Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Лабораторна робота №2

З курсу «Чисельні методи»

З теми «Ітераційні методи розв’язання СЛАР»

Варіант №5

Виконав студент 2 курсу групи КА-01

Вагін Олександр Вікторович

Перевірила старший викладач

Хоменко Ольга Володимирівна

Київ-2022

**Завдання №1**

Розв’язати систему рівнянь з точністю 0,00001:

Я обрав метод Якобі.

Перевіримо достатню умову:

Для матриці А виконується умова діагональної переваги:

|4.855| > |1.239| + |0.272| + |0.258|

|4.954| > |1.491| + |0.124| + |0.236|

|4.354| > |0.456| + |0.285| + |0.254|

|2.874| > |0.412| + |0.335| + |0.158|

Зведемо систему до вигляду *x = Bx + c*, для цього призначений цей програмний код:

def highlight\_root(A, b):

"""

Takes:

matrix of coefficients and free terms(A, b from Ax = b)

Returns:

matrix of coefficients and free terms(B, c from x = Bx + c)

"""

B = []

c = []

for i in range(len(A)):

B.append(tuple(round(-n / A[i][i], 6) for n in A[i][:i] + A[i][i + 1:]))

c.append((round(b[i] / A[i][i], 6),))

return tuple(B), tuple(c)

Результатом є:

Покладемо c як початкове наближення

Оскільки q =

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № ітерації |  |  |  |  |  |
| 0 | 0.24552 | 0.051675 | 0.195682 | 0.29993 | 0 |
| 1 | 0.205431 | -0.041405 | 0.149089 | 0.247953 | 0.09308 |
| 2 | 0.234557 | -0.025697 | 0.162412 | 0.267111 | 0.029126 |
| 3 | 0.228784 | -0.035709 | 0.157216 | 0.260372 | 0.010012 |
| 4 | 0.231989 | -0.033521 | 0.158869 | 0.262652 | 0.003205 |
| 5 | 0.231216 | -0.034635 | 0.158257 | 0.261847 | 0.001114 |
| 6 | 0.231578 | -0.034349 | 0.158458 | 0.262121 | 0.000362 |
| 7 | 0.231479 | -0.034476 | 0.158386 | 0.262025 | 0.000127 |
| 8 | 0.23152 | -0.03444 | 0.15841 | 0.262058 | 0.000041 |
| 9 | 0.231508 | -0.034454 | 0.158401 | 0.262046 | 0.000014 |
| 10 | 0.231513 | -0.03445 | 0.158404 | 0.26205 | 0.00005 |

Вектор невʼязки:

**Код програм:**

import numpy as np  
import random  
  
A = ((4.855, 1.239, 0.272, 0.258),  
 (1.491, 4.954, 0.124, 0.236),  
 (0.456, 0.285, 4.354, 0.254),  
 (0.412, 0.335, 0.158, 2.874))  
  
b = (1.192, 0.256, 0.852, 0.862)  
  
  
def check\_diagonal\_advantage(matrix):  
 *"""Takes square matrix, returns True if matrix has a diagonal advantage, else False"""* check\_line = []  
 for i in range(len(matrix)):  
 line = tuple(map(abs, matrix[i]))  
 check\_line.append(line[i] > sum(line[:i] + line[i + 1:]))  
 s = f'|{matrix[i][i]}| {">" if check\_line[-1] else "<"} ' + " + ".join(  
 [f'|{n}|' for n in matrix[i][:i] + matrix[i][i + 1:]])  
 print(s)  
 return all(check\_line)  
  
  
def highlight\_root(A, b):  
 *"""  
 Takes:  
 matrix of coefficients and free terms(A, b from Ax = b)  
 Returns:  
 matrix of coefficients and free terms(B, c from x = Bx + c)  
 """* B = []  
 c = []  
 for i in range(len(A)):  
 B.append(tuple(round(-n / A[i][i], 6) for n in A[i][:i] + A[i][i + 1:]))  
 c.append((round(b[i] / A[i][i], 6),))  
 return tuple(B), tuple(c)  
  
  
def matrix\_norm(matrix):  
 *"""Takes matrix, return matrix norm(maximum of sum's of absolutes of each line)"""* print(r"max{", end='')  
 print(\*[round(sum(map(abs, line)), 6) for line in matrix], sep=', ', end='')  
 print(r'}')  
 return max([sum(map(abs, line)) for line in matrix])  
  
  
def choose\_criterion(q):  
 *"""Takes q and returns a stopping criterion"""* if q <= 0.5:  
 return lambda x1, x0, eps: max(map(lambda x, y: abs(x - y), x1, x0)) < eps  
 return lambda x1, x0, eps: q / (1 - q) \* max(map(lambda x, y: abs(x - y), x1, x0)) < eps  
  
  
def next\_term(B, c, x0):  
 *"""Takes matrix of coefficients and free terms of equation x = Bx + c and previous term, returns next term"""* x1 = tuple(round(sum([B[i][j] \* x0[j if j < i else j + 1] for j in range(len(B[i]))]) + c[i][0], 6) for i in  
 range(len(x0)))  
 return x1  
  
  
def find\_root(criterion, B, c, x0, eps):  
 *"""Finds root of system of linear algebraic equations  
 Takes:  
 criterion - criterion of stopping  
 B, c - matrix of coefficients and free terms of equation x = Bx + c  
 x0 - initial approximation  
 eps - precision  
 Return:  
 x1 - root of system of linear algebraic equations  
 log - history of approximations  
 """* log = [x0]  
 x1 = next\_term(B, c, x0)  
 log.append(x1)  
 while not criterion(x1, x0, eps):  
 x0 = x1  
 x1 = next\_term(B, c, x0)  
 log.append(x1)  
 return x1, log  
  
  
def residual\_vector(A, b, x):  
 *"""Takes system of linear algebraic equations and its root, return residual vector(b - Ax)"""* res = tuple(abs(round(b[i] - sum([A[i][j] \* x[j] for j in range(len(x))]), 7)) for i in range(len(b)))  
 return res  
  
  
def print\_matrix(matrix):  
 *"""Takes matrix and print it to the stdout"""* for line in matrix:  
 for num in line:  
 print(str(num).ljust(10), end=' ')  
 print()  
  
  
def print\_matrix\_equation(A, b):  
 *"""Takes matrix of coefficients and free terms of equation Ax = b and print this equation"""* for i in range(len(A)):  
 s = [str(A[i][j]) + f'∙x{j + 1}' for j in range(len(A[i]))]  
 print(' + '.join(s) + ' = ' + str(b[i]))  
  
  
def print\_trans\_equation(B, c):  
 *"""Takes matrix of coefficients and free terms of equation x = Bx + c and print this equation"""* for i in range(len(B)):  
 s = [str(B[i][j]) + f'∙x{j + 1 if j < i else j + 2}' for j in range(len(B[i]))]  
 print(f'x{i + 1} = ' + ' + '.join(s) + ' + ' + str(\*c[i]))  
  
  
def print\_log(log):  
 *"""Takes log and print it"""* print("-" \* 89)  
 print('|' + "№ iteration".center(15) + '|' +  
 "|".join([f"x{i + 1}".center(12) for i in range(len(log[0]))]) + "|" +  
 "||x^k - x^k-1||".center(19) + "|")  
 print("-" \* 89)  
 for i in range(len(log)):  
 print('|' + f"{i}".center(15) + '|' +  
 "|".join([str(x).center(12) for x in log[i]]) + "|" +  
 str(round(max(map(lambda x, y: abs(x - y), log[i], log[i - 1])), 6) if i > 0 else 0).center(19) + "|")  
 print("-" \* 89)  
  
  
# Lab steps  
  
# Print input data  
print('Initial data:')  
print\_matrix\_equation(A, b)  
print()  
  
# Check diagonal advantage  
print('Checking diagonal advantage:')  
print('Result:', check\_diagonal\_advantage(A), '\n')  
  
# Transform equation from Ax = b to x = Bx + c  
print("Transforming equation from Ax = b to x = Bx + c:")  
B, c = highlight\_root(A, b)  
print\_trans\_equation(B, c)  
print()  
  
# Find matrix norm  
print("Calculating matrix norm:")  
q = matrix\_norm(B)  
print('Result:', q, '\n')  
  
# Put c as an initial approximation  
print('Put c as an initial approximation')  
x = tuple(n[0] for n in c)  
print(f'x0 = {x}\n')  
  
# Choose stopping criterion  
print("Choosing stopping criterion:")  
criterion = choose\_criterion(q)  
print(f"q = {q}")  
if q <= 0.5:  
 print("Criterion:||x^k - x^k-1||≤ ε")  
else:  
 print("Criterion: (q/1−q) \* ||x^k - x^k-1||≤ ε")  
  
# Search for root  
print("\nCalculating root:")  
x, log = find\_root(criterion, B, c, x, 0.00001)  
print\_log(log)  
print("Result:", x)  
  
# Calculate residual vector  
print("\nResidual vector:")  
print(residual\_vector(A, b, x))  
  
# Search for root with random initial approximation  
print("\nCalculating root with random initial approximation:")  
x = tuple(round(random.uniform(t - 1, t + 1), 6) for t in x)  
print("x =", x)  
x, log = find\_root(criterion, B, c, x, 0.00001)  
print\_log(log)  
print("Result:", x)  
  
# Calculate residual vector for this case  
print("\nResidual vector for this case:")  
print(residual\_vector(A, b, x))  
  
# Calculate root using numpy  
print("\nCalculating root using numpy:")  
print(np.linalg.solve(A, b))

**Результат роботи програм:**

Initial data:

4.855∙x1 + 1.239∙x2 + 0.272∙x3 + 0.258∙x4 = 1.192

1.491∙x1 + 4.954∙x2 + 0.124∙x3 + 0.236∙x4 = 0.256

0.456∙x1 + 0.285∙x2 + 4.354∙x3 + 0.254∙x4 = 0.852

0.412∙x1 + 0.335∙x2 + 0.158∙x3 + 2.874∙x4 = 0.862

Checking diagonal advantage:

|4.855| > |1.239| + |0.272| + |0.258|

|4.954| > |1.491| + |0.124| + |0.236|

|4.354| > |0.456| + |0.285| + |0.254|

|2.874| > |0.412| + |0.335| + |0.158|

Result: True

Transforming equation from Ax = b to x = Bx + c:

x1 = -0.255201∙x2 + -0.056025∙x3 + -0.053141∙x4 + 0.24552

x2 = -0.300969∙x1 + -0.02503∙x3 + -0.047638∙x4 + 0.051675

x3 = -0.104731∙x1 + -0.065457∙x2 + -0.058337∙x4 + 0.195682

x4 = -0.143354∙x1 + -0.116562∙x2 + -0.054976∙x3 + 0.29993

Calculating matrix norm:

max{0.364367, 0.373637, 0.228525, 0.314892}

Result: 0.373637

Put c as an initial approximation

x0 = (0.24552, 0.051675, 0.195682, 0.29993)

Choosing stopping criterion:

q = 0.373637

Criterion:||x^k - x^k-1||≤ ε

Calculating root:

-----------------------------------------------------------------------------------------

| № iteration | x1 | x2 | x3 | x4 | ||x^k - x^k-1|| |

-----------------------------------------------------------------------------------------

| 0 | 0.24552 | 0.051675 | 0.195682 | 0.29993 | 0 |

-----------------------------------------------------------------------------------------

| 1 | 0.205431 | -0.041405 | 0.149089 | 0.247953 | 0.09308 |

-----------------------------------------------------------------------------------------

| 2 | 0.234557 | -0.025697 | 0.162412 | 0.267111 | 0.029126 |

-----------------------------------------------------------------------------------------

| 3 | 0.228784 | -0.035709 | 0.157216 | 0.260372 | 0.010012 |

-----------------------------------------------------------------------------------------

| 4 | 0.231989 | -0.033521 | 0.158869 | 0.262652 | 0.003205 |

-----------------------------------------------------------------------------------------

| 5 | 0.231216 | -0.034635 | 0.158257 | 0.261847 | 0.001114 |

-----------------------------------------------------------------------------------------

| 6 | 0.231578 | -0.034349 | 0.158458 | 0.262121 | 0.000362 |

-----------------------------------------------------------------------------------------

| 7 | 0.231479 | -0.034476 | 0.158386 | 0.262025 | 0.000127 |

-----------------------------------------------------------------------------------------

| 8 | 0.23152 | -0.03444 | 0.15841 | 0.262058 | 4.1e-05 |

-----------------------------------------------------------------------------------------

| 9 | 0.231508 | -0.034454 | 0.158401 | 0.262046 | 1.4e-05 |

-----------------------------------------------------------------------------------------

| 10 | 0.231513 | -0.03445 | 0.158404 | 0.26205 | 5e-06 |

-----------------------------------------------------------------------------------------

Result: (0.231513, -0.03445, 0.158404, 0.26205)

Residual vector:

(6.9e-06, 6.5e-06, 3.4e-06, 2.1e-06)

Calculating root with random initial approximation:

x = (-0.210983, 0.028165, -0.754689, 0.742995)

-----------------------------------------------------------------------------------------

| № iteration | x1 | x2 | x3 | x4 | ||x^k - x^k-1|| |

-----------------------------------------------------------------------------------------

| 0 | -0.210983 | 0.028165 | -0.754689 | 0.742995 | 0 |

-----------------------------------------------------------------------------------------

| 1 | 0.24113 | 0.098669 | 0.172591 | 0.368382 | 0.92728 |

-----------------------------------------------------------------------------------------

| 2 | 0.191094 | -0.042767 | 0.142479 | 0.244374 | 0.141436 |

-----------------------------------------------------------------------------------------

| 3 | 0.235466 | -0.021046 | 0.164212 | 0.269688 | 0.044372 |

-----------------------------------------------------------------------------------------

| 4 | 0.227359 | -0.036151 | 0.156666 | 0.2596 | 0.015105 |

-----------------------------------------------------------------------------------------

| 5 | 0.232173 | -0.033041 | 0.159093 | 0.262938 | 0.004814 |

-----------------------------------------------------------------------------------------

| 6 | 0.231066 | -0.03471 | 0.15819 | 0.261752 | 0.001669 |

-----------------------------------------------------------------------------------------

| 7 | 0.231606 | -0.034298 | 0.158484 | 0.262155 | 0.00054 |

-----------------------------------------------------------------------------------------

| 8 | 0.231463 | -0.034487 | 0.158377 | 0.262013 | 0.000189 |

-----------------------------------------------------------------------------------------

| 9 | 0.231524 | -0.034434 | 0.158413 | 0.262062 | 6.1e-05 |

-----------------------------------------------------------------------------------------

| 10 | 0.231506 | -0.034456 | 0.1584 | 0.262045 | 2.2e-05 |

-----------------------------------------------------------------------------------------

| 11 | 0.231514 | -0.034449 | 0.158405 | 0.262051 | 8e-06 |

-----------------------------------------------------------------------------------------

Result: (0.231514, -0.034449, 0.158405, 0.262051)

Residual vector for this case:

(1.35e-05, 1.33e-05, 8.7e-06, 5.9e-06)

Calculating root using numpy:

[ 0.23151188 -0.03445094 0.15840342 0.26204956]

**Висновок:**

З отриманих результатів можемо зробити висновок, що метод Якобі працює достатньо швидко: 10 ітерацій, якщо за початкове наближення брати вектор вільних членів та 11, якщо випадково відхилятися на нього на відстань 1. До того ж за власними відчуттями реалізований алгоритм працює трохи швидше за linalg.solve з модуля NumPy