05 = |4.000018659690362 - 4.000038079725488| <= epsilon = 31.156211717193464. OK.

x\_2: 6.619352861392258e-06 = |1.9999888507827435 - 1.9999954701356049| <= epsilon = 31.156211717193464. OK.

x\_3: 1.8044521843840755e-05 = |-7.000002577170056 - -7.0000206216919| <= epsilon = 31.156211717193464. OK.

x\_4: 2.485390631434825e-05 = |9.000014733435052 - 9.000039587341366| <= epsilon = 31.156211717193464. OK.

Шаг 17:

x\_1: 1.1776022436293943e-05 = |4.0000068836679255 - 4.000018659690362| <= epsilon = 28.445985697934674. OK.

x\_2: 4.492128662603179e-06 = |1.999993342911406 - 1.9999888507827435| <= epsilon = 28.445985697934674. OK.

x\_3: 6.547269487455765e-06 = |-6.999996029900569 - -7.000002577170056| <= epsilon = 28.445985697934674. OK.

x\_4: 1.2685420795577329e-05 = |9.000002048014256 - 9.000014733435052| <= epsilon = 28.445985697934674. OK.

Шаг 18:

x\_1: 6.222588435633725e-06 = |4.00000066107949 - 4.0000068836679255| <= epsilon = 25.971517643801466. OK.

x\_2: 3.495944574050114e-06 = |1.9999968388559801 - 1.999993342911406| <= epsilon = 25.971517643801466. OK.

x\_3: 3.646628563558352e-07 = |-6.999996394563425 - -6.999996029900569| <= epsilon = 25.971517643801466. OK.

x\_4: 3.4065580685194163e-06 = |8.999998641456187 - 9.000002048014256| <= epsilon = 25.971517643801466. OK.

Шаг 19:

x\_1: 1.508010846862362e-06 = |3.999999153068643 - 4.00000066107949| <= epsilon = 23.712299369231015. OK.

x\_2: 2.2478585881557933e-06 = |1.9999990867145683 - 1.9999968388559801| <= epsilon = 23.712299369231015. OK.

x\_3: 1.7062401020950801e-06 = |-6.999998100803527 - -6.999996394563425| <= epsilon = 23.712299369231015. OK.

x\_4: 4.858955620079541e-07 = |8.999998155560625 - 8.999998641456187| <= epsilon = 23.712299369231015. OK.

Шаг 20:

x\_1: 8.515903449080042e-08 = |3.9999990679096085 - 3.999999153068643| <= epsilon = 21.649606660942645. OK.

x\_2: 9.486429357608017e-07 = |2.000000035357504 - 1.9999990867145683| <= epsilon = 21.649606660942645. OK.

x\_3: 1.1105020378820996e-06 = |-6.999999211305565 - -6.999998100803527| <= epsilon = 21.649606660942645. OK.

x\_4: 7.943053752512697e-07 = |8.999998949866 - 8.999998155560625| <= epsilon = 21.649606660942645. OK.

Шаг 21:

x\_1: 4.0389461020140516e-07 = |3.9999994718042187 - 3.9999990679096085| <= epsilon = 19.766344093214446. OK.

x\_2: 1.7144956476755624e-07 = |2.000000206807069 - 2.000000035357504| <= epsilon = 19.766344093214446. OK.

x\_3: 6.681900979188526e-07 = |-6.999999879495663 - -6.999999211305565| <= epsilon = 19.766344093214446. OK.

x\_4: 5.949295882601291e-07 = |8.999999544795589 - 8.999998949866| <= epsilon = 19.766344093214446. OK.

Шаг 22:

x\_1: 3.201761487048316e-07 = |3.9999997919803674 - 3.9999994718042187| <= epsilon = 18.04690334241582. OK.

x\_2: 2.3978681706893212e-08 = |2.000000182828387 - 2.000000206807069| <= epsilon = 18.04690334241582. OK.

x\_3: 1.750313476378551e-07 = |-7.0000000545270105 - -6.999999879495663| <= epsilon = 18.04690334241582. OK.

x\_4: 3.444151204234913e-07 = |8.99999988921071 - 8.999999544795589| <= epsilon = 18.04690334241582. OK.

Шаг 23:

x\_1: 1.603257846483075e-07 = |3.999999952306152 - 3.9999997919803674| <= epsilon = 16.477033826518536. OK.

x\_2: 9.582580862144141e-08 = |2.0000000870025785 - 2.000000182828387| <= epsilon = 16.477033826518536. OK.

x\_3: 3.126087833038582e-08 = |-7.000000085787889 - -7.0000000545270105| <= epsilon = 16.477033826518536. OK.

x\_4: 1.2968501650334474e-07 = |9.000000018895726 - 8.99999988921071| <= epsilon = 16.477033826518536. OK.

Шаг 24:

x\_1: 6.182751111083462e-08 = |4.000000014133663 - 3.999999952306152| <= epsilon = 15.043724597457347. OK.

x\_2: 5.518436196894072e-08 = |2.0000000318182165 - 2.0000000870025785| <= epsilon = 15.043724597457347. OK.

x\_3: 3.31473426484763e-08 = |-7.000000052640546 - -7.000000085787889| <= epsilon = 15.043724597457347. OK.

x\_4: 1.968897933579683e-08 = |9.000000038584705 - 9.000000018895726| <= epsilon = 15.043724597457347. OK.

Шаг 25:

x\_1: 6.457507417678698e-09 = |4.0000000205911705 - 4.000000014133663| <= epsilon = 13.735096507473848. OK.

x\_2: 2.7936210678092266e-08 = |2.000000003882006 - 2.0000000318182165| <= epsilon = 13.735096507473848. OK.

x\_3: 3.003426662928632e-08 = |-7.0000000226062795 - -7.000000052640546| <= epsilon = 13.735096507473848. OK.

x\_4: 9.850291249335896e-09 = |9.000000028734414 - 9.000000038584705| <= epsilon = 13.735096507473848. OK.

Шаг 26:

x\_1: 6.371552174755379e-09 = |4.000000014219618 - 4.0000000205911705| <= epsilon = 12.54030375572722. OK.

x\_2: 8.145089935851502e-09 = |1.999999995736916 - 2.000000003882006| <= epsilon = 12.54030375572722. OK.

x\_3: 1.6232708510699467e-08 = |-7.000000006373571 - -7.0000000226062795| <= epsilon = 12.54030375572722. OK.

x\_4: 1.5688755183873582e-08 = |9.000000013045659 - 9.000000028734414| <= epsilon = 12.54030375572722. OK.

Шаг 27:

x\_1: 7.909977561837422e-09 = |4.000000006309641 - 4.000000014219618| <= epsilon = 11.449443999198316. OK.

x\_2: 9.259903954728088e-11 = |1.999999995829515 - 1.999999995736916| <= epsilon = 11.449443999198316. OK.

x\_3: 7.042475935747916e-09 = |-6.999999999331095 - -7.000000006373571| <= epsilon = 11.449443999198316. OK.

x\_4: 8.839213805345025e-09 = |9.000000004206445 - 9.000000013045659| <= epsilon = 11.449443999198316. OK.

Шаг 28:

x\_1: 4.4961678824506635e-09 = |4.000000001813473 - 4.000000006309641| <= epsilon = 10.453476282893783. OK.

x\_2: 1.6353831622240023e-09 = |1.999999997464898 - 1.999999995829515| <= epsilon = 10.453476282893783. OK.

x\_3: 1.0870833122567092e-09 = |-6.999999998244012 - -6.999999999331095| <= epsilon = 10.453476282893783. OK.

x\_4: 4.023412714104779e-09 = |9.000000000183032 - 9.000000004206445| <= epsilon = 10.453476282893783. OK.

Шаг 29:

x\_1: 1.8262422685211277e-09 = |3.9999999999872307 - 4.000000001813473| <= epsilon = 9.544146109162526. OK.

x\_2: 1.5711414391716971e-09 = |1.9999999990360395 - 1.999999997464898| <= epsilon = 9.544146109162526. OK.

x\_3: 3.645928003948029e-10 = |-6.999999998608605 - -6.999999998244012| <= epsilon = 9.544146109162526. OK.

x\_4: 1.005307836976499e-09 = |8.999999999177724 - 9.000000000183032| <= epsilon = 9.544146109162526. OK.

Шаг 30:

x\_1: 4.4715342539802805e-10 = |3.9999999995400772 - 3.9999999999872307| <= epsilon = 8.713917025105259. OK.

x\_2: 7.372609189815194e-10 = |1.9999999997733005 - 1.9999999990360395| <= epsilon = 8.713917025105259. OK.

x\_3: 7.285390069000641e-10 = |-6.999999999337144 - -6.999999998608605| <= epsilon = 8.713917025105259. OK.

x\_4: 1.2650147596104944e-10 = |8.999999999304226 - 8.999999999177724| <= epsilon = 8.713917025105259. OK.

Шаг 31:

x\_1: 1.0742251532747105e-10 = |3.9999999996474997 - 3.9999999995400772| <= epsilon = 7.955908161079289. OK.

x\_2: 2.871187732012004e-10 = |2.0000000000604192 - 1.9999999997733005| <= epsilon = 7.955908161079289. OK.

x\_3: 4.445119827778399e-10 = |-6.9999999997816555 - -6.999999999337144| <= epsilon = 7.955908161079289. OK.

x\_4: 3.141007454132705e-10 = |8.999999999618327 - 8.999999999304226| <= epsilon = 7.955908161079289. OK.

Шаг 32:

x\_1: 1.6794743373793608e-10 = |3.999999999815447 - 3.9999999996474997| <= epsilon = 7.26383720262283. OK.

x\_2: 3.803624082365786e-11 = |2.0000000000984555 - 2.0000000000604192| <= epsilon = 7.26383720262283. OK.

x\_3: 1.9746604351666974e-10 = |-6.999999999979122 - -6.9999999997816555| <= epsilon = 7.26383720262283. OK.

x\_4: 2.460627257505621e-10 = |8.99999999986439 - 8.999999999618327| <= epsilon = 7.26383720262283. OK.

Шаг 33:

x\_1: 1.220143985847244e-10 = |3.9999999999374616 - 3.999999999815447| <= epsilon = 6.631968323154909. OK.

x\_2: 3.349409638531142e-11 = |2.0000000000649614 - 2.0000000000984555| <= epsilon = 6.631968323154909. OK.

x\_3: 5.836220395849523e-11 = |-7.000000000037484 - -6.999999999979122| <= epsilon = 6.631968323154909. OK.

x\_4: 1.1187495374542777e-10 = |8.999999999976264 - 8.99999999986439| <= epsilon = 6.631968323154909. OK.

Шаг 34:

x\_1: 5.447109430178898e-11 = |3.9999999999919327 - 3.9999999999374616| <= epsilon = 6.055064645921404. OK.

x\_2: 3.468336728928989e-11 = |2.000000000030278 - 2.0000000000649614| <= epsilon = 6.055064645921404. OK.

x\_3: 4.461320202153729e-12 = |-7.0000000000330225 - -7.000000000037484| <= epsilon = 6.055064645921404. OK.

x\_4: 3.8127723200886976e-11 = |9.000000000014392 - 8.999999999976264| <= epsilon = 6.055064645921404. OK.

Шаг 35:

x\_1: 1.644462344074782e-11 = |4.000000000008377 - 3.9999999999919327| <= epsilon = 5.528344841195785. OK.

x\_2: 2.1898927116126288e-11 = |2.000000000008379 - 2.000000000030278| <= epsilon = 5.528344841195785. OK.

x\_3: 1.4030554496002878e-11 = |-7.000000000018992 - -7.0000000000330225| <= epsilon = 5.528344841195785. OK.

x\_4: 2.3927526626721374e-12 = |9.000000000016785 - 9.000000000014392| <= epsilon = 5.528344841195785. OK.

Шаг 36:

x\_1: 3.8458125573015423e-13 = |4.000000000008762 - 4.000000000008377| <= epsilon = 5.047443499015742. OK.

x\_2: 8.343548074662976e-12 = |2.0000000000000355 - 2.000000000008379| <= epsilon = 5.047443499015742. OK.

x\_3: 1.1927347998152982e-11 = |-7.000000000007065 - -7.000000000018992| <= epsilon = 5.047443499015742. OK.

x\_4: 6.600942015211331e-12 = |9.000000000010184 - 9.000000000016785| <= epsilon = 5.047443499015742. OK.

Шаг 37:

x\_1: 3.768541034787631e-12 = |4.000000000004993 - 4.000000000008762| <= epsilon = 4.608374949028262. OK.

x\_2: 2.1527224447481785e-12 = |1.9999999999978828 - 2.0000000000000355| <= epsilon = 4.608374949028262. OK.

x\_3: 5.679012815562601e-12 = |-7.000000000001386 - -7.000000000007065| <= epsilon = 4.608374949028262. OK.

x\_4: 5.842437644787424e-12 = |9.000000000004341 - 9.000000000010184| <= epsilon = 4.608374949028262. OK.

Шаг 38:

x\_1: 2.957634137601417e-12 = |4.000000000002036 - 4.000000000004993| <= epsilon = 4.2075002275850135. OK.

x\_2: 4.531930386519889e-13 = |1.999999999998336 - 1.9999999999978828| <= epsilon = 4.2075002275850135. OK.

x\_3: 1.957545237019076e-12 = |-6.999999999999428 - -7.000000000001386| <= epsilon = 4.2075002275850135. OK.

x\_4: 3.298694650766265e-12 = |9.000000000001043 - 9.000000000004341| <= epsilon = 4.2075002275850135. OK.

Шаг 39:

x\_1: 1.6013856907193258e-12 = |4.000000000000434 - 4.000000000002036| <= epsilon = 3.841496918314095. OK.

x\_2: 8.100187187665142e-13 = |1.999999999999146 - 1.999999999998336| <= epsilon = 3.841496918314095. OK.

x\_3: 2.282618538629322e-13 = |-6.9999999999992 - -6.999999999999428| <= epsilon = 3.841496918314095. OK.

x\_4: 1.1883827255587676e-12 = |8.999999999999854 - 9.000000000001043| <= epsilon = 3.841496918314095. OK.

Шаг 40:

x\_1: 5.515587986337778e-13 = |3.9999999999998828 - 4.000000000000434| <= epsilon = 3.507331616209287. OK.

x\_2: 5.52891066263328e-13 = |1.999999999999699 - 1.999999999999146| <= epsilon = 3.507331616209287. OK.

x\_3: 3.028688411177427e-13 = |-6.999999999999503 - -6.9999999999992| <= epsilon = 3.507331616209287. OK.

x\_4: 2.327027459614328e-13 = |8.999999999999622 - 8.999999999999854| <= epsilon = 3.507331616209287. OK.

Шаг 41:

x\_1: 8.260059303211165e-14 = |3.9999999999998 - 3.9999999999998828| <= epsilon = 3.2022347870214904. OK.

x\_2: 2.6401103525586223e-13 = |1.999999999999963 - 1.999999999999699| <= epsilon = 3.2022347870214904. OK.

x\_3: 2.753353101070388e-13 = |-6.999999999999778 - -6.999999999999503| <= epsilon = 3.2022347870214904. OK.

x\_4: 1.1191048088221578e-13 = |8.999999999999734 - 8.999999999999622| <= epsilon = 3.2022347870214904. OK.

Шаг 42:

x\_1: 6.572520305780927e-14 = |3.999999999999866 - 3.9999999999998| <= epsilon = 2.9236778135890655. OK.

x\_2: 7.527312106958561e-14 = |2.000000000000038 - 1.999999999999963| <= epsilon = 2.9236778135890655. OK.

x\_3: 1.6431300764452317e-13 = |-6.999999999999942 - -6.999999999999778| <= epsilon = 2.9236778135890655. OK.

x\_4: 1.4033219031261979e-13 = |8.999999999999874 - 8.999999999999734| <= epsilon = 2.9236778135890655. OK.

Шаг 43:

x\_1: 7.37188088351104e-14 = |3.9999999999999396 - 3.999999999999866| <= epsilon = 2.669352038868964. OK.

x\_2: 2.6645352591003757e-15 = |2.000000000000041 - 2.000000000000038| <= epsilon = 2.669352038868964. OK.

x\_3: 6.217248937900877e-14 = |-7.000000000000004 - -6.999999999999942| <= epsilon = 2.669352038868964. OK.

x\_4: 8.704148513061227e-14 = |8.999999999999961 - 8.999999999999874| <= epsilon = 2.669352038868964. OK.

Шаг 44:

x\_1: 4.263256414560601e-14 = |3.9999999999999822 - 3.9999999999999396| <= epsilon = 2.4371496319790467. OK.

x\_2: 1.7319479184152442e-14 = |2.0000000000000235 - 2.000000000000041| <= epsilon = 2.4371496319790467. OK.

x\_3: 1.3322676295501878e-14 = |-7.000000000000018 - -7.000000000000004| <= epsilon = 2.4371496319790467. OK.

x\_4: 3.907985046680551e-14 = |9.0 - 8.999999999999961| <= epsilon = 2.4371496319790467. OK.

Шаг 45:

x\_1: 1.865174681370263e-14 = |4.000000000000001 - 3.9999999999999822| <= epsilon = 2.225146118670928. OK.

x\_2: 1.4210854715202004e-14 = |2.0000000000000093 - 2.0000000000000235| <= epsilon = 2.225146118670928. OK.

x\_3: 3.552713678800501e-15 = |-7.000000000000014 - -7.000000000000018| <= epsilon = 2.225146118670928. OK.

x\_4: 7.105427357601002e-15 = |9.000000000000007 - 9.0| <= epsilon = 2.225146118670928. OK.

Шаг 46:

x\_1: 3.552713678800501e-15 = |4.000000000000004 - 4.000000000000001| <= epsilon = 2.03158443144736. OK.

x\_2: 6.661338147750939e-15 = |2.0000000000000027 - 2.0000000000000093| <= epsilon = 2.03158443144736. OK.

x\_3: 8.881784197001252e-15 = |-7.000000000000005 - -7.000000000000014| <= epsilon = 2.03158443144736. OK.

x\_4: 0.0 = |9.000000000000007 - 9.000000000000007| <= epsilon = 2.03158443144736. OK.

Шаг 47:

x\_1: 8.881784197001252e-16 = |4.0000000000000036 - 4.000000000000004| <= epsilon = 1.8548603471328602. OK.

x\_2: 3.1086244689504383e-15 = |1.9999999999999996 - 2.0000000000000027| <= epsilon = 1.8548603471328602. OK.

x\_3: 2.6645352591003757e-15 = |-7.000000000000003 - -7.000000000000005| <= epsilon = 1.8548603471328602. OK.

x\_4: 3.552713678800501e-15 = |9.000000000000004 - 9.000000000000007| <= epsilon = 1.8548603471328602. OK.

Шаг 48:

x\_1: 1.7763568394002505e-15 = |4.000000000000002 - 4.0000000000000036| <= epsilon = 1.6935091912053672. OK.

x\_2: 4.440892098500626e-16 = |1.9999999999999991 - 1.9999999999999996| <= epsilon = 1.6935091912053672. OK.

x\_3: 2.6645352591003757e-15 = |-7.0 - -7.000000000000003| <= epsilon = 1.6935091912053672. OK.

x\_4: 1.7763568394002505e-15 = |9.000000000000002 - 9.000000000000004| <= epsilon = 1.6935091912053672. OK.

Шаг 49:

x\_1: 8.881784197001252e-16 = |4.000000000000001 - 4.000000000000002| <= epsilon = 1.5461936986955436. OK.

x\_2: 0.0 = |1.9999999999999991 - 1.9999999999999991| <= epsilon = 1.5461936986955436. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.5461936986955436. OK.

x\_4: 1.7763568394002505e-15 = |9.0 - 9.000000000000002| <= epsilon = 1.5461936986955436. OK.

Шаг 50:

x\_1: 8.881784197001252e-16 = |4.0 - 4.000000000000001| <= epsilon = 1.4116929310458581. OK.

x\_2: 6.661338147750939e-16 = |1.9999999999999998 - 1.9999999999999991| <= epsilon = 1.4116929310458581. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.4116929310458581. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 1.4116929310458581. OK.

Шаг 51:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 1.2888921570733016. OK.

x\_2: 2.220446049250313e-16 = |2.0 - 1.9999999999999998| <= epsilon = 1.2888921570733016. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.2888921570733016. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 1.2888921570733016. OK.

Шаг 52:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 1.1767736141699954. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 1.1767736141699954. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.1767736141699954. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 1.1767736141699954. OK.

Шаг 53:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 1.0744080731712895. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 1.0744080731712895. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.0744080731712895. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 1.0744080731712895. OK.

Шаг 54:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.9809471369816817. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.9809471369816817. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.9809471369816817. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.9809471369816817. OK.

Шаг 55:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.8956162091302049. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.8956162091302049. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.8956162091302049. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.8956162091302049. OK.

Шаг 56:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.8177080739792587. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.8177080739792587. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.8177080739792587. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.8177080739792587. OK.

Шаг 57:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.7465770353801857. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.7465770353801857. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.7465770353801857. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.7465770353801857. OK.

Шаг 58:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.6816335651972601. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.6816335651972601. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.6816335651972601. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.6816335651972601. OK.

Шаг 59:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.6223394173474984. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.6223394173474984. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.6223394173474984. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.6223394173474984. OK.

Шаг 60:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.5682031668618608. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.5682031668618608. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.5682031668618608. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.5682031668618608. OK.

Шаг 61:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.5187761369959533. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.5187761369959533. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.5187761369959533. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.5187761369959533. OK.

Шаг 62:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.47364868063446314. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.47364868063446314. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.47364868063446314. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.47364868063446314. OK.

Шаг 63:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.4324467851699155. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.4324467851699155. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.4324467851699155. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.4324467851699155. OK.

Шаг 64:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.39482897271726936. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.39482897271726936. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.39482897271726936. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.39482897271726936. OK.

Шаг 65:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.36048346997358877. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.36048346997358877. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.36048346997358877. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.36048346997358877. OK.

Шаг 66:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.3291256242668219. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.3291256242668219. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.3291256242668219. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.3291256242668219. OK.

Шаг 67:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.30049554437811443. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.30049554437811443. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.30049554437811443. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.30049554437811443. OK.

Шаг 68:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.27435594658498896. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.27435594658498896. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.27435594658498896. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.27435594658498896. OK.

Шаг 69:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.2504901880735754. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.2504901880735754. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.2504901880735754. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.2504901880735754. OK.

Шаг 70:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.22870047142097638. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.22870047142097638. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.22870047142097638. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.22870047142097638. OK.

Шаг 71:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.2088062052666663. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.2088062052666663. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.2088062052666663. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.2088062052666663. OK.

Шаг 72:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.1906425075863057. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.1906425075863057. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.1906425075863057. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.1906425075863057. OK.

Шаг 73:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.17405883916322792. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.17405883916322792. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.17405883916322792. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.17405883916322792. OK.

Шаг 74:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.15891775593192373. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.15891775593192373. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.15891775593192373. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.15891775593192373. OK.

Шаг 75:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.14509376985305025. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.14509376985305025. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.14509376985305025. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.14509376985305025. OK.

Шаг 76:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.13247230887899106. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.13247230887899106. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.13247230887899106. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.13247230887899106. OK.

Шаг 77:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.12094876739024842. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.12094876739024842. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.12094876739024842. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.12094876739024842. OK.

Шаг 78:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.11042763923276336. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.11042763923276336. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.11042763923276336. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.11042763923276336. OK.

Шаг 79:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.100821726170849. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.100821726170849. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.100821726170849. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.100821726170849. OK.

Шаг 80:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.09205141519546081. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.09205141519546081. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.09205141519546081. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.09205141519546081. OK.

Шаг 81:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.08404401869819485. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.08404401869819485. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.08404401869819485. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.08404401869819485. OK.

Шаг 82:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.07673317204243083. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.07673317204243083. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.07673317204243083. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.07673317204243083. OK.

Шаг 83:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.07005828353873984. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.07005828353873984. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.07005828353873984. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.07005828353873984. OK.

Шаг 84:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.06396403226599859. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.06396403226599859. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.06396403226599859. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.06396403226599859. OK.

Шаг 85:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.058399909576193156. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.058399909576193156. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.058399909576193156. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.058399909576193156. OK.

Шаг 86:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.05331980048294244. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.05331980048294244. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.05331980048294244. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.05331980048294244. OK.

Шаг 87:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.048681601464323904. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.048681601464323904. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.048681601464323904. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.048681601464323904. OK.

Шаг 88:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.04444687151238345. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.04444687151238345. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.04444687151238345. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.04444687151238345. OK.

Шаг 89:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.04058051353725652. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.04058051353725652. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.04058051353725652. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.04058051353725652. OK.

Шаг 90:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.03705048348540453. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.03705048348540453. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.03705048348540453. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.03705048348540453. OK.

Шаг 91:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.03382752476116247. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.03382752476116247. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.03382752476116247. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.03382752476116247. OK.

Шаг 92:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.030884925750505794. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.030884925750505794. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.030884925750505794. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.030884925750505794. OK.

Шаг 93:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.02819829843741355. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.02819829843741355. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.02819829843741355. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.02819829843741355. OK.

Шаг 94:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.025745376278018657. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.025745376278018657. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.025745376278018657. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.025745376278018657. OK.

Шаг 95:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.023505829657343057. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.023505829657343057. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.023505829657343057. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.023505829657343057. OK.

Шаг 96:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.021461097399138505. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.021461097399138505. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.021461097399138505. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.021461097399138505. OK.

Шаг 97:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.01959423293240058. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.01959423293240058. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.01959423293240058. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.01959423293240058. OK.

Шаг 98:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.01788976383959673. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.01788976383959673. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.01788976383959673. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.01788976383959673. OK.

Шаг 99:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.016333563622555784. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.016333563622555784. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.016333563622555784. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.016333563622555784. OK.

Шаг 100:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.014912734623225272. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.014912734623225272. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.014912734623225272. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.014912734623225272. OK.

Шаг 101:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.013615501128953482. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.013615501128953482. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.013615501128953482. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.013615501128953482. OK.

Шаг 102:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.012431111776361768. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.012431111776361768. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.012431111776361768. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.012431111776361768. OK.

Шаг 103:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.011349750444938486. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.011349750444938486. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.011349750444938486. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.011349750444938486. OK.

Шаг 104:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.010362454901848075. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.010362454901848075. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.010362454901848075. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.010362454901848075. OK.

Метод Зейделя.

Шаг 1:

x\_1: 1.5546334290476969 = |3.657278753184249 - 2.102645324136552| <= epsilon = 122.01115308436363. OK.

x\_2: 0.00666769637907505 = |2.376280216295933 - 2.382947912675008| <= epsilon = 122.01115308436363. OK.

x\_3: 0.9615126945784898 = |-7.38049660576513 - -6.41898391118664| <= epsilon = 122.01115308436363. OK.

x\_4: 0.0 = |8.429405584113937 - 8.429405584113937| <= epsilon = 122.01115308436363. OK.

Шаг 2:

x\_1: 0.2809815886084541 = |3.938260341792703 - 3.657278753184249| <= epsilon = 111.39760979705422. OK.

x\_2: 0.04286283824858028 = |2.073668959200581 - 2.0308061209520005| <= epsilon = 111.39760979705422. OK.

x\_3: 0.14370069017897524 = |-7.04863581962964 - -6.9049351294506645| <= epsilon = 111.39760979705422. OK.

x\_4: 0.0 = |8.884457769680692 - 8.884457769680692| <= epsilon = 111.39760979705422. OK.

Шаг 3:

x\_1: 0.050549171456556685 = |3.98880951324926 - 3.938260341792703| <= epsilon = 101.70732064073151. OK.

x\_2: 0.0011563218056935831 = |2.0123849279821733 - 2.0112286061764797| <= epsilon = 101.70732064073151. OK.

x\_3: 0.030484598448142997 = |-7.011723235307363 - -6.98123863685922| <= epsilon = 101.70732064073151. OK.

x\_4: 0.0 = |8.980992331433145 - 8.980992331433145| <= epsilon = 101.70732064073151. OK.

Шаг 4:

x\_1: 0.009168891321127859 = |3.9979784045703877 - 3.98880951324926| <= epsilon = 92.8599733042936. OK.

x\_2: 0.001227085179937859 = |2.0023788639495432 - 2.0011517787696054| <= epsilon = 92.8599733042936. OK.

x\_3: 0.004824581235899572 = |-7.001663011012671 - -6.996838429776772| <= epsilon = 92.8599733042936. OK.

x\_4: 0.0 = |8.996270172013492 - 8.996270172013492| <= epsilon = 92.8599733042936. OK.

Шаг 5:

x\_1: 0.0016550396997425665 = |3.9996334442701302 - 3.9979784045703877| <= epsilon = 84.78224170837917. OK.

x\_2: 6.786924996804444e-05 = |2.0004092017948163 - 2.0003413325448482| <= epsilon = 84.78224170837917. OK.

x\_3: 0.0009785217228879262 = |-7.000369999310161 - -6.999391477587273| <= epsilon = 84.78224170837917. OK.

x\_4: 0.0 = |8.99936882413597 - 8.99936882413597| <= epsilon = 84.78224170837917. OK.

Шаг 6:

x\_1: 0.0003002695322260607 = |3.9999337138023563 - 3.9996334442701302| <= epsilon = 77.40717828491637. OK.

x\_2: 3.590980478396233e-05 = |2.0000773249996575 - 2.0000414151948736| <= epsilon = 77.40717828491637. OK.

x\_3: 0.0001611013607600853 = |-7.000056390270986 - -6.999895288910226| <= epsilon = 77.40717828491637. OK.

x\_4: 0.0 = |8.999878978445045 - 8.999878978445045| <= epsilon = 77.40717828491637. OK.

Шаг 7:

x\_1: 5.426978571776431e-05 = |3.999987983588074 - 3.9999337138023563| <= epsilon = 70.67365912124309. OK.

x\_2: 2.9071255465140666e-06 = |2.000013505562487 - 2.0000105984369405| <= epsilon = 70.67365912124309. OK.

x\_3: 3.1620924907826975e-05 = |-7.000011821080506 - -6.999980200155598| <= epsilon = 70.67365912124309. OK.

x\_4: 0.0 = |8.999979111373918 - 8.999979111373918| <= epsilon = 70.67365912124309. OK.

Шаг 8:

x\_1: 9.842363762224693e-06 = |3.9999978259518363 - 3.999987983588074| <= epsilon = 64.52587737019928. OK.

x\_2: 1.075532936933854e-06 = |2.000002521143042 - 2.000001445610105| <= epsilon = 64.52587737019928. OK.

x\_3: 5.352404788716569e-06 = |-7.000001894445423 - -6.999996542040634| <= epsilon = 64.52587737019928. OK.

x\_4: 0.0 = |8.999996060655748 - 8.999996060655748| <= epsilon = 64.52587737019928. OK.

Шаг 9:

x\_1: 1.7800391454159126e-06 = |3.9999996059909817 - 3.9999978259518363| <= epsilon = 58.912880727616155. OK.

x\_2: 1.1107552122524567e-07 = |2.0000004450266164 - 2.000000333951095| <= epsilon = 58.912880727616155. OK.

x\_3: 1.0262658403092928e-06 = |-7.00000038056773 - -6.99999935430189| <= epsilon = 58.912880727616155. OK.

x\_4: 0.0 = |8.999999310497577 - 8.999999310497577| <= epsilon = 58.912880727616155. OK.

Шаг 10:

x\_1: 3.2270278715884615e-07 = |3.999999928693769 - 3.9999996059909817| <= epsilon = 53.78814914385422. OK.

x\_2: 3.288609917717622e-08 = |2.0000000823484614 - 2.000000049462362| <= epsilon = 53.78814914385422. OK.

x\_3: 1.7715597522283133e-07 = |-7.000000063193893 - -6.999999886037918| <= epsilon = 53.78814914385422. OK.

x\_4: 0.0 = |8.9999998714915 - 8.9999998714915| <= epsilon = 53.78814914385422. OK.

Шаг 11:

x\_1: 5.838594674401065e-08 = |3.9999999870797156 - 3.999999928693769| <= epsilon = 49.10920926949848. OK.

x\_2: 4.00797839361644e-09 = |2.000000014644709 - 2.0000000106367306| <= epsilon = 49.10920926949848. OK.

x\_3: 3.3407993704770433e-08 = |-7.000000012316823 - -6.999999978908829| <= epsilon = 49.10920926949848. OK.

x\_4: 0.0 = |8.99999997728348 - 8.99999997728348| <= epsilon = 49.10920926949848. OK.

Шаг 12:

x\_1: 1.0581564957590217e-08 = |3.9999999976612806 - 3.9999999870797156| <= epsilon = 44.837282439768714. OK.

x\_2: 1.0229177505038933e-09 = |2.000000002692972 - 2.0000000016700543| <= epsilon = 44.837282439768714. OK.

x\_3: 5.847758721699847e-09 = |-7.000000002097361 - -6.999999996249603| <= epsilon = 44.837282439768714. OK.

x\_4: 0.0 = |8.999999995801382 - 8.999999995801382| <= epsilon = 44.837282439768714. OK.

Шаг 13:

x\_1: 1.9150236951759325e-09 = |3.9999999995763043 - 3.9999999976612806| <= epsilon = 40.936963280168946. OK.

x\_2: 1.3993384229138428e-10 = |2.0000000004814464 - 2.0000000003415126| <= epsilon = 40.936963280168946. OK.

x\_3: 1.0898517643909145e-09 = |-7.00000000040013 - -6.999999999310278| <= epsilon = 40.936963280168946. OK.

x\_4: 0.0 = |8.99999999925258 - 8.99999999925258| <= epsilon = 40.936963280168946. OK.

Шаг 14:

x\_1: 3.469917686516055e-10 = |3.999999999923296 - 3.9999999995763043| <= epsilon = 37.37592626968641. OK.

x\_2: 3.22537552222002e-11 = |2.0000000000881393 - 2.0000000000558855| <= epsilon = 37.37592626968641. OK.

x\_3: 1.9266099826609207e-10 = |-7.000000000069363 - -6.999999999876702| <= epsilon = 37.37592626968641. OK.

x\_4: 0.0 = |8.999999999862673 - 8.999999999862673| <= epsilon = 37.37592626968641. OK.

Шаг 15:

x\_1: 6.280931330593376e-11 = |3.9999999999861053 - 3.999999999923296| <= epsilon = 34.12465782956018. OK.

x\_2: 4.786393503763975e-12 = |2.0000000000158167 - 2.0000000000110303| <= epsilon = 34.12465782956018. OK.

x\_3: 3.560796102419772e-11 = |-7.000000000013034 - -6.999999999977426| <= epsilon = 34.12465782956018. OK.

x\_4: 0.0 = |8.999999999975433 - 8.999999999975433| <= epsilon = 34.12465782956018. OK.

Шаг 16:

x\_1: 1.1379341913198004e-11 = |3.9999999999974847 - 3.9999999999861053| <= epsilon = 31.156211717193464. OK.

x\_2: 1.0276224315930449e-12 = |2.0000000000028866 - 2.000000000001859| <= epsilon = 31.156211717193464. OK.

x\_3: 6.338929381399794e-12 = |-7.000000000002289 - -6.99999999999595| <= epsilon = 31.156211717193464. OK.

x\_4: 0.0 = |8.999999999995506 - 8.999999999995506| <= epsilon = 31.156211717193464. OK.

Шаг 17:

x\_1: 2.0596857552845904e-12 = |3.9999999999995444 - 3.9999999999974847| <= epsilon = 28.445985697934674. OK.

x\_2: 1.616484723854228e-13 = |2.0000000000005196 - 2.000000000000358| <= epsilon = 28.445985697934674. OK.

x\_3: 1.1644019082268642e-12 = |-7.000000000000425 - -6.999999999999261| <= epsilon = 28.445985697934674. OK.

x\_4: 0.0 = |8.999999999999194 - 8.999999999999194| <= epsilon = 28.445985697934674. OK.

Шаг 18:

x\_1: 3.739231146937527e-13 = |3.9999999999999183 - 3.9999999999995444| <= epsilon = 25.971517643801466. OK.

x\_2: 3.241851231905457e-14 = |2.000000000000094 - 2.0000000000000617| <= epsilon = 25.971517643801466. OK.

x\_3: 2.0872192862952943e-13 = |-7.0000000000000755 - -6.999999999999867| <= epsilon = 25.971517643801466. OK.

x\_4: 0.0 = |8.999999999999854 - 8.999999999999854| <= epsilon = 25.971517643801466. OK.

Шаг 19:

x\_1: 6.661338147750939e-14 = |3.999999999999985 - 3.9999999999999183| <= epsilon = 23.712299369231015. OK.

x\_2: 5.773159728050814e-15 = |2.000000000000017 - 2.000000000000011| <= epsilon = 23.712299369231015. OK.

x\_3: 3.8191672047105385e-14 = |-7.000000000000013 - -6.999999999999975| <= epsilon = 23.712299369231015. OK.

x\_4: 0.0 = |8.999999999999973 - 8.999999999999973| <= epsilon = 23.712299369231015. OK.

Шаг 20:

x\_1: 1.3322676295501878e-14 = |3.9999999999999982 - 3.999999999999985| <= epsilon = 21.649606660942645. OK.

x\_2: 8.881784197001252e-16 = |2.0000000000000027 - 2.0000000000000018| <= epsilon = 21.649606660942645. OK.

x\_3: 7.105427357601002e-15 = |-7.000000000000003 - -6.999999999999996| <= epsilon = 21.649606660942645. OK.

x\_4: 0.0 = |8.999999999999996 - 8.999999999999996| <= epsilon = 21.649606660942645. OK.

Шаг 21:

x\_1: 1.7763568394002505e-15 = |4.0 - 3.9999999999999982| <= epsilon = 19.766344093214446. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 19.766344093214446. OK.

x\_3: 8.881784197001252e-16 = |-7.0 - -6.999999999999999| <= epsilon = 19.766344093214446. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 19.766344093214446. OK.

Шаг 22:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 18.04690334241582. OK.

x\_2: 4.440892098500626e-16 = |2.0 - 1.9999999999999996| <= epsilon = 18.04690334241582. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 18.04690334241582. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 18.04690334241582. OK.

Шаг 23:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 16.477033826518536. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 16.477033826518536. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 16.477033826518536. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 16.477033826518536. OK.

Шаг 24:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 15.043724597457347. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 15.043724597457347. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 15.043724597457347. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 15.043724597457347. OK.

Шаг 25:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 13.735096507473848. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 13.735096507473848. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 13.735096507473848. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 13.735096507473848. OK.

Шаг 26:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 12.54030375572722. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 12.54030375572722. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 12.54030375572722. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 12.54030375572722. OK.

Шаг 27:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 11.449443999198316. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 11.449443999198316. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 11.449443999198316. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 11.449443999198316. OK.

Шаг 28:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 10.453476282893783. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 10.453476282893783. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 10.453476282893783. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 10.453476282893783. OK.

Шаг 29:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 9.544146109162526. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 9.544146109162526. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 9.544146109162526. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 9.544146109162526. OK.

Шаг 30:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 8.713917025105259. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 8.713917025105259. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 8.713917025105259. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 8.713917025105259. OK.

Шаг 31:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 7.955908161079289. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 7.955908161079289. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 7.955908161079289. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 7.955908161079289. OK.

Шаг 32:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 7.26383720262283. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 7.26383720262283. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 7.26383720262283. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 7.26383720262283. OK.

Шаг 33:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 6.631968323154909. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 6.631968323154909. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 6.631968323154909. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 6.631968323154909. OK.

Шаг 34:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 6.055064645921404. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 6.055064645921404. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 6.055064645921404. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 6.055064645921404. OK.

Шаг 35:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 5.528344841195785. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 5.528344841195785. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 5.528344841195785. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 5.528344841195785. OK.

Шаг 36:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 5.047443499015742. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 5.047443499015742. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 5.047443499015742. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 5.047443499015742. OK.

Шаг 37:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 4.608374949028262. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 4.608374949028262. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 4.608374949028262. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 4.608374949028262. OK.

Шаг 38:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 4.2075002275850135. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 4.2075002275850135. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 4.2075002275850135. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 4.2075002275850135. OK.

Шаг 39:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 3.841496918314095. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 3.841496918314095. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 3.841496918314095. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 3.841496918314095. OK.

Шаг 40:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 3.507331616209287. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 3.507331616209287. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 3.507331616209287. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 3.507331616209287. OK.

Шаг 41:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 3.2022347870214904. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 3.2022347870214904. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 3.2022347870214904. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 3.2022347870214904. OK.

Шаг 42:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 2.9236778135890655. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 2.9236778135890655. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 2.9236778135890655. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 2.9236778135890655. OK.

Шаг 43:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 2.669352038868964. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 2.669352038868964. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 2.669352038868964. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 2.669352038868964. OK.

Шаг 44:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 2.4371496319790467. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 2.4371496319790467. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 2.4371496319790467. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 2.4371496319790467. OK.

Шаг 45:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 2.225146118670928. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 2.225146118670928. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 2.225146118670928. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 2.225146118670928. OK.

Шаг 46:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 2.03158443144736. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 2.03158443144736. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 2.03158443144736. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 2.03158443144736. OK.

Шаг 47:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 1.8548603471328602. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 1.8548603471328602. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.8548603471328602. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 1.8548603471328602. OK.

Шаг 48:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 1.6935091912053672. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 1.6935091912053672. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.6935091912053672. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 1.6935091912053672. OK.

Шаг 49:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 1.5461936986955436. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 1.5461936986955436. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.5461936986955436. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 1.5461936986955436. OK.

Шаг 50:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 1.4116929310458581. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 1.4116929310458581. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.4116929310458581. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 1.4116929310458581. OK.

Шаг 51:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 1.2888921570733016. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 1.2888921570733016. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.2888921570733016. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 1.2888921570733016. OK.

Шаг 52:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 1.1767736141699954. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 1.1767736141699954. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.1767736141699954. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 1.1767736141699954. OK.

Шаг 53:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 1.0744080731712895. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 1.0744080731712895. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 1.0744080731712895. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 1.0744080731712895. OK.

Шаг 54:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.9809471369816817. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.9809471369816817. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.9809471369816817. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.9809471369816817. OK.

Шаг 55:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.8956162091302049. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.8956162091302049. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.8956162091302049. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.8956162091302049. OK.

Шаг 56:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.8177080739792587. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.8177080739792587. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.8177080739792587. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.8177080739792587. OK.

Шаг 57:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.7465770353801857. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.7465770353801857. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.7465770353801857. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.7465770353801857. OK.

Шаг 58:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.6816335651972601. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.6816335651972601. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.6816335651972601. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.6816335651972601. OK.

Шаг 59:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.6223394173474984. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.6223394173474984. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.6223394173474984. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.6223394173474984. OK.

Шаг 60:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.5682031668618608. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.5682031668618608. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.5682031668618608. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.5682031668618608. OK.

Шаг 61:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.5187761369959533. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.5187761369959533. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.5187761369959533. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.5187761369959533. OK.

Шаг 62:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.47364868063446314. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.47364868063446314. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.47364868063446314. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.47364868063446314. OK.

Шаг 63:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.4324467851699155. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.4324467851699155. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.4324467851699155. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.4324467851699155. OK.

Шаг 64:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.39482897271726936. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.39482897271726936. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.39482897271726936. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.39482897271726936. OK.

Шаг 65:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.36048346997358877. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.36048346997358877. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.36048346997358877. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.36048346997358877. OK.

Шаг 66:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.3291256242668219. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.3291256242668219. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.3291256242668219. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.3291256242668219. OK.

Шаг 67:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.30049554437811443. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.30049554437811443. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.30049554437811443. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.30049554437811443. OK.

Шаг 68:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.27435594658498896. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.27435594658498896. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.27435594658498896. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.27435594658498896. OK.

Шаг 69:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.2504901880735754. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.2504901880735754. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.2504901880735754. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.2504901880735754. OK.

Шаг 70:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.22870047142097638. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.22870047142097638. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.22870047142097638. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.22870047142097638. OK.

Шаг 71:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.2088062052666663. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.2088062052666663. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.2088062052666663. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.2088062052666663. OK.

Шаг 72:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.1906425075863057. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.1906425075863057. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.1906425075863057. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.1906425075863057. OK.

Шаг 73:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.17405883916322792. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.17405883916322792. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.17405883916322792. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.17405883916322792. OK.

Шаг 74:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.15891775593192373. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.15891775593192373. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.15891775593192373. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.15891775593192373. OK.

Шаг 75:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.14509376985305025. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.14509376985305025. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.14509376985305025. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.14509376985305025. OK.

Шаг 76:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.13247230887899106. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.13247230887899106. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.13247230887899106. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.13247230887899106. OK.

Шаг 77:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.12094876739024842. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.12094876739024842. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.12094876739024842. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.12094876739024842. OK.

Шаг 78:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.11042763923276336. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.11042763923276336. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.11042763923276336. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.11042763923276336. OK.

Шаг 79:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.100821726170849. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.100821726170849. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.100821726170849. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.100821726170849. OK.

Шаг 80:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.09205141519546081. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.09205141519546081. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.09205141519546081. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.09205141519546081. OK.

Шаг 81:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.08404401869819485. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.08404401869819485. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.08404401869819485. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.08404401869819485. OK.

Шаг 82:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.07673317204243083. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.07673317204243083. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.07673317204243083. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.07673317204243083. OK.

Шаг 83:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.07005828353873984. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.07005828353873984. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.07005828353873984. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.07005828353873984. OK.

Шаг 84:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.06396403226599859. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.06396403226599859. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.06396403226599859. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.06396403226599859. OK.

Шаг 85:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.058399909576193156. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.058399909576193156. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.058399909576193156. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.058399909576193156. OK.

Шаг 86:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.05331980048294244. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.05331980048294244. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.05331980048294244. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.05331980048294244. OK.

Шаг 87:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.048681601464323904. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.048681601464323904. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.048681601464323904. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.048681601464323904. OK.

Шаг 88:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.04444687151238345. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.04444687151238345. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.04444687151238345. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.04444687151238345. OK.

Шаг 89:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.04058051353725652. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.04058051353725652. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.04058051353725652. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.04058051353725652. OK.

Шаг 90:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.03705048348540453. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.03705048348540453. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.03705048348540453. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.03705048348540453. OK.

Шаг 91:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.03382752476116247. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.03382752476116247. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.03382752476116247. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.03382752476116247. OK.

Шаг 92:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.030884925750505794. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.030884925750505794. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.030884925750505794. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.030884925750505794. OK.

Шаг 93:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.02819829843741355. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.02819829843741355. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.02819829843741355. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.02819829843741355. OK.

Шаг 94:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.025745376278018657. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.025745376278018657. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.025745376278018657. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.025745376278018657. OK.

Шаг 95:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.023505829657343057. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.023505829657343057. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.023505829657343057. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.023505829657343057. OK.

Шаг 96:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.021461097399138505. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.021461097399138505. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.021461097399138505. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.021461097399138505. OK.

Шаг 97:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.01959423293240058. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.01959423293240058. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.01959423293240058. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.01959423293240058. OK.

Шаг 98:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.01788976383959673. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.01788976383959673. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.01788976383959673. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.01788976383959673. OK.

Шаг 99:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.016333563622555784. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.016333563622555784. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.016333563622555784. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.016333563622555784. OK.

Шаг 100:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.014912734623225272. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.014912734623225272. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.014912734623225272. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.014912734623225272. OK.

Шаг 101:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.013615501128953482. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.013615501128953482. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.013615501128953482. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.013615501128953482. OK.

Шаг 102:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.012431111776361768. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.012431111776361768. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.012431111776361768. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.012431111776361768. OK.

Шаг 103:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.011349750444938486. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.011349750444938486. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.011349750444938486. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.011349750444938486. OK.

Шаг 104:

x\_1: 0.0 = |4.0 - 4.0| <= epsilon = 0.010362454901848075. OK.

x\_2: 0.0 = |2.0 - 2.0| <= epsilon = 0.010362454901848075. OK.

x\_3: 0.0 = |-7.0 - -7.0| <= epsilon = 0.010362454901848075. OK.

x\_4: 0.0 = |9.0 - 9.0| <= epsilon = 0.010362454901848075. OK.

# Лабораторная работа 1.4

Задание: Реализовать метод вращений в виде программы, задавая в качестве входных данных матрицу и точность вычислений. Используя разработанное программное обеспечение, найти собственные значения и собственные векторы симметрических матриц. Проанализировать зависимость погрешности вычислений от числа итераций.

Условие:



Исходный код:

from math import atan, cos, sin, pi, sqrt

def matrix\_multiply(A, B): #функция перемножения двух матриц

res = []

for i in range(len(A)):

res.append([])

for j in range(len(A)):

c = 0

for k in range(len(A)):

c += A[i][k]\*B[k][j]

res[i].append(c)

return res

def transp(A): #функция транспонирования

res = []

for i in range(len(A)):

res.append([])

for j in range(len(A)):

res[i].append(A[j][i])

return res

def matrix\_output(A): #функция для вывода матрицы

for i in range(len(A)):

for j in range(len(A[i])):

print("%.16f"%A[i][j], end='\t')

print()

N = 3

epsilon = 0.001 #точность

matrix = [] #инициализация списка строк матрицы

matrix = [[-9, 7, 5],

[7, 8, 9],

[5, 9, 8]]

matrix\_original = matrix #сохраняем исходную матрицу для проверки решения

print("Исходная матрица:")

matrix\_output(matrix)

print("\nИтерации")

V = []

for i in range(N):

V.append([])

for j in range(N):

V[i].append(1 if i == j else 0)

iter = 1 #счётчик итераций

sum\_offdiag = 1 #сумма квадратов внедиагональных элементов матрицы

while sum\_offdiag > epsilon:

print(f'Шаг {iter}')

a\_max = 0

k = 0

m = 0

for i in range(N): #поиск максимального по модулю элемента выше главн. диагонали

for j in range(i+1, N):

if abs(matrix[i][j]) > abs(a\_max):

k = i

m = j

a\_max = matrix[i][j]

print(f'a\_max = {a\_max}, k = {k+1}, m = {m+1}')

phi = (atan((2\*a\_max)/(matrix[k][k]-matrix[m][m])) / 2 if matrix[k][k] != matrix[m][m]

else (pi/4 if matrix[k][m] > 0 else -pi/4)) #вычисление угла поворота по формуле

print(f'phi = {phi}') #в случае деления на ноль присваивается значение pi/4

H = []

for i in range(N): #построение матрицы поворота H по алгоритму

H.append([])

for j in range(N):

h = 0.0

if i == j:

if (i == k) or (i == m):

h = cos(phi)

else:

h = 1.0

elif i == k and j == m:

h = -sin(phi)

elif i == m and j == k:

h = sin(phi)

H[i].append(h)

print(f'H\_{iter}:')

matrix\_output(H)

H\_t = transp(H)

V = matrix\_multiply(V, H)

print(f'H\_{iter}^-1:')

matrix\_output(H\_t)

matrix = matrix\_multiply(matrix\_multiply(H\_t, matrix), H) #вычисление поворота

print(f'A\_{iter}:')

matrix\_output(matrix)

print()

iter += 1

sum\_offdiag = 0

for i in range(N): #сумма квадратов внедиагональных элементов матрицы

for j in range(i+1, N):

sum\_offdiag += matrix[i][j]\*\*2

sum\_offdiag = sqrt(sum\_offdiag)

print("Конец итерационного процесса.\n")

lamdas = []

for i in range(N): #собственные значения на главной диагонали

lamdas.append(matrix[i][i])

print("матрица V:")

matrix\_output(V)

for i in range(N):

norm = 0

for j in range(N):

norm = (V[j][i] if abs(norm) < abs(V[j][i]) else norm)

for j in range(N):

V[j][i] /= norm

print()

print("ПРОВЕРКА:")

V = transp(V)

for k in range(N):

print(f'\tПара {k+1}')

res1 = []

res2 = []

for i in range(N):

tmp = 0

for j in range(N):

tmp += matrix\_original[i][j] \* V[k][j]

res1.append(tmp)

for i in range(N):

res2.append(lamdas[k]\*V[k][i])

print(f'A\*V\_{k+1}:')

for i in range(N):

print(res1[i], end='\t')

print(f'\nlambda\_{k+1}\*V\_{k+1}:')

for i in range(N):

print(res2[i], end='\t')

print()

print("\nРЕЗУЛЬТАТЫ:")

for i in range(N):

print(f'lambda\_{i+1} = {lamdas[i]}')

for j in range(N):

print(f'V\_{i+1}{j+1}', end='\t\t\t')

print()

for j in range(N):

print(V[j][i], end='\t')

print()

Результат:

Исходная матрица:

-9.0000000000000000 7.0000000000000000 5.0000000000000000

7.0000000000000000 8.0000000000000000 9.0000000000000000

5.0000000000000000 9.0000000000000000 8.0000000000000000

Итерации

Шаг 1

a\_max = 9, k = 2, m = 3

phi = 0.7853981633974483

H\_1:

1.0000000000000000 0.0000000000000000 0.0000000000000000

0.0000000000000000 0.7071067811865476 -0.7071067811865475

0.0000000000000000 0.7071067811865475 0.7071067811865476

H\_1^-1:

1.0000000000000000 0.0000000000000000 0.0000000000000000

0.0000000000000000 0.7071067811865476 0.7071067811865475

0.0000000000000000 -0.7071067811865475 0.7071067811865476

A\_1:

-9.0000000000000000 8.4852813742385695 -1.4142135623730940

8.4852813742385695 17.0000000000000000 0.0000000000000018

-1.4142135623730940 0.0000000000000013 -1.0000000000000004

Шаг 2

a\_max = 8.48528137423857, k = 1, m = 2

phi = -0.2891403649986587

H\_2:

0.9584893359659982 0.2851283795756921 0.0000000000000000

-0.2851283795756921 0.9584893359659982 0.0000000000000000

0.0000000000000000 0.0000000000000000 1.0000000000000000

H\_2^-1:

0.9584893359659982 -0.2851283795756921 0.0000000000000000

0.2851283795756921 0.9584893359659982 0.0000000000000000

0.0000000000000000 0.0000000000000000 1.0000000000000000

A\_2:

-11.5241746962600242 -0.0000000000000004 -1.3555086183130960

0.0000000000000000 19.5241746962600224 -0.4032324214134055

-1.3555086183130960 -0.4032324214134060 -1.0000000000000004

Шаг 3

a\_max = -1.355508618313096, k = 1, m = 3

phi = 0.12605890443951115

H\_3:

0.9920650923603646 0.0000000000000000 -0.1257253058060361

0.0000000000000000 1.0000000000000000 0.0000000000000000

0.1257253058060361 0.0000000000000000 0.9920650923603646

H\_3^-1:

0.9920650923603646 0.0000000000000000 0.1257253058060361

0.0000000000000000 1.0000000000000000 0.0000000000000000

-0.1257253058060361 0.0000000000000000 0.9920650923603646

A\_3:

-11.6959595285986513 -0.0506965194931093 -0.0000000000000002

-0.0506965194931088 19.5241746962600224 -0.4000328093921836

-0.0000000000000001 -0.4000328093921841 -0.8282151676613716

Шаг 4

a\_max = -0.4000328093921836, k = 2, m = 3

phi = -0.01964520837182084

H\_4:

1.0000000000000000 0.0000000000000000 0.0000000000000000

0.0000000000000000 0.9998070390999847 0.0196439447698691

0.0000000000000000 -0.0196439447698691 0.9998070390999847

H\_4^-1:

1.0000000000000000 0.0000000000000000 0.0000000000000000

0.0000000000000000 0.9998070390999847 -0.0196439447698691

0.0000000000000000 0.0196439447698691 0.9998070390999847

A\_4:

-11.6959595285986513 -0.0506867370470803 -0.0009958796289475

-0.0506867370470798 19.5320344352961790 -0.0000000000000001

-0.0009958796289473 -0.0000000000000005 -0.8360749066975255

Шаг 5

a\_max = -0.05068673704708031, k = 1, m = 2

phi = 0.001623112873028517

H\_5:

0.9999986827525899 -0.0016231121603480 0.0000000000000000

0.0016231121603480 0.9999986827525899 0.0000000000000000

0.0000000000000000 0.0000000000000000 1.0000000000000000

H\_5^-1:

0.9999986827525899 0.0016231121603480 0.0000000000000000

-0.0016231121603480 0.9999986827525899 0.0000000000000000

0.0000000000000000 0.0000000000000000 1.0000000000000000

A\_5:

-11.6960417989662915 0.0000000000000000 -0.0009958783171276

0.0000000000000005 19.5321167056638210 0.0000016164243359

-0.0009958783171274 0.0000016164243355 -0.8360749066975255

Конец итерационного процесса.

матрица V:

0.9513491073652171 0.2858968116149279 -0.1148820657188184

-0.2877958229188600 0.6913737726511047 -0.6627034561556433

-0.1100383579751200 0.6635249201036814 0.7400177296359475

ПРОВЕРКА:

Пара 1

A\*V\_1:

-11.695921539686717 3.538902722843562 1.352054946808912

lambda\_1\*V\_1:

-11.696041798966291 3.538209000636266 1.3528296020911104

Пара 2

A\*V\_2:

8.076918629888189 19.53211515627037 18.74535581971548

lambda\_2\*V\_2:

8.076918898481205 19.53211670566382 18.745354089562106

Пара 3

A\*V\_3:

0.12851455140998524 0.7491137512602144 -0.8359253733172345

lambda\_3\*V\_3:

0.1297942042879594 0.748724940071412 -0.8360749066975255

РЕЗУЛЬТАТЫ:

lambda\_1 = -11.696041798966291

V\_11 V\_12 V\_13

1.0 0.41351989752032875 -0.1552423153095734

lambda\_2 = 19.53211670566382

V\_21 V\_22 V\_23

-0.30251336832166387 1.0 -0.8955237551965964

lambda\_3 = -0.8360749066975255

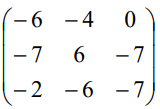
V\_31 V\_32 V\_33

-0.11566559228701413 0.9597195415142296 1.0

# Лабораторная работа 1.5

Задание: Реализовать алгоритм QR-разложения матриц в виде программы. На его основе разработать программу, реализующую QR – алгоритм решения полной проблемы собственных значений произвольных матриц, задавая в качестве входных данных матрицу и точность вычислений.

Условие:



Исходный код:

import numpy as np

epsilon = 0.01

def qr(A):

m, n = A.shape

Q = np.eye(m)

for i in range(n - (m == n)):

H = np.eye(m)

H[i:, i:] = make\_householder(A[i:, i])

Q = np.dot(Q, H)

A = np.dot(H, A)

return Q, A

def make\_householder(a):

v = a / (a[0] + np.copysign(np.linalg.norm(a), a[0]))

v[0] = 1

H = np.eye(a.shape[0])

H -= (2 / np.dot(v, v)) \* np.dot(v[:, None], v[None, :])

return H

# построим qr-разложение матрицы методом Хаусхолдера

a = np.array([[-6, -4, 0],

[-7, 6, -7],

[-2, -6, -7]])

q, r = qr(a)

print('Q:\n', q.round(6))

print('R:\n', r.round(6))

# вычислим собственные значения матрицы А

n\_i = 0

n\_j = 0

k = 0

eps = 1

f = 0

while epsilon < eps:

k += 1

for i in range(k):

if k == 1:

q, r = qr(a)

a = np.dot(r, q)

print(f'Iteration {i+1}:')

print(a)

n\_i = a[0][0]

i += 1

if i > 0 and epsilon < eps:

f += 1

q, r = qr(a)

a = np.dot(r, q)

print(f'Iteration {f+1}:')

print(a)

n\_j = n\_i

n\_i = a[0][0]

eps = abs(n\_i - n\_j)

print(f'n\_i={n\_i},n\_j={n\_j},eps={eps}')

i += 1

Результат:

Q:

[[-0.635999 0.470603 -0.611586]

[-0.741999 -0.590655 0.317119]

[-0.212 0.655483 0.724843]]

R:

[[ 9.433981 -0.635999 6.677987]

[ 0. -9.359247 -0.453796]

[-0. -0. -7.293735]]

Iteration 1:

[[-6.94382022 9.19263024 -1.13088901]

[ 7.04075204 5.23063398 -3.29692517]

[ 1.54626863 -4.7809222 -5.28681375]]

Iteration 2:

[[-9.72454015 -4.67735619 1.03668933]

[-7.65317692 8.00928725 -1.03868888]

[-0.89659337 -2.2248208 -5.2847471 ]]

n\_i=-9.72454015253476,n\_j=-6.943820224719097,eps=2.780719927815662

Iteration 3:

[[-9.05283036 8.18131591 -2.51757861]

[ 5.83417729 7.3526246 -1.2856189 ]

[ 0.39595112 -1.33341856 -5.29979424]]

n\_i=-9.052830364636753,n\_j=-9.72454015253476,eps=0.6717097878980063

Iteration 4:

[[-10.59993906 -3.15845961 1.99030156]

[ -5.71644859 9.15282405 0.98169485]

[ -0.20550733 -0.6416091 -5.55288499]]

n\_i=-10.599939058489785,n\_j=-9.052830364636753,eps=1.5471086938530316

Iteration 5:

[[-9.82649072 6.90745683 -2.77934171]

[ 4.5512279 8.39400633 -0.81943462]

[ 0.09523782 -0.38934423 -5.5675156 ]]

n\_i=-9.826490721567131,n\_j=-10.599939058489785,eps=0.7734483369226535

Iteration 6:

[[-10.96069144 -1.98361064 2.29522768]

[ -4.40851605 9.60648731 1.42300677]

[ -0.04968166 -0.19601575 -5.64579587]]

n\_i=-10.96069144138607,n\_j=-9.826490721567131,eps=1.1342007198189386

Iteration 7:

[[-10.29897876 5.96840529 -2.80634073]

[ 3.60524056 8.94358837 -0.75615662]

[ 0.02374256 -0.11891175 -5.6446096 ]]

n\_i=-10.298978761770746,n\_j=-10.96069144138607,eps=0.6617126796153237

Iteration 8:

[[-11.17876627 -1.05090304 2.42250273]

[ -3.43729935 9.84572002 1.48011292]

[ -0.01233127 -0.06154075 -5.66695375]]

n\_i=-11.178766273986543,n\_j=-10.298978761770746,eps=0.8797875122157972

Iteration 9:

[[-1.06206441e+01 5.22488856e+00 -2.78995844e+00]

[ 2.85751725e+00 9.28565606e+00 -7.97502227e-01]

[ 5.97319512e-03 -3.70290793e-02 -5.66501195e+00]]

n\_i=-10.620644104504297,n\_j=-11.178766273986543,eps=0.558122169482246

Iteration 10:

[[-1.13033685e+01 -3.06407747e-01 2.49107963e+00]

[-2.68159954e+00 9.97481690e+00 1.44250816e+00]

[-3.08018016e-03 -1.95557280e-02 -5.67144837e+00]]

n\_i=-11.303368530284416,n\_j=-10.620644104504297,eps=0.682724425780119

Iteration 11:

[[-1.08399655e+01 4.62616088e+00 -2.76766327e+00]

[ 2.25687853e+00 9.51041807e+00 -8.59127783e-01]

[ 1.50347726e-03 -1.16511316e-02 -5.67045258e+00]]

n\_i=-10.839965496640188,n\_j=-11.303368530284416,eps=0.4634030336442283

Iteration 12:

[[-1.13676495e+01 2.82075160e-01 2.53490956e+00]

[-2.08981358e+00 1.00399877e+01 1.38819242e+00]

[-7.70223794e-04 -6.25079919e-03 -5.67233825e+00]]

n\_i=-11.367649455941825,n\_j=-10.839965496640188,eps=0.5276839593016369

Iteration 13:

[[-1.09890805e+01 4.14627814e+00 -2.74717922e+00]

[ 1.77623705e+00 9.66100923e+00 -9.16835133e-01]

[ 3.77971932e-04 -3.68962680e-03 -5.67192870e+00]]

n\_i=-10.98908053402719,n\_j=-11.367649455941825,eps=0.37856892191463487

Iteration 14:

[[-1.13961392e+01 7.43565806e-01 2.56561707e+00]

[-1.62733637e+00 1.00686289e+01 1.33793769e+00]

[-1.92606754e-04 -2.00394737e-03 -5.67248970e+00]]

n\_i=-11.396139151915868,n\_j=-10.98908053402719,eps=0.40705861788867814

Iteration 15:

[[-1.10908293e+01 3.76451835e+00 -2.72985717e+00]

[ 1.39419695e+00 9.76316550e+00 -9.64987708e-01]

[ 9.49057275e-05 -1.17334522e-03 -5.67233616e+00]]

n\_i=-11.090829341841754,n\_j=-11.396139151915868,eps=0.30530981007411384

Iteration 16:

[[-1.14047923e+01 1.10384475e+00 2.58810531e+00]

[-1.26675921e+00 1.00772975e+01 1.29615978e+00]

[-4.81613571e-05 -6.43390084e-04 -5.67250528e+00]]

n\_i=-11.404792257546053,n\_j=-11.090829341841754,eps=0.31396291570429824

Iteration 17:

[[-1.11608586e+01 3.46264277e+00 -2.71562594e+00]

[ 1.09222215e+00 9.83330912e+00 -1.00354545e+00]

[ 2.38078115e-05 -3.74192502e-04 -5.67245052e+00]]

n\_i=-11.160858606515102,n\_j=-11.404792257546053,eps=0.24393365103095022

Iteration 18:

[[-1.14032195e+01 1.38449139e+00 2.60493474e+00]

[-9.86021465e-01 1.00757216e+01 1.26269513e+00]

[-1.20427819e-05 -2.06704825e-04 -5.67250207e+00]]

n\_i=-11.403219530053272,n\_j=-11.160858606515102,eps=0.24236092353817007

Iteration 19:

[[-1.12096146e+01 3.22494979e+00 -2.70410218e+00]

[ 8.54495022e-01 9.88209768e+00 -1.03391998e+00]

[ 5.96836006e-06 -1.19562785e-04 -5.67248313e+00]]

n\_i=-11.209614554912052,n\_j=-11.403219530053272,eps=0.19360497514121988

Iteration 20:

[[-1.13969355e+01 1.60290602e+00 2.61767755e+00]

[-7.67578747e-01 1.00694345e+01 1.23627918e+00]

[-3.01148435e-06 -6.64246286e-05 -5.67249900e+00]]

n\_i=-11.396935464019448,n\_j=-11.209614554912052,eps=0.1873209091073953

Iteration 21:

[[-1.12439988e+01 3.03833855e+00 -2.69485746e+00]

[ 6.67872229e-01 9.91649140e+00 -1.05768880e+00]

[ 1.49549069e-06 -3.82531671e-05 -5.67249257e+00]]

n\_i=-11.243998824728,n\_j=-11.396935464019448,eps=0.15293663929144863

Iteration 22:

[[-1.13889718e+01 1.77284426e+00 2.62739600e+00]

[-5.97631784e-01 1.00614693e+01 1.21555656e+00]

[-7.53134353e-07 -2.13459018e-05 -5.67249750e+00]]

n\_i=-11.388971824278512,n\_j=-11.243998824728,eps=0.14497299955051268

Iteration 23:

[[-1.12685688e+01 2.89212376e+00 -2.68749090e+00]

[ 5.21653595e-01 9.94106414e+00 -1.07623798e+00]

[ 3.74597588e-07 -1.22499611e-05 -5.67249535e+00]]

n\_i=-11.268568784773139,n\_j=-11.388971824278512,eps=0.12040303950537279

Iteration 24:

[[-1.13809085e+01 1.90507314e+00 2.63484496e+00]

[-4.65400174e-01 1.00534054e+01 1.19934602e+00]

[-1.88367351e-07 -6.85901146e-06 -5.67249689e+00]]

n\_i=-11.380908523864436,n\_j=-11.268568784773139,eps=0.11233973909129702

Iteration 25:

[[-1.12863506e+01 2.77771921e+00 -2.68165066e+00]

[ 4.07247767e-01 9.95884678e+00 -1.09069839e+00]

[ 9.38080459e-08 -3.92536783e-06 -5.67249618e+00]]

n\_i=-11.286350602972377,n\_j=-11.380908523864436,eps=0.0945579208920595

Iteration 26:

[[-1.13734997e+01 2.00798068e+00 2.64057557e+00]

[-3.62491778e-01 1.00459964e+01 1.18668257e+00]

[-4.71170682e-08 -2.20369948e-06 -5.67249666e+00]]

n\_i=-11.373499740821636,n\_j=-11.286350602972377,eps=0.08714913784925926

Iteration 27:

[[-1.12993727e+01 2.68829142e+00 -2.67703835e+00]

[ 3.17819563e-01 9.97186915e+00 -1.10196709e+00]

[ 2.34875600e-08 -1.25841607e-06 -5.67249643e+00]]

n\_i=-11.299372725499063,n\_j=-11.373499740821636,eps=0.07412701532257238

Iteration 28:

[[-1.13670441e+01 2.08808816e+00 2.64499680e+00]

[-2.82384027e-01 1.00395407e+01 1.17679727e+00]

[-1.17865332e-08 -7.07918997e-07 -5.67249658e+00]]

n\_i=-11.36704412578786,n\_j=-11.299372725499063,eps=0.06767140028879659

Iteration 29:

[[-1.13090111e+01 2.61843605e+00 -2.67340659e+00]

[ 2.47964054e-01 9.98150761e+00 -1.11074777e+00]

[ 5.88002004e-09 -4.03562833e-07 -5.67249650e+00]]

n\_i=-11.309011109077504,n\_j=-11.36704412578786,eps=0.05803301671035577

Iteration 30:

[[-1.13615996e+01 2.15046213e+00 2.64841543e+00]

[-2.20009965e-01 1.00340961e+01 1.16908390e+00]

[-2.94865557e-09 -2.27383369e-07 -5.67249655e+00]]

n\_i=-11.36159956569662,n\_j=-11.309011109077504,eps=0.05258845661911593

Iteration 31:

[[-1.13162117e+01 2.56389686e+00 -2.67055339e+00]

[ 1.93424821e-01 9.98870825e+00 -1.11758990e+00]

[ 1.47189692e-09 -1.29450128e-07 -5.67249653e+00]]

n\_i=-11.316211722932453,n\_j=-11.36159956569662,eps=0.04538784276416763

Iteration 32:

[[-1.13571054e+01 2.19903891e+00 2.65106347e+00]

[-1.71433162e-01 1.00296020e+01 1.16306683e+00]

[-7.37712494e-10 -7.30271464e-08 -5.67249654e+00]]

n\_i=-11.357105430948842,n\_j=-11.316211722932453,eps=0.04089370801638914

Iteration 33:

[[-1.13216342e+01 2.52133139e+00 -2.66831577e+00]

[ 1.50859341e-01 9.99413074e+00 -1.12292176e+00]

[ 3.68420891e-10 -4.15308556e-08 -5.67249654e+00]]

n\_i=-11.321634201722249,n\_j=-11.357105430948842,eps=0.03547122922659263

Iteration 34:

[[-1.13534505e+01 2.23687759e+00 2.65311745e+00]

[-1.33594473e-01 1.00259470e+01 1.15837384e+00]

[-1.84574125e-10 -2.34513516e-08 -5.67249654e+00]]

n\_i=-11.353450493463345,n\_j=-11.321634201722249,eps=0.031816291741096236

Iteration 35:

[[-1.13257451e+01 2.48811998e+00 -2.66656328e+00]

[ 1.17647919e-01 9.99824166e+00 -1.12707699e+00]

[ 9.22119323e-11 -1.33259049e-08 -5.67249654e+00]]

n\_i=-11.325745119117544,n\_j=-11.353450493463345,eps=0.02770537434580156

Iteration 36:

[[-1.13505093e+01 2.26635655e+00 2.65471237e+00]

[-1.04115515e-01 1.00230059e+01 1.15471400e+00]

[-4.61818311e-11 -7.53037311e-09 -5.67249654e+00]]

n\_i=-11.350509332134052,n\_j=-11.325745119117544,eps=0.024764213016508663

Iteration 37:

[[-1.13288791e+01 2.46221229e+00 -2.66519216e+00]

[ 9.17402333e-02 1.00013756e+01 -1.13031542e+00]

[ 2.30787227e-11 -4.27627578e-09 -5.67249654e+00]]

n\_i=-11.328879072748737,n\_j=-11.350509332134052,eps=0.021630259385315043

Iteration 38:

[[-1.13481607e+01 2.28932569e+00 2.65595189e+00]

[-8.11463720e-02 1.00206573e+01 1.15186013e+00]

[-1.15553958e-11 -2.41789031e-09 -5.67249654e+00]]

n\_i=-11.348160728197453,n\_j=-11.328879072748737,eps=0.019281655448715185

Iteration 39:

[[-1.13312791e+01 2.44200525e+00 -2.66412029e+00]

[ 7.15331868e-02 1.00037757e+01 -1.13283947e+00]

[ 5.77593751e-12 -1.37235784e-09 -5.67249654e+00]]

n\_i=-11.331279126233106,n\_j=-11.348160728197453,eps=0.016881601964346515

Iteration 40:

[[-1.13462960e+01 2.30722446e+00 2.65691585e+00]

[-6.32475953e-02 1.00187925e+01 1.14963488e+00]

[-2.89140637e-12 -7.76307326e-10 -5.67249654e+00]]

n\_i=-11.346295975916885,n\_j=-11.331279126233106,eps=0.015016849683778588

Iteration 41:

[[-1.13331239e+01 2.42624633e+00 -2.66328288e+00]

[ 5.57742752e-02 1.00056205e+01 -1.13480682e+00]

[ 1.44551472e-12 -4.40447244e-10 -5.67249654e+00]]

n\_i=-11.33312392502956,n\_j=-11.346295975916885,eps=0.013172050887325426

Iteration 42:

[[-1.13448217e+01 2.32117333e+00 2.65766591e+00]

[-4.92987343e-02 1.00173183e+01 1.14789986e+00]

[-7.23505540e-13 -2.49236913e-10 -5.67249654e+00]]

n\_i=-11.344821717304145,n\_j=-11.33312392502956,eps=0.011697792274585339

Iteration 43:

[[-1.13345461e+01 2.41395750e+00 -2.66262895e+00]

[ 4.34854376e-02 1.00070427e+01 -1.13634029e+00]

[ 3.61754819e-13 -1.41364202e-10 -5.67249654e+00]]

n\_i=-11.334546131941526,n\_j=-11.344821717304145,eps=0.010275585362618855

Iteration 44:

[[-1.13436599e+01 2.33204468e+00 2.65824978e+00]

[-3.84273833e-02 1.00161565e+01 1.14654714e+00]

[-1.81042790e-13 -8.00159197e-11 -5.67249654e+00]]

n\_i=-11.34365994783107,n\_j=-11.334546131941526,eps=0.00911381588954363

# Лабораторная работа 2.1

Задание: Реализовать методы простой итерации и Ньютона решения нелинейных уравнений в виде программ, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения найти положительный корень нелинейного уравнения (начальное приближение определить графически). Проанализировать зависимость погрешности вычислений от количества итераций

Условие:



Исходный код:

import numpy as np

import math

from tabulate import tabulate

import matplotlib.pyplot as plt

def newton(fun, der, x, epsilon=0.001):

iter\_info = [[0, x, fun(x)]]

iter\_number = 1

while True:

new\_x = x - fun(x) / der(x)

iter\_info.append([iter\_number, new\_x, fun(new\_x)])

if np.abs(new\_x - x) < epsilon:

return iter\_info

x = new\_x

iter\_number = iter\_number + 1

def simple\_iteration\_prep(fi\_der, x0, x1, step = 0.001):

interval = np.arange(x0, x1, step)

return max(np.abs(fi\_der(interval)))

def simple\_iteration(fi, x, q, fun, epsilon = 0.001):

iter\_info = [[0, x, fi(x)]]

iter\_number = 1

while True:

new\_x = fi(x)

iter\_info.append([iter\_number, new\_x, fun(x)])

if (q / (1 - q)) \* np.abs(new\_x - x) < epsilon:

return iter\_info

x = new\_x

iter\_number = iter\_number + 1

#начало

def f(x):

return np.log(x+2) - np.power(x,2)

# функция фи, полученная заменой уравнения f(x) = 0 эквивалентным x = fi(x)

def fi(x):

return np.sqrt(np.log(x+2))

def fi\_der(x):

return 1 / (2 \* (x + 2) \* np.sqrt(np.log(x+2)))

def get\_derivative(fun, epsilon=0.0001):

return lambda x: (fun(x + epsilon) - fun(x)) / epsilon

t1 = np.arange(-1, 1, 0.0001)

der = get\_derivative(f)

der2 = get\_derivative(der)

plt.plot(t1, f(t1), label="f(x)")

plt.plot(t1, der(t1), label="f'(x)")

plt.plot(t1, der2(t1), label="f''(x)")

plt.legend()

plt.grid()

plt.show()

t2 = np.arange(-np.pi / 4, 1, 0.01)

plt.plot(t2, fi(t2), label="phi(x)")

plt.plot(t2, get\_derivative(fi)(t2), label="phi'(x)")

plt.xticks(np.arange(-1, 1, 0.1))

plt.axis([-1, 1, -2, 2])

plt.legend()

plt.grid()

plt.show()

first\_root\_interval = (-0.75, 0)

second\_root\_interval = (0.5, 0.8)

epsilon = 0.001

newton\_roots = [newton(f, get\_derivative(f, epsilon), -0.5, epsilon), newton(f, get\_derivative(f, epsilon), 0.75, epsilon)]

q = simple\_iteration\_prep(fi\_der, \*second\_root\_interval)

simple\_iteration\_root = simple\_iteration(fi, -0.3, q, f, epsilon)

print("Newton roots")

for info in newton\_roots:

print(tabulate(info, headers=['iteration', 'x', 'f(x)']))

print("Simple iterations")

print("Chosen interval: [{},{}]".format(\*second\_root\_interval))

print("q: {}".format(q))

print(tabulate(simple\_iteration\_root, headers=['iteration', 'x', 'f(x)']))

Результат:

Newton roots

iteration x f(x)

----------- --------- ------------

0 -0.5 0.155465

1 -0.593348 -0.0108485

2 -0.587627 -3.38087e-05

3 -0.587609 2.20609e-08

iteration x f(x)

----------- ------- ------------

0 0.75 0.449101

1 1.14484 -0.164892

2 1.06125 -0.0074339

3 1.05712 -2.23872e-05

4 1.0571 -1.33543e-08

Simple iterations

Chosen interval: [0.5,0.8]

q: 0.20893603499035696

iteration x f(x)

----------- --------- ----------

0 -0.3 0.728442

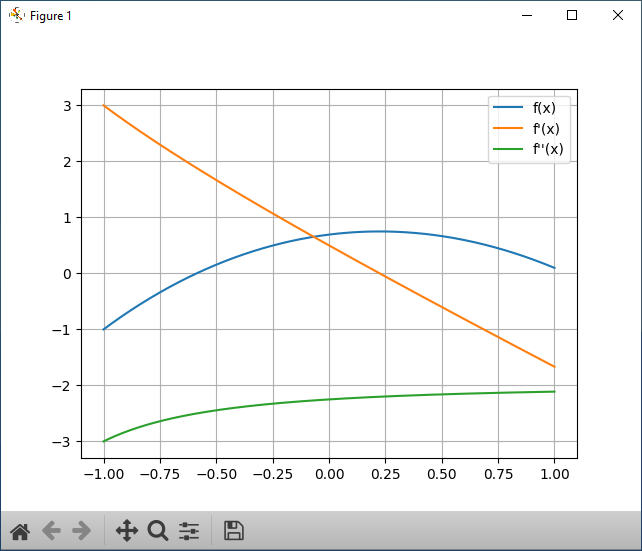
1 0.728442 0.440628

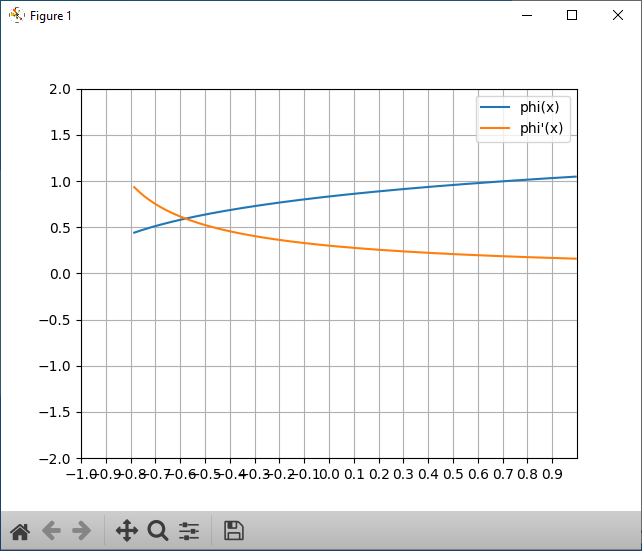
2 1.00186 0.473103

3 1.04844 0.0955025

4 1.05576 0.0153977

5 1.0569 0.00239756

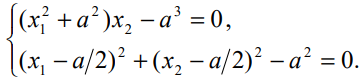




**Лабораторная работа 2.2**

Задание: Реализовать методы простой итерации и Ньютона решения систем нелинейных уравнений в виде программного кода, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения решить систему нелинейных уравнений (при наличии нескольких решений найти то из них, в котором значения неизвестных являются положительными); начальное приближение определить графически. Проанализировать зависимость погрешности вычислений от количества итераций.

Условие:



Исходный код:

import numpy as np

import math

import matplotlib.pyplot as plt

from scipy.linalg import lu

def lu\_decomposition(other):

n, m = other.shape

L = np.zeros(other.shape)

U = np.copy(other)

P = np.eye(n)

odd = False

for i in range(0,n):

index = i

for j in range(i,n):

if np.abs(U[j, i]) > np.abs(U[index, i]):

index = j

if i != index:

L[[i, index]] = L[[index, i]]

U[[i, index]] = U[[index, i]]

P[[i, index]] = P[[index, i]]

odd = not odd

L[i, i] = 1

for j in range(i + 1, n):

L[j][i] = U[j][i] / U[i][i]

U[j][i] = 0

for j in range(i + 1, n):

for k in range(i + 1, n):

U[j][k] = U[j][k] - U[i][k] \* L[j][i]

return L, U, P, odd

def solve\_eq(A, b):

L,U,P,odd = lu\_decomposition(A)

n, m = A.shape

b = P.dot(b)

z = np.empty(n, dtype=float)

z[0] = b[0]

for i in range(1, n):

sum = 0

for j in range(0, i):

sum += L[i,j] \* z[j]

z[i] = b[i] - sum

x = np.empty(n, dtype=float)

x[n - 1] = z[n - 1] / U[n - 1, n - 1]

for i in range(n - 1, -1, -1):

sum = 0

for j in range(i + 1, n):

sum += U[i, j] \* x[j]

x[i] = (z[i] - sum) / U[i,i]

return x

def f1(x):

#return np.power(x[0], 2) - 2 \* np.log10(x[1]) - 1

return (np.power(x[0], 2) + 9) \* x[1] - 27

def f2(x):

#return np.power(x[0], 2) - 2 \* x[0] \* x[1] + 2

return np.power((x[0] - 3 / 2), 2) + np.power((x[1] - 3 / 2), 2) - 9

def construct\_equiv\_function(f, var, lbd, coef):

return lambda x: (coef \* x[var] - lbd \* f(x)) / coef

def f1\_equiv(x):

#return np.sqrt(2\*np.log10(x[1]) + 1)

return np.sqrt(27 / x[1] - 9)

def f2\_equiv(x):

#return (np.power(x[0], 2) + 2)/(2 \* x[0])

return np.sqrt(9 - np.power(x[0] - 3 / 2, 2)) + 3 / 2

#производная f1\_equiv по x1

def df1edx1(x):

return 0

#производная f1\_equiv по x2

def df1edx2(x):

return - 27 / (2 \* np.power(x[1], 2) \* np.sqrt((27 - 9 \* x[1]) / (x[1])))

#производная f2\_equiv по x1

def df2edx1(x):

return (2 \* x[0] - 3) \* np.sqrt(- 4 \* np.power(x[0], 2) + 12 \* x[0] + 27) / (4 \* np.power(x[0], 2) - 12 \* x[0] - 27)

#производная f2\_equiv по x2

def df2edx2(x):

return 0

def derivative(fun, var\_num, epsilon=0.001):

def res(x):

eps\_vector = [(0 if i != var\_num else epsilon) for i in range(0, len(x))]

return (fun(x + eps\_vector) - fun(x)) / epsilon

return res

def simple\_iterations(functions, x, q, epsilon=0.001):

x = np.array(x, dtype=float)

while True:

new\_x = np.array([f(x) for f in functions], dtype=float)

if (q / (1 - q)) \* np.linalg.norm(new\_x - x) < epsilon:

return new\_x

x = new\_x

def newton(functions, x, epsilon=0.001):

jacobi\_matrix = [[derivative(functions[i], j, epsilon) for j in range(0, len(x))] for i in range(0, len(functions))]

while True:

der\_values = [[der(x) for der in ders] for ders in jacobi\_matrix]

b = [-f(x) for f in functions]

dx = solve\_eq(np.array(der\_values, dtype=float), np.array(b, dtype=float))

new\_x = x + dx

if (np.linalg.norm(x - new\_x) < epsilon):

return new\_x

x = new\_x

def make\_mesh(interval):

indices = np.zeros((len(interval),), dtype=np.int64)

dims = [len(part) for part in interval]

result\_points = []

for k in range(0, np.int64(np.prod([len(x) for x in interval]))):

result\_points.append([interval[i, indices[i]] for i in range(0, len(indices))])

for i in range(0, len(indices)):

indices[i] += 1

if indices[i] == dims[i]:

indices[i] = 0

else:

break

return result\_points

def test\_function(f, nvar, start, end, step=0.01, ders=None):

if ders is None:

ders = [derivative(f, i) for i in range(0, nvar)]

interval = np.array([np.arange(x0, x1, step) for x0, x1 in zip(start, end)])

mesh = make\_mesh(interval)

a = [[np.abs(der(x)) for x in mesh] for der in ders]

b = np.sum(a, axis=0)

return max(b)

start\_x = np.array([1.1, 1.4])

g\_wide = 0.2

x1\_int = [1.14 - g\_wide / 2, 1.14 + g\_wide / 2]

x2\_int = [1.4 - g\_wide / 2, 1.4 + g\_wide / 2]

q = max([test\_function(f1\_equiv, 2, x1\_int, x2\_int, 0.001, [df1edx1,df1edx2]),

test\_function(f2\_equiv, 2, x1\_int, x2\_int, 0.001, [df2edx1,df2edx2])])

print("Q in simple iteration method: ", q)

epsilon = 0.001

n\_root = newton([f1, f2], start\_x, epsilon)

si\_root = simple\_iterations([f1\_equiv, f2\_equiv], start\_x, q, epsilon)

print(n\_root, f1(n\_root), f2(n\_root))

print(si\_root, f1(si\_root), f2(si\_root))

Результаты:

Q in simple iteration method: 2.4565389363230548

[-1.32469087 2.51050556] -2.3830786233247636e-08 -1.3684712030226365e-08

[3.2071349 4.47321375] 59.269122311085695 2.7543095768670103

# Лабораторная работа 3.1

Задание: Используя таблицу значений функции , вычисленных в точках , построить интерполяционные многочлены Лагранжа и Ньютона, проходящие через точки . Вычислить значение погрешности интерполяции в точке

Условие:



Исходный код:

import numpy as np

import math

import matplotlib.pyplot as plt

def f(x):

return np.cos(x)

points1 = np.array([0, np.pi / 6, 2 \* np.pi / 6, 3 \* np.pi / 6])

points2 = np.array([0, np.pi / 6, 5 \* np.pi / 12, np.pi / 2])

def Lagrange(points, values=None, function=None):

if values is None and function is None:

raise ValueError("function needs function values or function itself")

if values is None:

values = function(points)

if len(values) != len(points):

raise ValueError("length is wrong")

values = values[:]

points = points[:]

n = len(values)

return lambda x: np.sum(

[values[i] \* np.prod([1 if j == i else ((x - points[j]) / (points[i] - points[j])) for j in range(0, n)]) for i

in range(0, n)], axis = 0)

def divided\_diff(points, function):

if (len(points) == 1):

return function(points[0])

if (len(points) == 2):

return (function(points[0]) - function(points[1])) / (points[0] - points[1])

n = len(points)

return (divided\_diff(points[:n-1], function) - divided\_diff(points[1:], function)) / (points[0] - points[n - 1])

def Newton(points, function):

points = points[:]

n = len(points)

cur\_arr = []

divided\_sums\_counted = []

for i in range(0, n):

cur\_arr.append(points[i])

divided\_sums\_counted.append(divided\_diff(cur\_arr, function))

def result(x):

cur\_prod = 1

res\_sum = 0

for i in range(0, n):

res\_sum += divided\_sums\_counted[i] \* cur\_prod

cur\_prod \*= (x - points[i])

return res\_sum

return result

interval = np.arange(-3, 3, np.pi / 4)

plt.figure()

plt.subplot(221)

plt.plot(interval, [Lagrange(points1, function=f)(x) for x in interval])

plt.plot(interval, f(interval))

plt.title("Lagrange 1")

plt.subplot(222)

plt.plot(interval, [Lagrange(points2, function=f)(x) for x in interval])

plt.plot(interval, f(interval))

plt.title("Lagrange 2")

plt.subplot(223)

plt.plot(interval, [Newton(points1, function=f)(x) for x in interval])

plt.plot(interval, f(interval))

plt.title("Newton 1")

plt.subplot(224)

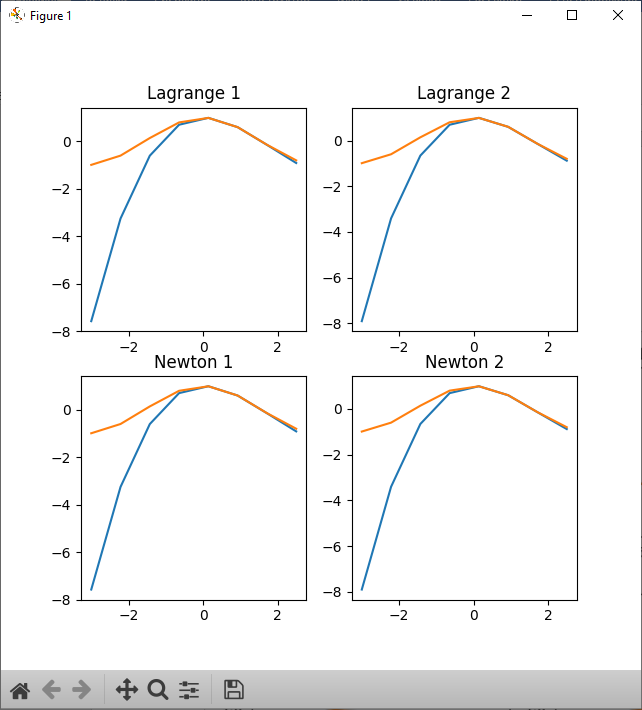
plt.plot(interval, [Newton(points2, function=f)(x) for x in interval])

plt.plot(interval, f(interval))

plt.title("Newton 2")

plt.show()

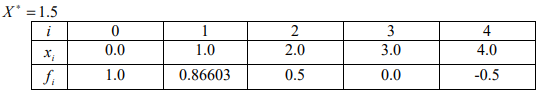
Результат:



# Лабораторная работа 3.2

Задание: Построить кубический сплайн для функции, заданной в узлах интерполяции, предполагая, что сплайн имеет нулевую кривизну при и . Вычислить значение функции в точке

Условие:



Исходный код:

import numpy as np

import math

import matplotlib.pyplot as plt

def sweep\_method(matrix, d):

P = np.zeros((len(d),))

Q = np.zeros((len(d),))

n = len(d)

M = lambda i, j, inc: matrix[i+inc, j+inc]

a = lambda i: M(0, -1, i)

b = lambda i: M(0, 0, i)

c = lambda i: M(0, 1, i)

P[0] = -c(0) / b(0)

Q[0] = d[0] / b(0)

for i in range(1, n - 1):

P[i] = -c(i) / (b(i) + a(i) \* P[i - 1])

Q[i] = (d[i] - a(i) \* Q[i - 1]) / (b(i) + a(i) \* P[i - 1])

P[n - 1] = 0

Q[n - 1] = (d[n - 1] - a(n - 1) \* Q[n - 2]) / (b(n - 1) + a(n - 1) \* P[n - 2])

x = np.zeros(n)

x[n - 1] = Q[-1]

for i in range(n - 2, -1, -1):

x[i] = P[i] \* x[i + 1] + Q[i]

return x

def make\_spline(a,b,c,d, points):

a = np.copy(a)

b = np.copy(b)

c = np.copy(c)

d = np.copy(d)

points = np.copy(points)

def result(x):

i = np.searchsorted(points, x)

if i == 0 and x == points[0]:

i += 1

if i == 0 or i == len(points):

raise ValueError("This point {} doesnt belong to spline".format(x))

i -= 1

return a[i] + b[i] \* (x - points[i]) + c[i] \* np.power((x - points[i]),2) + d[i] \* np.power((x - points[i]), 3)

return result

def find\_coefs(points, values):

n = len(points) - 1

h = [points[i] - points[i - 1] for i in range(1, n + 1)]

A = np.zeros((n-1,n-1), dtype=np.float32)

r = np.array([3 \* (((values[i + 1] - values[i]) / h[i]) - ((values[i] - values[i - 1]) / h[i - 1])) for i in range(1, n)])

last = n - 2

A[0,0] = 2 \* (h[0] + h[1])

A[0,1] = h[1]

A[last, last - 1] = h[n - 2]

A[last, last] = 2 \* (h[n - 2] + h[n - 1])

for i in range(1, last):

A[i, i - 1] = h[i]

A[i, i] = 2 \* (h[i] + h[i + 1])

A[i, i + 1] = h[i]

c = sweep\_method(A, r)

c = np.insert(c,0,0)

a = [values[i] for i in range(0, n)]

b = [(values[i] - values[i - 1]) / h[i - 1] - (1/3) \* h[i - 1] \* (c[i] + 2 \* c[i - 1]) for i in range(1, n)]

b.append((values[-1] - values[-2]) / h[-1] - (2/3) \* h[-1] \* c[-1])

d = [(c[i + 1] - c[i]) / (3 \* h[i]) for i in range(0, n - 1)]

d.append(-(c[-1] / (3 \* h[-1])))

return a,b,c,d

points = [0, 1.0, 2.0, 3.0, 4.0]

values = [1.0, 0.86603, 0.5, 0, -0.5]

a,b,c,d = find\_coefs(points, values)

for a0, b0, c0, d0 in zip(a,b,c,d):

print("{}x^3 + {}x^2 + {}x + {}".format(a0, b0, c0, d0))

spline\_function = make\_spline(\*find\_coefs(points, values), points)

interval = np.linspace(0.0, 4.0, 120, endpoint=False)

plt.plot(points, values, "ro")

plt.plot(interval, [spline\_function(x) for x in interval])

plt.show()

Результат:

1.0x^3 + -0.0813803571428572x^2 + 0.0x + -0.05258964285714284

0.86603x^3 + -0.2391492857142857x^2 + -0.15776892857142852x + 0.030888214285714255

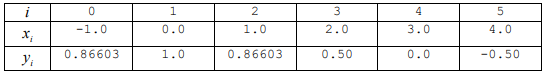
0.5x^3 + -0.4620225x^2 + -0.06510428571428575x + 0.02712678571428573

0x^3 + -0.5108507142857143x^2 + 0.01627607142857144x + -0.005425357142857146

# Лабораторная работа 3.3

Задание: Для таблично заданной функции путем решения нормальной системы МНК найти приближающие многочлены a) 1-ой и б) 2-ой степени. Для каждого из приближающих многочленов вычислить сумму квадратов ошибок. Построить графики приближаемой функции и приближающих многочленов

Условие:



Исходный код:

import numpy as np

import math

import matplotlib.pyplot as plt

def lu\_decomposition(other):

n, m = other.shape

L = np.zeros(other.shape)

U = np.copy(other)

P = np.eye(n)

odd = False

for i in range(0, n):

index = i

for j in range(i, n):

if np.abs(U[j, i]) > np.abs(U[index, i]):

index = j

if i != index:

L[[i, index]] = L[[index, i]]

U[[i, index]] = U[[index, i]]

P[[i, index]] = P[[index, i]]

odd = not odd

L[i, i] = 1

for j in range(i + 1, n):

L[j][i] = U[j][i] / U[i][i]

U[j][i] = 0

for j in range(i + 1, n):

for k in range(i + 1, n):

U[j][k] = U[j][k] - U[i][k] \* L[j][i]

return L, U, P, odd

def solve\_eq(A, b):

L, U, P, odd = lu\_decomposition(A)

n, m = A.shape

b = P.dot(b)

z = np.empty(n, dtype=float)

z[0] = b[0]

for i in range(1, n):

sum = 0

for j in range(0, i):

sum += L[i, j] \* z[j]

z[i] = b[i] - sum

x = np.empty(n, dtype=float)

x[n - 1] = z[n - 1] / U[n - 1, n - 1]

for i in range(n - 1, -1, -1):

sum = 0

for j in range(i + 1, n):

sum += U[i, j] \* x[j]

x[i] = (z[i] - sum) / U[i, i]

return x

#метод наименьших квадратов

x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0])

y = np.array([0.86603, 1.0, 0.86603, 0.50, 0.0, -0.50])

def count\_mse(approx, x, y):

return np.sum(np.square(approx(x) - y))

def LSM(x, y, polynom\_power):

n = polynom\_power + 1

m = len(x)

A = np.zeros((n,n), dtype=np.float32)

A[-1] = np.array([np.sum([np.power(x\_cur,i) for x\_cur in x]) for i in range(n - 1, -1, -1)])

for i in range(n - 2, -1, -1):

A[i,1:n] = A[i + 1, 0:n-1]

A[i, 0] = np.sum([np.power(x\_cur,2\*n - i - 2) for x\_cur in x])

b = np.array([np.sum([np.power(x[j], i) \* y[j] for j in range(0, m)]) for i in range(0, n)], dtype=np.float32)

b = np.flip(b)

coefs = solve\_eq(A, b)

return lambda arg: np.sum([coefs[i] \* np.power(arg, len(coefs) - i - 1) for i in range(0, len(coefs))], axis=0)

linear\_approximator = LSM(x,y,1)

square\_approximator = LSM(x,y,2)

cubic\_approximator = LSM(x,y,3)

interval = np.arange(-1, 4, 0.01)

plt.plot(x, y, 'ro', label="points")

plt.plot(interval, linear\_approximator(interval), label="linear approximator")

plt.plot(interval, square\_approximator(interval), label="square approximator")

plt.plot(interval, cubic\_approximator(interval), label="cubic approximator")

print("Mean squared errors:\nlinear {}\nsquare {}\ncubic {}".format(count\_mse(linear\_approximator, x, y),

count\_mse(square\_approximator, x, y),

count\_mse(cubic\_approximator, x, y)))

plt.legend()

plt.show()

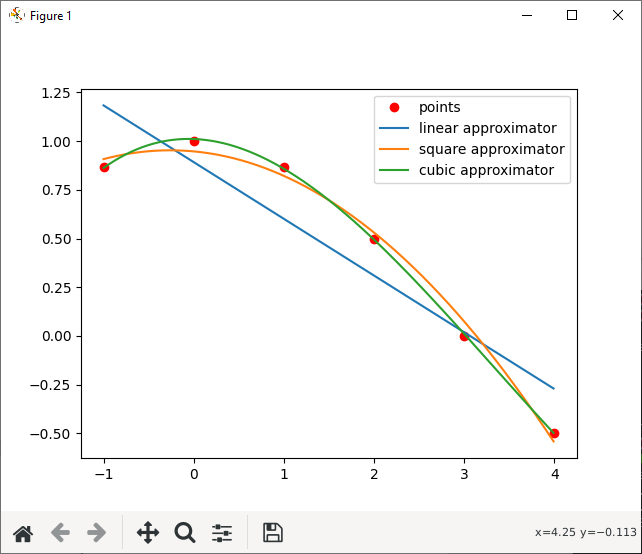
Результат:

Mean squared errors:

linear 0.27081794892762573

square 0.015178925583584685

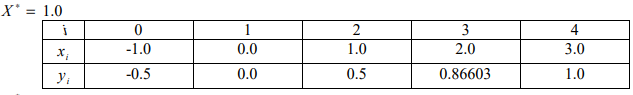
cubic 0.00034638135636763277



# Лабораторная работа 3.4

Задание: Вычислить первую и вторую производную от таблично заданной функции в точке

Условие:



Исходный код:

import numpy as np

import math

import matplotlib.pyplot as plt

x = np.array([-1.0, 0.0, 1.0, 2.0, 3.0])

y = np.array([-0.5, 0.0, 0.5, 0.86603, 1.0])

def get\_polynom(x,y):

x = np.copy(x)

y = np.copy(y)

def polynom(arg):

i = np.searchsorted(x, arg)

if (arg == x[0]):

i = 1

i -= 1

return y[i] \

+ (arg - x[i]) \* (y[i + 1] - y[i]) / (x[i + 1] - x[i]) \

+ (arg - x[i]) \* (arg - x[i + 1]) \* ((y[i + 2] - y[i + 1])/(x[i + 2] - x[i + 1]) - (y[i + 1] - y[i])/(x[i + 1] - x[i])) / (x[i + 2] - x[i])

return polynom

def get\_polynom\_derivative(x,y):

x = np.copy(x)

y = np.copy(y)

def polynom(arg):

i = np.searchsorted(x, arg)

if (arg == x[0]):

i = 1

i -= 1

return (y[i + 1] - y[i])/(x[i + 1] - x[i]) + \

((y[i + 2] - y[i + 1])/(x[i + 2] - x[i + 1]) - (y[i + 1] - y[i])/(x[i + 1] - x[i])) \* (2 \* arg - x[i] - x[i + 1]) / (x[i + 2] - x[i])

return polynom

def get\_polynom\_derivative2(x,y):

x = np.copy(x)

y = np.copy(y)

def polynom(arg):

i = np.searchsorted(x, arg)

if (arg == x[0]):

i = 1

i -= 1

return 2 \* ((y[i + 2] - y[i + 1])/(x[i + 2] - x[i + 1]) - (y[i + 1] - y[i])/(x[i + 1] - x[i])) / (x[i + 2] - x[i])

return polynom

def approx\_polynom(x,y):

x = np.copy(x)

y = np.copy(y)

return get\_polynom(x,y), get\_polynom\_derivative(x,y), get\_polynom\_derivative2(x,y)

f, fd, fd2 = approx\_polynom(x,y)

print("f(x) = {}, f'(x) = {}, f''(x) = {}".format(f(0.2), fd(0.2), fd2(0.2)))

Результат:

f(x) = 0.11071760000000001, f'(x) = 0.540191, f''(x) = -0.13397000000000003

# Лабораторная работа 3.5

Задание: Вычислить определенный интеграл , методами прямоугольников, трапеций, Симпсона с шагами . Оценить погрешность вычислений, используя Метод Рунге-Ромберга:

Условие:



Исходный код:

import numpy as np

import math

import matplotlib.pyplot as plt

def f(x):

return x / np.power(3 \*x + 4, 2)

def rect\_method(x, f):

return np.sum([f((x[i + 1] + x[i]) / 2) \* (x[i + 1] - x[i]) for i in range(0,len(x) - 1)])

def trapeze\_method(x, f):

y = f(x)

return np.sum([(y[i + 1] + y[i]) \* (x[i + 1] - x[i]) for i in range(0, len(x) - 1)]) / 2.

def simpson\_method(x, f):

y = f(x)

return np.sum([(f(x[i]) + 4 \* f((x[i] + x[i + 1]) / 2) + f(x[i + 1])) \* (x[i + 1] - x[i]) for i in range(0, len(x) - 1)]) / 6.

rect\_method.runge\_coef = 2

trapeze\_method.runge\_coef = 2

simpson\_method.runge\_coef = 4

def integrate(method, function, x0, xk, h):

res = method(np.arange(x0, xk, h), f)

return {"result": res, "h": h}

def print\_results(name, result\_list, err):

print("method: {}, results: {}, errors: {}".format(name,result\_list, err))

def test\_method(method, function, x0, xk, h):

result\_list = [integrate(method, function, x0, xk, h\_cur) for h\_cur in h]

k = result\_list[0]["h"] / result\_list[1]["h"]

err = np.abs(result\_list[0]["result"] - result\_list[1]["result"]) / (np.power(k, method.runge\_coef) - 1)

return result\_list, err

x0 = 0

xk = 4

h = [1.0, 0.5, 0.1, 0.01]

print("h: {}".format(h))

print\_results("rectangle method", \*test\_method(rect\_method, f, x0, xk, h))

print\_results("trapeze method", \*test\_method(trapeze\_method, f, x0, xk, h))

print\_results("simpson method", \*test\_method(simpson\_method, f, x0, xk, h))

Результат:

h: [1.0, 0.5, 0.1, 0.01]

method: rectangle method, results: [{'result': 0.056193762976994854, 'h': 1.0}, {'result': 0.06326531814994407, 'h': 0.5}, {'result': 0.06915384100187219, 'h': 0.1}, {'result': 0.07054329430273676, 'h': 0.01}], errors: 0.002357185057649739

method: trapeze method, results: [{'result': 0.04928390291027653, 'h': 1.0}, {'result': 0.06133841501344551, 'h': 0.5}, {'result': 0.06907339287742782, 'h': 0.1}, {'result': 0.0705424886093769, 'h': 0.01}], errors: 0.004018170701056326

method: simpson method, results: [{'result': 0.05389047628808875, 'h': 1.0}, {'result': 0.06262301710444455, 'h': 0.5}, {'result': 0.06912702496039073, 'h': 0.1}, {'result': 0.07054302573828346, 'h': 0.01}], errors: 0.0005821693877570538

# Лабораторная работа 4.1

Задание: Реализовать методы Эйлера, Рунге-Кутты и Адамса 4-го порядка в виде программ, задавая в качестве входных данных шаг сетки . С использованием разработанного программного обеспечения решить задачу Коши для ОДУ 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

Условие:



Исходный код:

import numpy as np

import math

from tabulate import tabulate

import matplotlib.pyplot as plt

def precise\_solution(x):

return x \* np.sin(x) + np.cos(x)

def f1(x, y1, y2):

return y2

def f2(x, y1, y2):

return 2 \* np.cos(x) - y1

class Euler:

def \_\_init\_\_(self, functions, x0, start\_conditions):

self.functions = functions

self.x0 = x0

self.start\_conditions = start\_conditions

self.label = "Euler"

self.last\_points = None

self.last\_values = None

self.p = 1

def integrate(self, x\_end, h):

points = [self.x0]

values = [self.start\_conditions] # type: list

while points[-1] <= x\_end:

cur\_x = points[-1]

cur\_condition = values[-1]

new\_x = cur\_x + h

new\_condition = [cur\_condition[i] + h \* self.functions[i](cur\_x, \*cur\_condition) for i in range(0, len(cur\_condition))]

points.append(new\_x)

values.append(new\_condition)

self.last\_points = np.array(points)

self.last\_values = np.array(values)

return np.array(points), np.array(values)

class EulerKoshi:

def \_\_init\_\_(self, functions, x0, start\_conditions):

self.functions = functions

self.x0 = x0

self.start\_conditions = start\_conditions

self.label = "Euler Koshi"

self.last\_points = None

self.last\_values = None

self.p = 2

def integrate(self, x\_end, h):

points = [self.x0]

values = [self.start\_conditions] # type: list

while points[-1] <= x\_end:

cur\_x = points[-1]

cur\_condition = values[-1]

new\_x = cur\_x + h

new\_condition = [cur\_condition[i] + h \* self.functions[i](cur\_x, \*cur\_condition) for i in range(0, len(cur\_condition))]

new\_condition\_corrected = [cur\_condition[i] + h \* (self.functions[i](cur\_x, \*cur\_condition) + self.functions[i](new\_x, \*new\_condition)) / 2 for i in range(0, len(cur\_condition))]

points.append(new\_x)

values.append(new\_condition\_corrected)

self.last\_points = np.array(points)

self.last\_values = np.array(values)

return np.array(points), np.array(values)

class EulerAdvanced:

def \_\_init\_\_(self, functions, x0, start\_conditions):

self.functions = functions

self.x0 = x0

self.start\_conditions = start\_conditions

self.last\_points = None

self.last\_values = None

self.label = "Euler advanced"

self.p = 2

def integrate(self, x\_end, h):

points = [self.x0]

values = [self.start\_conditions] # type: list

while points[-1] <= x\_end:

cur\_x = points[-1]

cur\_condition = values[-1]

new\_x = cur\_x + h

half\_condition = [cur\_condition[i] + (h / 2) \* self.functions[i](cur\_x, \*cur\_condition) for i in range(0, len(cur\_condition))]

new\_condition = [cur\_condition[i] + h \* self.functions[i](cur\_x + h / 2, \*half\_condition) for i in range(0, len(cur\_condition))]

points.append(new\_x)

values.append(new\_condition)

self.last\_points = np.array(points)

self.last\_values = np.array(values)

return np.array(points), np.array(values)

class RungeKutta:

def \_\_init\_\_(self, functions, x0, start\_conditions):

self.functions = functions

self.x0 = x0

self.start\_conditions = start\_conditions

self.last\_points = None

self.last\_values = None

self.label = "Runge Kutta 4-th"

self.p = 4

def integrate(self, x\_end, h):

points = np.array([self.x0])

values = np.array([self.start\_conditions])

while points[-1] <= x\_end:

cur\_x = points[-1]

cur\_condition = values[-1]

k1 = np.array([h \* self.functions[i](cur\_x, \*cur\_condition) for i in range(0, len(cur\_condition))])

k2 = np.array([h \* self.functions[i](cur\_x + h / 2, \*(cur\_condition + (k1 / 2))) for i in range(0, len(cur\_condition))])

k3 = np.array([h \* self.functions[i](cur\_x + h / 2, \*(cur\_condition + (k2 / 2))) for i in range(0, len(cur\_condition))])

k4 = np.array([h \* self.functions[i](cur\_x + h, \*(cur\_condition + k3)) for i in range(0, len(cur\_condition))])

new\_x = cur\_x + h

new\_condition = cur\_condition + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6

num = k2 - k3

den = k1 - k2

points = np.append(points, [new\_x], axis=0)

values = np.append(values, [new\_condition], axis=0)

self.last\_points = np.array(points)

self.last\_values = np.array(values)

return np.array(points), np.array(values)

class Adams:

def \_\_init\_\_(self, functions, x0, start\_conditions):

self.functions = functions

self.x0 = x0

self.start\_conditions = start\_conditions

self.last\_points = None

self.last\_values = None

self.label = "Adams 4-th"

self.p = 4

def integrate(self, x\_end, h):

points = np.array(self.x0)

values = np.array(self.start\_conditions)

while points[-1] <= x\_end:

cur\_points = points[-4:]

cur\_values = values[-4:]

f = np.array([[self.functions[i](cur\_points[j], \*cur\_values[j]) for i in range(0, len(self.functions))] for j in range(0,len(cur\_points))])

new\_values = np.array([cur\_values[-1,i] + (h/24)\*(55 \* f[-1,i] - 59 \* f[-2,i] + 37 \* f[-3,i] - 9 \* f[-4,i]) for i in range(0, len(self.functions))])

new\_x = points[-1] + h

points = np.append(points, [new\_x], axis=0)

values = np.append(values, [new\_values], axis=0)

self.last\_points = points

self.last\_values = values

return points, values

fig, ax = plt.subplots(2)

ax[0].set\_ylim([-1, 4])

ax[1].set\_ylim([-1, 4])

def plot\_solved(solver):

for i in range(0, solver.last\_values.shape[-1]):

ax[i].plot(solver.last\_points, solver.last\_values[:, i], label=solver.label)

x0 = 0

x1 = 1

start\_conditions = [1.0, 0.0]

h = 0.1

x\_interval = np.arange(x0, x1 + h/2, h)

ax[0].plot(x\_interval, precise\_solution(x\_interval), label="precise solution")

solvers = [Euler, EulerKoshi, RungeKutta]

adams\_start\_x = None

adams\_start\_y = None

solver\_instances = []

for solver\_class in solvers:

solver = solver\_class([f1, f2], x0, start\_conditions)

points, values = solver.integrate(x1, h)

if (solver.\_\_class\_\_.\_\_name\_\_ == "RungeKutta"):

adams\_start\_x = points[:3]

adams\_start\_y = values[:3]

solver\_instances.append(solver)

plot\_solved(solver)

x0 = np.append([x0], adams\_start\_x, axis=0)

start\_conditions = np.append([start\_conditions], adams\_start\_y, axis=0)

adams = Adams([f1,f2], x0, start\_conditions)

adams.integrate(x1,h)

solver\_instances.append(adams)

plot\_solved(adams)

ax[0].legend()

plt.show()

for solver in solver\_instances:

points1, values1 = solver.integrate(x1, h / 2)

points1 = points1[::2]

values1 = values1[::2]

points2, values2 = solver.integrate(x1, h)

if (values1.shape[0] > values2.shape[0]):

values1 = np.resize(values1, values2.shape)

else:

values2 = np.resize(values2, values1.shape)

error = np.linalg.norm(np.abs(values1[:, 0] - values2[:, 0]) / (np.power(2, solver.p) - 1))

print("{} error: {}".format(solver.label, error))

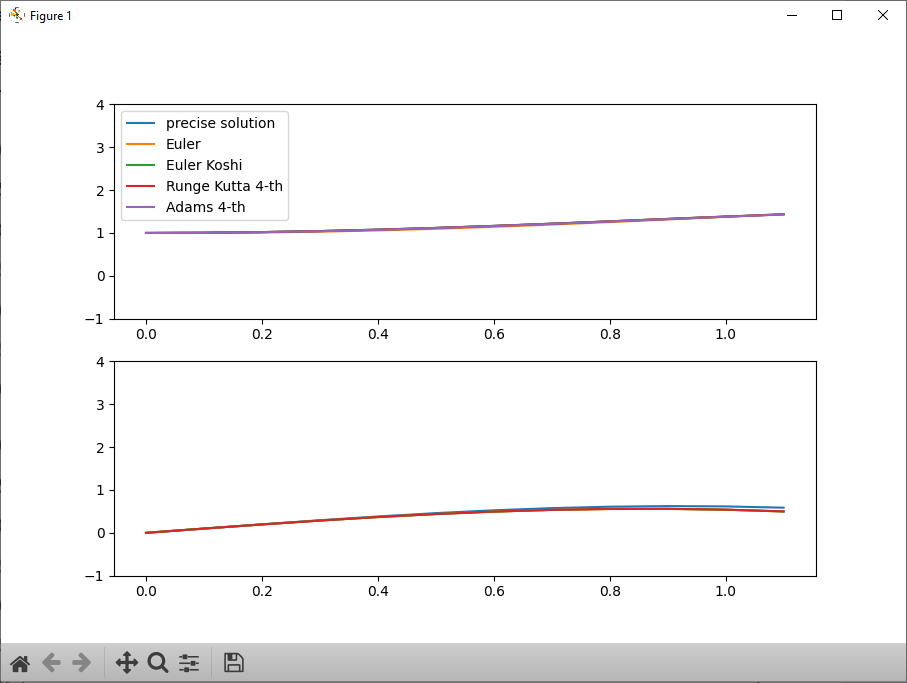
Результат:

Euler error: 0.025678664889541354

Euler Koshi error: 0.0007711337576050863

Runge Kutta 4-th error: 1.2030891457124265e-07

Adams 4-th error: 0.012622080111985774



# Лабораторная работа 4.2

Задание: Реализовать метод стрельбы и конечно-разностный метод решения краевой задачи для ОДУ в виде программ. С использованием разработанного программного обеспечения решить краевую задачу для обыкновенного дифференциального уравнения 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

Условие:



Исходный код:

import numpy as np

import math

from tabulate import tabulate

import matplotlib.pyplot as plt

class RungeKutta:

def \_\_init\_\_(self, functions, x0, start\_conditions):

self.functions = functions

self.x0 = x0

self.start\_conditions = start\_conditions

self.last\_points = None

self.last\_values = None

self.label = "Runge Kutta 4-th"

self.p = 4

def integrate(self, x\_end, h):

points = np.array([self.x0])

values = np.array([self.start\_conditions])

while points[-1] <= x\_end:

cur\_x = points[-1]

cur\_condition = values[-1]

k1 = np.array([h \* self.functions[i](cur\_x, \*cur\_condition) for i in range(0, len(cur\_condition))])

k2 = np.array([h \* self.functions[i](cur\_x + h / 2, \*(cur\_condition + (k1 / 2))) for i in range(0, len(cur\_condition))])

k3 = np.array([h \* self.functions[i](cur\_x + h / 2, \*(cur\_condition + (k2 / 2))) for i in range(0, len(cur\_condition))])

k4 = np.array([h \* self.functions[i](cur\_x + h, \*(cur\_condition + k3)) for i in range(0, len(cur\_condition))])

new\_x = cur\_x + h

new\_condition = cur\_condition + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6

num = k2 - k3

den = k1 - k2

points = np.append(points, [new\_x], axis=0)

values = np.append(values, [new\_condition], axis=0)

self.last\_points = np.array(points)

self.last\_values = np.array(values)

return np.array(points), np.array(values)

def sweep\_method(matrix, d):

P = np.zeros((len(d),))

Q = np.zeros((len(d),))

n = len(d)

M = lambda i, j, inc: matrix[i+inc, j+inc]

a = lambda i: M(0, -1, i)

b = lambda i: M(0, 0, i)

c = lambda i: M(0, 1, i)

P[0] = -c(0) / b(0)

Q[0] = d[0] / b(0)

for i in range(1, n - 1):

P[i] = -c(i) / (b(i) + a(i) \* P[i - 1])

Q[i] = (d[i] - a(i) \* Q[i - 1]) / (b(i) + a(i) \* P[i - 1])

P[n - 1] = 0

Q[n - 1] = (d[n - 1] - a(n - 1) \* Q[n - 2]) / (b(n - 1) + a(n - 1) \* P[n - 2])

x = np.zeros(n)

x[n - 1] = Q[-1]

for i in range(n - 2, -1, -1):

x[i] = P[i] \* x[i + 1] + Q[i]

return x

def increase\_matrix\_size(matrix, begin):

shape = matrix.shape

shape = tuple([i + 1 for i in shape])

temp = np.zeros(shape)

if begin:

temp[1:, 1:] = matrix

else:

temp[:-1,:-1] = matrix

return temp

def increase\_vec\_size(vec, begin):

size = vec.shape[0] + 1

temp = np.zeros(size)

if begin:

temp[1:] = vec

else:

temp[:-1] = vec

return temp

def solution(x):

return np.exp(-x) / x

def f(x):

return 0

def p(x):

return 2

def q(x):

return x

h = 0.01

x0 = 1

x1 = 2

y0 = np.exp(-1)

y1 = 0.5 \* np.exp(-2)

cond1 = {"order": 1, "value": y0}

cond2 = {"order": 1, "value": y1}

def finite\_differences(p, q, f, h, x0, x1, cond1, cond2):

points = np.arange(x0 + h, x1, h)

a = lambda x: (1 / np.power(h, 2) - p(x) / (2 \* h))

b = lambda x: -2 / np.power(h, 2) + q(x)

c = lambda x: (1 / np.power(h, 2) + p(x) / (2 \* h))

n = len(points)

matrix = np.zeros((n, n))

d = np.zeros((n,))

for i in range(0, n):

d[i] = f(points[i])

if i - 1 >= 0:

matrix[i,i - 1] = a(points[i])

matrix[i,i] = b(points[i])

if i + 1 < len(points):

matrix[i, i + 1] = c(points[i])

if cond1["order"] == 1:

d[0] -= cond1["value"] \* a(points[0])

else:

matrix = increase\_matrix\_size(matrix, True)

d = increase\_vec\_size(d, True)

matrix[1,0] = a(points[0])

matrix[0,0] = -(2 / (h \* (2 - p(x0) \* h))) + (q(x0) \* h) / (2 - p(x0) \* h) + cond1["alpha"]

matrix[0,1] = 2 / (h \* (2 - p(x0) \* h))

d[0] = cond1["value"] + (h \* f(x0)) / (2 - p(x0) \* h)

points = np.insert(points, 0, x0)

if cond2["order"] == 1:

d[-1] -= cond2["value"] \* c(points[-1])

else:

matrix = increase\_matrix\_size(matrix, False)

d = increase\_vec\_size(d, False)

matrix[-2, -1] = c(points[-1])

matrix[-1, -2] = - 2 / (h \* (2 + p(x1) \* h))

matrix[-1, -1] = (2 / (h \* (2 + p(x1) \* h))) - (q(x1) \* h) / (2 + p(x1) \* h) + cond2["beta"]

d[-1] = cond2["value"] - (h \* f(x1)) / (2 + p(x1) \* h)

points = np.insert(points, len(points), x1)

y = sweep\_method(matrix, d)

return points, y

def f1(x,y1,y2):

return y2

def f2(x,y1,y2):

return (x \* y1 - 2 \* y2) / x

def secant(fun, x0, x1, epsilon=0.001):

fx0 = None

while True:

fx1 = fun(x1)

if fx0 is None:

fx0 = fun(x0)

new\_x0, new\_x1 = x1, x1 - (fx1 \* (x1 - x0))/(fx1 - fx0)

if np.abs(new\_x1 - new\_x0) < epsilon:

return new\_x1

x0, x1 = new\_x0, new\_x1

fx0 = fx1

def shooting\_method(functions, x0, x1, y0, y1, tan1, h, epsilon):

def solve(x):

solver = RungeKutta(functions, x0, [y0, x])

points, values = solver.integrate(x1, h)

return values[-1, 0] - y1

result\_tan = secant(solve, tan1, tan1 + h / 2, epsilon)

p, y = RungeKutta(functions, x0, [y0, result\_tan]).integrate(x1, h)

return p, y[:,0]

def fd\_error():

hp, hy = finite\_differences(p, q, f, h, x0, x1, cond1, cond2)

h2p, h2y = finite\_differences(p, q, f, h / 2, x0, x1, cond1, cond2)

error = [np.abs(hy[i] - h2y[2\*i]) for i in range(0, len(hp))]

error = np.array(error) / (np.power(2,2) - 1)

return error

def sm\_error():

hp, hy = shooting\_method([f1, f2], x0, x1, y0, y1, 0.5, h, 0.001)

h2p, h2y = shooting\_method([f1, f2], x0, x1, y0, y1, 0.5, h / 2, 0.001)

error = [np.abs(hy[i] - h2y[2\*i]) for i in range(0, min([len(hp), len(h2p) // 2]))]

error = np.array(error) / (np.power(2, 2) - 1)

return error

fd\_points, fd\_y = finite\_differences(p, q, f, h, x0, x1, cond1, cond2)

precise = solution(fd\_points)

sm\_points, sm\_y = shooting\_method([f1, f2], x0, x1, y0, y1, 0.5, h, 0.001)

plt.plot(fd\_points, solution(fd\_points), label="precise solution")

plt.plot(fd\_points, fd\_y, label="finite differences solution")

plt.plot(sm\_points, sm\_y, label="shooting solution")

print("finite differences error {}".format(fd\_error()))

print()

print("shooting method error {}".format(sm\_error()))

print()

print("finide differences precise error {}".format(np.abs(solution(fd\_points) - fd\_y)))

print()

print("shooting method precise error {}".format(np.abs(solution(sm\_points) - sm\_points)))

plt.legend()

plt.show()

Результат:

finite differences error [0.00083539 0.00083294 0.00083051 0.00082811 0.00082572 0.00082336

0.00082101 0.00081867 0.00081635 0.00081404 0.00081174 0.00080945

0.00080716 0.00080488 0.00080261 0.00080034 0.00079807 0.0007958

0.00079353 0.00079125 0.00078898 0.0007867 0.00078442 0.00078213

0.00077983 0.00077753 0.00077521 0.00077289 0.00077056 0.00076821

0.00076585 0.00076348 0.00076109 0.00075869 0.00075628 0.00075385

0.0007514 0.00074893 0.00074645 0.00074394 0.00074142 0.00073888

0.00073632 0.00073374 0.00073114 0.00072851 0.00072587 0.0007232

0.00072051 0.00071779 0.00071506 0.00071229 0.00070951 0.0007067

0.00070387 0.00070101 0.00069813 0.00069522 0.00069229 0.00068933

0.00068634 0.00068334 0.0006803 0.00067724 0.00067415 0.00067104

0.0006679 0.00066474 0.00066155 0.00065833 0.00065509 0.00065182

0.00064852 0.0006452 0.00064186 0.00063848 0.00063509 0.00063166

0.00062822 0.00062474 0.00062124 0.00061772 0.00061417 0.0006106

0.000607 0.00060338 0.00059974 0.00059607 0.00059238 0.00058866

0.00058492 0.00058116 0.00057738 0.00057357 0.00056975 0.0005659

0.00056203 0.00055814 0.00055423]

shooting method error [0.00000000e+00 2.82717662e-06 5.59919884e-06 8.31795033e-06

1.09852424e-05 1.36028173e-05 1.61723519e-05 1.86954602e-05

2.11736970e-05 2.36085597e-05 2.60014920e-05 2.83538854e-05

3.06670821e-05 3.29423771e-05 3.51810202e-05 3.73842182e-05

3.95531365e-05 4.16889009e-05 4.37925998e-05 4.58652850e-05

4.79079740e-05 4.99216510e-05 5.19072682e-05 5.38657478e-05

5.57979823e-05 5.77048363e-05 5.95871477e-05 6.14457283e-05

6.32813654e-05 6.50948222e-05 6.68868392e-05 6.86581350e-05

7.04094069e-05 7.21413320e-05 7.38545680e-05 7.55497535e-05

7.72275095e-05 7.88884391e-05 8.05331291e-05 8.21621500e-05

8.37760569e-05 8.53753898e-05 8.69606744e-05 8.85324226e-05

9.00911328e-05 9.16372907e-05 9.31713694e-05 9.46938302e-05

9.62051227e-05 9.77056854e-05 9.91959463e-05 1.00676323e-04

1.02147222e-04 1.03609042e-04 1.05062170e-04 1.06506988e-04

1.07943864e-04 1.09373163e-04 1.10795237e-04 1.12210433e-04

1.13619090e-04 1.15021540e-04 1.16418107e-04 1.17809109e-04

1.19194857e-04 1.20575655e-04 1.21951802e-04 1.23323590e-04

1.24691307e-04 1.26055233e-04 1.27415644e-04 1.28772811e-04

1.30127000e-04 1.31478471e-04 1.32827481e-04 1.34174280e-04

1.35519116e-04 1.36862231e-04 1.38203865e-04 1.39544252e-04

1.40883622e-04 1.42222204e-04 1.43560220e-04 1.44897891e-04

1.46235432e-04 1.47573059e-04 1.48910981e-04 1.50249405e-04

1.51588535e-04 1.52928574e-04 1.54269719e-04 1.55612168e-04

1.56956113e-04 1.58301745e-04 1.59649254e-04 1.60998825e-04

1.62350643e-04 1.63704889e-04 1.65061745e-04 1.66421387e-04

1.67783992e-04]

finide differences precise error [0.00226269 0.00440998 0.00644633 0.00837597 0.01020296 0.01193121

0.01356444 0.01510623 0.01656 0.01792904 0.0192165 0.0204254

0.02155865 0.02261903 0.02360923 0.02453182 0.02538928 0.02618397

0.0269182 0.02759416 0.02821398 0.02877968 0.02929324 0.02975655

0.03017144 0.03053965 0.03086289 0.03114279 0.03138093 0.03157883

0.03173795 0.03185972 0.0319455 0.03199663 0.03201436 0.03199995

0.03195459 0.03187943 0.03177558 0.03164414 0.03148614 0.0313026

0.03109451 0.0308628 0.0306084 0.0303322 0.03003506 0.02971783

0.02938131 0.02902629 0.02865354 0.02826379 0.02785777 0.02743617

0.02699967 0.02654892 0.02608458 0.02560725 0.02511754 0.02461603

0.02410331 0.02357991 0.02304638 0.02250323 0.02195099 0.02139014

0.02082117 0.02024454 0.01966071 0.01907012 0.01847321 0.01787039

0.01726207 0.01664866 0.01603054 0.01540809 0.01478168 0.01415166

0.01351839 0.0128822 0.01224342 0.01160239 0.01095941 0.01031479

0.00966883 0.00902182 0.00837404 0.00772578 0.0070773 0.00642885

0.00578071 0.00513312 0.00448631 0.00384054 0.00319602 0.00255299

0.00191166 0.00127225 0.00063496]

shooting method precise error [0.63212056 0.64938715 0.66647555 0.6833913 0.70013973 0.71672595

0.7331549 0.74943129 0.7655597 0.7815445 0.79738992 0.81310004

0.82867875 0.84412986 0.859457 0.87466368 0.88975329 0.90472911

0.91959429 0.93435188 0.94900482 0.96355597 0.97800806 0.99236376

1.00662563 1.02079616 1.03487776 1.04887274 1.06278336 1.0766118

1.09036016 1.10403049 1.11762477 1.13114492 1.14459278 1.15797018

1.17127884 1.18452047 1.1976967 1.21080913 1.22385931 1.23684874

1.24977886 1.2626511 1.27546683 1.28822739 1.30093406 1.31358811

1.32619075 1.33874318 1.35124656 1.363702 1.3761106 1.38847342

1.40079149 1.41306582 1.42529739 1.43748714 1.44963601 1.4617449

1.47381468 1.4858462 1.49784031 1.50979781 1.52171949 1.53360612

1.54545845 1.55727721 1.56906311 1.58081685 1.5925391 1.60423053

1.61589177 1.62752346 1.63912621 1.6507006 1.66224724 1.67376667

1.68525947 1.69672616 1.70816728 1.71958335 1.73097486 1.74234231

1.75368618 1.76500694 1.77630504 1.78758093 1.79883505 1.81006783

1.82127967 1.832471 1.84364221 1.85479368 1.8659258 1.87703894

1.88813346 1.89920972 1.91026806 1.92130883 1.93233236]

