­­­­Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа 1. «Решение системы линейных алгебраических уравнений СЛАУ»

  Вариант: Метод простых итераций

Выполнил: Бурейко Роман Олегович 412902

Группа: p3215

ФИО преподавателя: Малышева Татьяна Алексеевна

Санкт-Петербург, 2025

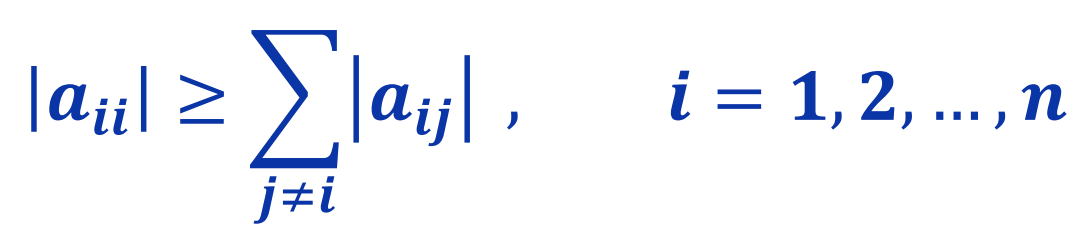
**Цель Работы:**

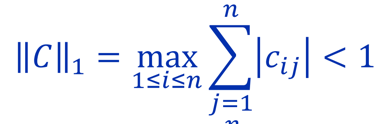
Изучить алгоритм метода простых итераций для решения СЛАУ, его теоретические основы и

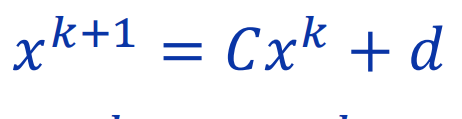
условия сходимости.

Реализовать программный модуль, выполняющий решение СЛАУ методом простых итераций

**Описание метода:**

1. Проверка условия преобладания диагональных элементов  
   
2. Проверка условия сходимости   
   ||C|| < 1, где

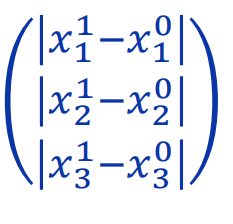


1. Выполнение расчётов по формуле 

Где d - вектор свободных членов,

Сxk – преобразованная матрица,  
k – количество итераций

1. Проверка критерия по абсолютным отклонениям:

 < **ε  
  
  
  
  
  
  
  
  
  
Листинг программы:**

import sys

class Values:

def \_\_init\_\_(*self*, *coefMatrix*, *svobodChlenMatrix*, *epsilon*):

*self*.coefMatrix = coefMatrix

*self*.svobodChlenMatrix = svobodChlenMatrix

*self*.epsilon = epsilon

coefMatrix = []

svobodChlenMatrix = []

epsilon = 0

def inputValues():

while True:

try:

n = *int*(input("Введите количество уравнений (и переменных) системы: "))

if n <= 0:

print("Число уравнений должно быть положительным.")

continue

break

except *ValueError*:

print("Ошибка ввода. Пожалуйста, введите целое число.")

coefMatrix = []

svobodChlenMatrix = []

for i in range(n):

while True:

try:

row = *list*(

map(

*int*,

input(

f"Введите коэффициенты {i+1}-го уравнения через пробел: "

).split(),

)

)

if len(row) != n:

print(f"Ошибка: должно быть {n} коэффициентов.")

continue

coefMatrix.append(row)

break

except *ValueError*:

print("Ошибка ввода. Пожалуйста, введите только целые числа.")

while True:

try:

svobodChlenMatrix = *list*(

map(

*int*, input("Введите свободные члены системы через пробел: ").split()

)

)

if len(svobodChlenMatrix) != n:

print(f"Ошибка: должно быть {n} свободных членов.")

continue

break

except *ValueError*:

print("Ошибка ввода. Пожалуйста, введите только целые числа.")

while True:

try:

print("введите ε")

epsilon = *float*(input())

break

except *ValueError*:

print("Ошибка ввода.")

print("\nМатрица коэффициентов:")

for row in coefMatrix:

print(row)

print("\nВектор свободных членов:", svobodChlenMatrix)

print("ε: ", epsilon)

print("данные введены верно? 1/0")

if *int*(input()) == 0:

inputValues(

*coefMatrix*=coefMatrix, *svobodChlenMatrix*=svobodChlenMatrix, *epsilon*=epsilon

)

else:

return Values(

*coefMatrix*=coefMatrix, *svobodChlenMatrix*=svobodChlenMatrix, *epsilon*=epsilon

)

def is\_diagonally\_dominant(*matrix*):

n = len(matrix)

for i in range(n):

diag = abs(matrix[i][i])

sum\_rest = sum(abs(matrix[i][j]) for j in range(n) if j != i)

if diag < sum\_rest:

return False

return True

def rearrange\_matrix(*matrix*):

n = len(matrix)

if n == 0:

return matrix, []

unique\_rows = []

for row in matrix:

if row not in unique\_rows:

unique\_rows.append(row)

if len(unique\_rows) < n:

print("В матрице есть одинаковые строки. Они будут обработаны как одна.")

graph = [[] for \_ in range(n)]

for j in range(n):

for i in range(n):

diag\_candidate = abs(matrix[j][i])

sum\_rest = sum(abs(matrix[j][k]) for k in range(n) if k != i)

if diag\_candidate >= sum\_rest:

graph[i].append(j)

# Алгоритм Куна

match = [-1] \* n

visited = [False] \* n

def dfs(*u*):

for v in graph[u]:

if not visited[v]:

visited[v] = True

if match[v] == -1 or dfs(match[v]):

match[v] = u

return True

return False

for u in range(n):

visited = [False] \* n

dfs(u)

perm = [-1] \* n

for j in range(n):

if match[j] != -1:

perm[match[j]] = j

if -1 in perm:

return None

return perm

def normaCount(*matrix*) -> *int*:

max = 0

for i in range(0, len(matrix)):

summ = sum(abs(matrix[i][j]) for j in range(0, len(matrix)))

if summ > max:

max = summ

return max

def expressVariables(*matrix*, *array*):

for i in range(0, len(matrix)):

num = matrix[i][i]

matrix[i][i] = 0

if num == 0:

print(f"нельзя выразить х{i+1} так как он равен нулю")

return None

if num == 1:

for j in range(0, len(matrix)):

if matrix[i][j] == matrix[i][i]:

continue

matrix[i][j] = -matrix[i][j]

continue

if num != 1:

array[i] = array[i] / num

for j in range(0, len(matrix)):

if matrix[i][j] == matrix[i][i]:

continue

matrix[i][j] = -matrix[i][j] / num

continue

return matrix, array

def drawSlauSystem(*matrix*, *array*):

print("Выразим диагональные коэффиценты")

for i in range(0, len(matrix)):

out = f"x{i+1} = "

for j in range(0, len(matrix)):

if matrix[i][j] != 0:

if matrix[i][j] > 0:

out += f"+ {matrix[i][j]}x{j+1} "

elif matrix[i][j] < 0:

out += f"{matrix[i][j]} x{j+1} "

if array[i] != 0:

if array[i] > 0:

out += f"+ {array[i]} "

elif array[i] < 0:

out += f"{array[i]} "

print(out)

def drawCMatrix(*C*):

print("матрица C:")

for row in C:

print(row)

print(" ")

def drawdMatrix(*d*):

print("Матрица d:\n")

for row in d:

print(row)

print(" ")

def convergenceСondition(*num*):

if num < 1:

print(f"норма: {num} < 1 => условие сходимости выполняется, всё ок")

else:

print(f"норма: {num} !< 1 => условие сходимости не выполняется, метод не применим")

sys.exit(0)

def absoluteDeviationCriterion(*x\_prev*, *x\_current*, *epsilon*):

max\_diff = max(abs(x\_current[i] - x\_prev[i]) for i in range(len(x\_prev)))

print(f"Максимальное отклонение: {max\_diff}")

if max\_diff < epsilon:

print(f"{max\_diff} < {epsilon} → Критерий выполнен")

return True, max\_diff

else:

print(f"{max\_diff} >= {epsilon} → Критерий не выполнен")

return False, max\_diff

def simpleIterations(*C*, *d*, *epsilon*):

max\_iterations = 500

iteration = 0

x\_prev = d.copy()

x\_current = [0.0] \* len(d)

while iteration < max\_iterations:

print(f"\nИтерация {iteration + 1}:")

for i in range(len(C)):

x\_current[i] = sum(C[i][j] \* x\_prev[j] for j in range(len(C))) + d[i]

print(f"x\_{i+1} = {x\_current[i]:.4f}")

is\_stop, max\_diff = absoluteDeviationCriterion(x\_prev, x\_current, epsilon)

if is\_stop:

break

x\_prev = x\_current.copy()

iteration += 1

return x\_current

inputValues = inputValues()

if is\_diagonally\_dominant(inputValues.coefMatrix):

print("Матрица уже обладает диагональным преобладанием.")

print("Свободные члены:", inputValues.svobodChlenMatrix)

perm = rearrange\_matrix(inputValues.coefMatrix)

if perm is not None:

new\_matrix = [inputValues.coefMatrix[i].copy() for i in perm]

new\_svobod = [inputValues.svobodChlenMatrix[i] for i in perm]

print("Преобразованная матрица:")

for row in new\_matrix:

print(row)

print("Преобразованные свободные члены:", new\_svobod)

else:

print("Невозможно преобразовать матрицу.")

sys.exit(0)

C, d = expressVariables(new\_matrix, new\_svobod)

drawSlauSystem(new\_matrix, new\_svobod)

drawCMatrix(C)

drawdMatrix(d)

normaCount(C)

convergenceСondition(normaCount(C)) simpleIterations(C,d,inputValues.epsilon)

Вывод: В ходе лабораторной работы познакомился с методом решения СЛАУ простых итераций. Понял его плюсы и минусы, а также реализовал программу на языке программирования python. Вспомнил линейную алгебру