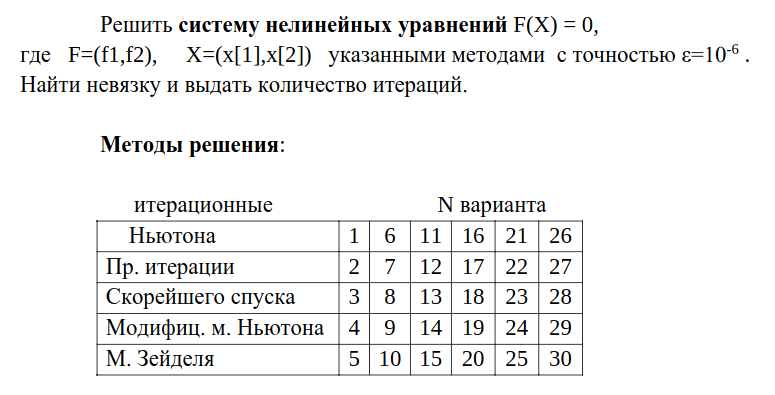
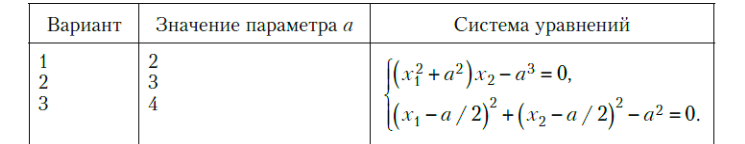
Лабораторная работа 2 вариант 3



код программы   
import numpy as np



import math

import matplotlib.pyplot as plt

# Определение функции F(X) - система нелинейных уравнений

def f(X):

a = 4

f1 = (X[0]\*\*2 + a\*\*2) \* X[1] - a\*\*3

f2 = (X[0] - a/2)\*\*2 + (X[1] - a/2)\*\*2 - a\*\*2

return [f1, f2]

# Определение градиента функции F(X)

def gradient(X):

a = 4

df1\_dx1 = 2 \* X[0] \* X[1]

df1\_dx2 = X[0]\*\*2 + a\*\*2 - 3 \* a\*\*2

df2\_dx1 = X[0] - a/2

df2\_dx2 = 2 \* X[1] - a

W = [[df1\_dx1, df1\_dx2],

[df2\_dx1, df2\_dx2]]

Wt = transpose\_matrix(W)

tmp = MatrixVectorMult(MatrixMatrixMult(W, Wt), f(X))

u = dot(f(X), tmp) / dot(tmp, tmp)

return scalar(u, MatrixVectorMult(Wt, f(X)))

def transpose\_matrix(matrix):

rows, cols = len(matrix), len(matrix[0])

transposed = [[0] \* rows for \_ in range(cols)]

for i in range(rows):

for j in range(cols):

transposed[j][i] = matrix[i][j]

return transposed

def dot(vector1, vector2):

if len(vector1) != len(vector2):

raise ValueError("Длины векторов должны быть одинаковыми")

result = 0

for i in range(len(vector1)):

result += vector1[i] \* vector2[i]

return result

def scalar(scalar, vector):

result = [scalar \* x for x in vector]

return result

def vector\_sub(vector1, vector2):

if len(vector1) != len(vector2):

raise ValueError("Длины векторов должны быть одинаковыми")

result = [x - y for x, y in zip(vector1, vector2)]

return result

def MatrixMatrixMult(matrix1, matrix2):

if len(matrix1[0]) != len(matrix2):

raise ValueError("Количество столбцов в первой матрице должно быть равно количеству строк во второй матрице")

result = [[0] \* len(matrix2[0]) for \_ in range(len(matrix1))]

for i in range(len(matrix1)):

for j in range(len(matrix2[0])):

for k in range(len(matrix2)):

result[i][j] += matrix1[i][k] \* matrix2[k][j]

return result

def MatrixVectorMult(matrix, vector):

if len(matrix[0]) != len(vector):

raise ValueError("Количество столбцов в матрице должно быть равно количеству элементов вектора")

result = [0] \* len(matrix)

for i in range(len(matrix)):

for j in range(len(vector)):

result[i] += matrix[i][j] \* vector[j]

return result

def norm(a):

norm = 0

for i in range(len(a)):

norm += math.pow(a[i], 2)

return math.sqrt(norm)

def nev(x):

nev\_vector = []

for i in range (len(x)):

nev\_vector.append(f(x)[i] - 0)

print(f"Вектор невязки: {nev\_vector}")

print(f"Норма вектора невязки: {norm(nev\_vector)}")

# Начальное приближение

X = [1.0, 1.0]

# Точность ε=10^-6

epsilon = 1e-6

iter = 1

# Для отображения изменений X на графике

X\_history = [X]

while True:

grad = gradient(X)

X = vector\_sub(X, grad)

X\_history.append(X)

if norm(f(X)) < epsilon:

break

iter += 1

X\_history = np.array(X\_history)

print(f"Финальное значение X: {X}")

print(f"Количество итераций: {iter}")

nev(X)

# Визуализация изменений X на графике

plt.plot(X\_history[:, 0], X\_history[:, 1], marker='o')

plt.xlabel('X[0]')

plt.ylabel('X[1]')

plt.title('Изменение X на графике')

plt.grid(True)

plt.show()

