Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота № 3

з дисципліни «Спеціальні розділи математики-2.  
Чисельні методи»

на тему

«**Розв’язання систем лінійних алгебраїчних рівнянь (СЛАР) ітераційними методами. Метод простої ітерації. Метод Зейделя**»

Виконала:

студент гр. ІС-34

Колосов Ігор

Викладач:

доц. Рибачук Л.В.

Київ – 2024

**Зміст**

[**Зміст** 2](#_Toc158533784)

[**1 Постановка задачі** 3](#_Toc158533785)

[**2 Розв’язок** 4](#_Toc158533786)

[**3 Контрольні питання** 5](#_Toc158533787)

[4 Висновок 6](#_Toc158533788)

**1 Постановка задачі**

**1 Постановка задачі**

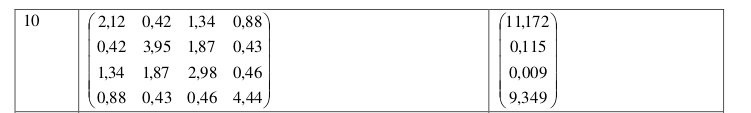
Якщо матриця не є матрицею із діагональною перевагою, звести систему до еквівалентної, у якій є діагональна перевага. Можна, наприклад, провести одну ітерацію метода Гауса, зкомбінувавши рядки з метою отримати нульовий недіагональний елемент у стовпчику.

Розробити програму, що реалізує розв’язання системи методом простої ітерації та методом Зейделя. Обчислення проводити з з кількістю значущих цифр m = 6. Для кожної ітерації розраховувати нев’язку ***r = b – Ax***, де x – отриманий розв’язок.

Розв’язати задану систему рівнянь за допомогою програмного забезпечення Mathcad. Навести результат перевірки: вектор нев’язки **r = b – Axm**, де xm – отриманий у Mathcad розв’язок.

Порівняти корені рівнянь, отримані у Mathcad, із власними результатами за допомогою методу середньоквадратичної похибки.

Розв’язати систему рівнянь методом Зейделя та алгоритмом Якобі з кількістю значущих цифр m = 6.  
*Варіант №10*



Скріншот 1 Варіант завдання

**2 Розв’язок**

Перевіримо основну матрицю А на наявність діагональної переваги .

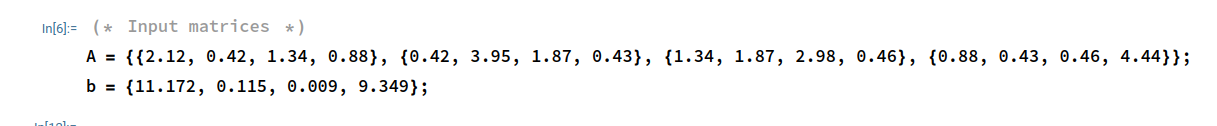
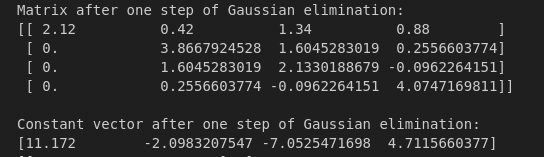


Рис 1.1 Вхідні матриці

Результат усіх 4 дій булевий 0, тобто матриця не є діаганально панівною. Ми маємо виконати елементарні перетворення щоб виправити це.

Рис 1.3 Обчислення до еквівалентної матриці

Рис 1.4 Еквівалентні матриці



Алгоритм Якобі

==================================================

Вхідна матриця

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

[[ 2.12 0.42 1.34 0.88 ]

[ 0. 3.8667924528 1.6045283019 0.2556603774]

[ 0. 1.6045283019 2.1330188679 -0.0962264151]

[ 0. 0.2556603774 -0.0962264151 4.0747169811]]

==================================================

Зведена матриця

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

[[ 2.12 0.42 1.34 0.88 11.172 ]

[ 0. 3.8667924528 1.6045283019 0.2556603774 -2.0983207547]

[ 0. 1.6045283019 2.1330188679 -0.0962264151 -7.0525471698]

[ 0. 0.2556603774 -0.0962264151 4.0747169811 4.7115660377]]

Метод простих ітерацій

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

Ітерація №1

Наближення

[[ 5.2698113208]

[-0.5426515078]

[-3.3063688633]

[ 1.156292832 ]]

Вектор нев'язки

[[ 3.640910218 ]

[ 5.0095441559]

[ 0.9819656163]

[-0.1794255335]]

Ітерація №2

Наближення

[[ 6.9872218009]

[ 0.752878112 ]

[-2.8460046204]

[ 1.112258968 ]]

Вектор нев'язки

[[-1.1222607255]

[-0.7274097427]

[-2.0829511616]

[-0.2869163907]]

Ітерація №3

Наближення

[[ 6.4578535342]

[ 0.5647610223]

[-3.8225319231]

[ 1.0418451473]]

Вектор нев'язки

[[ 1.4495199254]

[ 1.5848677187]

[ 0.2950635249]

[-0.0458736354]]

Ітерація №4

Наближення

[[ 7.1415893481]

[ 0.9746272714]

[-3.6842005049]

[ 1.0305870316]]

Вектор нев'язки

[[-0.3476007831]

[-0.2190784213]

[-0.6587253248]

[-0.0914754234]]

Ітерація №5

Наближення

[[ 6.9776267145]

[ 0.917970902 ]

[-3.9930235232]

[ 1.0081375157]]

Вектор нев'язки

[[ 0.4573740935]

[ 0.5012547247]

[ 0.0887465116]

[-0.0152321432]]

Ітерація №6

Наближення

[[ 7.1933692115]

[ 1.0476015276]

[-3.9514174629]

[ 1.0043993069]]

Вектор нев'язки

[[-0.1069073598]

[-0.0658023894]

[-0.208355722 ]

[-0.0291378126]]

Ітерація №7

Наближення

[[ 7.1429412116]

[ 1.0305842219]

[-4.0490986063]

[ 0.9972484267]]

Вектор нев'язки

[[ 0.1443327751]

[ 0.1585603559]

[ 0.0266166451]

[-0.0050488554]]

Відповідь:

[[ 7.1429412116]

[ 1.0305842219]

[-4.0490986063]

[ 0.9972484267]]

==================================================

Метод Зейделя

[ 0.9920538105]]

Ітерація №1

Наближення

[[ 5.2698113208]

[-0.5426515078]

[-2.8981682068]

[ 1.1218988242]]

Вектор нев'язки

[[ 3.1241880651 -10.1461326896 -15.1003591047 -3.3362458972]

[ 17.6336885894 4.3633678347 -0.5908585804 11.1732546272]

[ 18.3325034718 5.062182717 0.107956302 11.8720695095]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №2

Наближення

[[ 6.7434849364]

[ 0.5857689823]

[-3.6963919527]

[ 1.0322477835]]

Вектор нев'язки

[[ 0.6745761295 -12.5957446253 -17.5499710403 -5.7858578328]

[ 14.5740135651 1.3036928104 -3.6505336047 8.1135796029]

[ 18.2159203716 4.9455996168 -0.0086267983 11.7554864093]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №3

Наближення

[[ 7.0616812238]

[ 0.9229199482]

[-3.9540526201]

[ 1.0050091051]]

Вектор нев'язки

[[ 0.2276319256 -13.0426888291 -17.9969152442 -6.2328020366]

[ 13.6907084386 0.4203876839 -4.5338387312 7.2302744764]

[ 18.2219260894 4.9516053347 -0.0026210804 11.7614921272]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №4

Наближення

[[ 7.1690547737]

[ 1.0316373643]

[-4.0370623178]

[ 0.9962275231]]

Вектор нев'язки

[[ 0.0732994723 -13.1970212824 -18.1512476975 -6.38713449 ]

[ 13.4057572665 0.1354365118 -4.8187899033 6.9453233043]

[ 18.2237021497 4.9533813949 -0.0008450202 11.7632681874]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №5

Наближення

[[ 7.2036299964]

[ 1.066662909 ]

[-4.0638058683]

[ 0.9933983499]]

Вектор нев'язки

[[ 0.0236153013 -13.2467054534 -18.2009318685 -6.4368186609]

[ 13.3139548459 0.0436340912 -4.9105923239 6.8535208837]

[ 18.2242749286 4.9539541739 -0.0002722412 11.7638409664]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №6

Наближення

[[ 7.2147692895]

[ 1.0779472206]

[-4.0724219377]

[ 0.9924868646]]

Вектор нев'язки

[[ 0.0076082292 -13.2627125255 -18.2169389406 -6.452825733 ]

[ 13.2843785126 0.0140577579 -4.9401686572 6.8239445504]

[ 18.2244594608 4.9541387061 -0.000087709 11.7640254986]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №7

Наближення

[[ 7.2183580769]

[ 1.0815827294]

[-4.0751978089]

[ 0.992193208 ]]

Вектор нев'язки

[[ 0.0024511716 -13.2678695831 -18.2220959982 -6.4579827907]

[ 13.274849795 0.0045290403 -4.9496973748 6.8144158328]

[ 18.2245189123 4.9541981576 -0.0000282575 11.76408495 ]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №8

Наближення

[[ 7.2195142899]

[ 1.0827539948]

[-4.0760921217]

[ 0.9920985995]]

Вектор нев'язки

[[ 0.0007897031 -13.2695310516 -18.2237574667 -6.4596442592]

[ 13.2717798925 0.0014591378 -4.9527672773 6.8113459303]

[ 18.224538066 4.9542173113 -0.0000091038 11.7641041037]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №9

Наближення

[[ 7.2198867914]

[ 1.0831313458]

[-4.0763802458]

[ 0.9920681191]]

Вектор нев'язки

[[ 0.0002544216 -13.2700663331 -18.2242927482 -6.4601795407]

[ 13.2707908506 0.0004700959 -4.9537563192 6.8103568883]

[ 18.2245442368 4.9542234821 -0.000002933 11.7641102745]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №10

Наближення

[[ 7.2200068016]

[ 1.0832529184]

[-4.0764730718]

[ 0.9920582992]]

Вектор нев'язки

[[ 0.000081968 -13.2702387868 -18.2244652019 -6.4603519943]

[ 13.2704722073 0.0001514525 -4.9540749626 6.810038245 ]

[ 18.2245462249 4.9542254702 -0.0000009449 11.7641122626]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №11

Наближення

[[ 7.2200454657]

[ 1.0832920859]

[-4.0765029779]

[ 0.9920551354]]

Вектор нев'язки

[[ 0.0000264079 -13.2702943468 -18.2245207619 -6.4604075543]

[ 13.2703695487 0.000048794 -4.9541776211 6.8099355865]

[ 18.2245468654 4.9542261107 -0.0000003044 11.7641129031]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №12

Наближення

[[ 7.2200579223]

[ 1.0833047046]

[-4.0765126129]

[ 0.9920541162]]

Вектор нев'язки

[[ 0.0000085079 -13.2703122468 -18.2245386619 -6.4604254543]

[ 13.2703364749 0.0000157202 -4.9542106949 6.8099025126]

[ 18.2245470717 4.954226317 -0.0000000981 11.7641131095]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №13

Наближення

[[ 7.2200619354]

[ 1.08330877 ]

[-4.076515717 ]

[ 0.9920537878]]

Вектор нев'язки

[[ 0.000002741 -13.2703180137 -18.2245444288 -6.4604312212]

[ 13.2703258193 0.0000050646 -4.9542213505 6.8098918571]

[ 18.2245471382 4.9542263835 -0.0000000316 11.7641131759]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №14

Наближення

[[ 7.2200632284]

[ 1.0833100798]

[-4.0765167171]

[ 0.992053682 ]]

Вектор нев'язки

[[ 0.0000008831 -13.2703198716 -18.2245462867 -6.4604330792]

[ 13.2703223864 0.0000016317 -4.9542247834 6.8098884241]

[ 18.2245471596 4.9542264049 -0.0000000102 11.7641131974]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Ітерація №15

Наближення

[[ 7.2200636449]

[ 1.0833105018]

[-4.0765170393]

[ 0.9920536479]]

Вектор нев'язки

[[ 0.0000002845 -13.2703204702 -18.2245468853 -6.4604336778]

[ 13.2703212804 0.0000005257 -4.9542258894 6.8098873181]

[ 18.2245471665 4.9542264118 -0.0000000033 11.7641132043]

[ 6.4604339623 -6.8098867925 -11.7641132075 0. ]]

Відповідь:

[[ 7.2200636449]

[ 1.0833105018]

[-4.0765170393]

[ 0.9920536479]]

==================================================

Отже, обидва методи дали точний результат і методи працюють.

Перевіряємо через MathCad:

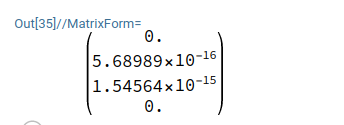
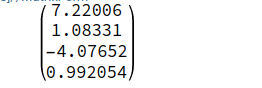


Рис 1.5 Перевірка коренів та вектора нев’язкості

Тепер маючи обидва розв’язки розрахуємо середньоквадратичну похибку.

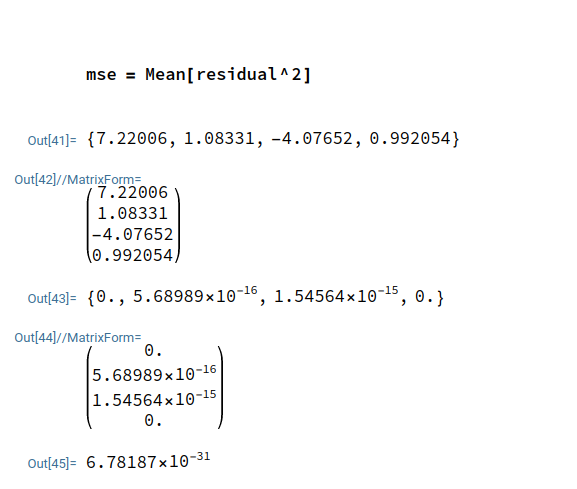


Рис 1.6 Похибка

**3 Лістинг програми**

import numpy as np

def gaussian\_elimination\_one\_step(A, b, column):

A\_copy = np.copy(A)

b\_copy = np.copy(b)

n = len(A\_copy)

# Find the row with the maximum absolute value in the current column

max\_row = column

for i in range(column + 1, n):

if abs(A\_copy[i][column]) > abs(A\_copy[max\_row][column]):

max\_row = i

# Swap the rows to move the maximum value to the diagonal position

A\_copy[[column, max\_row]] = A\_copy[[max\_row, column]]

b\_copy[[column, max\_row]] = b\_copy[[max\_row, column]]

# Perform elimination

for i in range(column + 1, n):

factor = A\_copy[i][column] / A\_copy[column][column]

A\_copy[i] -= factor \* A\_copy[column]

b\_copy[i] -= factor \* b\_copy[column]

return A\_copy, b\_copy

# Input matrix

A = np.array([

[2.12, 0.42, 1.34, 0.88],

[0.42, 3.95, 1.87, 0.43],

[1.34, 1.87, 2.98, 0.46],

[0.88, 0.43, 0.46, 4.44]

])

# Constant vector

b = np.array([11.172, 0.115, 0.009, 9.349])

# Perform one step of Gaussian elimination

column = 0 # Choose the column for which to eliminate non-diagonal elements

A\_step, b\_step = gaussian\_elimination\_one\_step(A, b, column)

# Print the resulting matrix and vector after one step of Gaussian elimination

print("Original Matrix:")

print(A)

print("\nConstant vector:")

print(b)

print("\nMatrix after one step of Gaussian elimination:")

print(A\_step)

print("\nConstant vector after one step of Gaussian elimination:")

print(b\_step)

print(A\_step.tolist())

print(b\_step.tolist())

import numpy as np

from numpy import ndarray

np.set\_printoptions(precision=10, suppress=True)

def diagonal\_dominant(a: ndarray, b: ndarray):

matrix = np.column\_stack([a, b])

matrix[0], matrix[2] = matrix[2].copy(), matrix[0].copy()

matrix /= matrix[:, 0].reshape((-1, 1))

for i in range(1, 4):

matrix[i] -= matrix[i][0] \* matrix[0]

matrix[2] += 8 \* matrix[1]

matrix[0] = matrix[0] - 11 \* matrix[1] + matrix[2]

return matrix[:, :-1], matrix[:, -1:]

def jacobi(A, b, tolerance=1e-6, max\_iterations=10000):

global jacobi\_b

x = np.zeros\_like(b, dtype=np.double)

T = A - np.diag(np.diagonal(A))

for k in range(max\_iterations):

print(f"Ітерація №{k+1}")

x\_old = x

x = (b - np.dot(T, x)) / np.diagonal(A).reshape((-1, 1))

print("Наближення")

print(x.reshape((-1, 1)))

print("Вектор нев'язки")

vector = b - np.dot(A, x.reshape((-1, 1)))

print(vector)

if np.allclose(x\_old, x, atol=tolerance, rtol=0.):

break

jacobi\_b = vector

return x

def seidel(A, b, tolerance=1e-6, max\_iterations=10000):

global seidel\_b

n = len(A)

x = np.zeros\_like(b, dtype=np.double)

# zero vector

converge = False

for k in range(max\_iterations):

if converge:

break

print(f"Ітерація №{k+1}")

x\_new = np.copy(x)

for i in range(n):

s1 = sum(A[i][j] \* x\_new[j] for j in range(i))

s2 = sum(A[i][j] \* x[j] for j in range(i + 1, n))

x\_new[i] = (b[i] - s1 - s2) / A[i][i]

print("Наближення")

print(x\_new.reshape((-1, 1)))

print("Вектор нев'язки")

vector = b - np.dot(A, x\_new.reshape((-1, 1)))

converge = np.allclose(x, x\_new, atol=tolerance, rtol=0.)

print(vector)

seidel\_b = vector

x = x\_new

return x.reshape((-1, 1))

#

A = np.array([

[2.12, 0.42, 1.34, 0.88],

[0.42, 3.95, 1.87, 0.43],

[1.34, 1.87, 2.98, 0.46],

[0.88, 0.43, 0.46, 4.44]

])

b = np.array([

[11.172],

[0.115],

[0.009],

[9.349]

])

# Еквівалентні

A = np.array([[2.12, 0.42, 1.34, 0.88], [0.0, 3.8667924528301887, 1.6045283018867926, 0.25566037735849056], [0.0, 1.6045283018867926, 2.1330188679245285, -0.09622641509433955], [0.0, 0.25566037735849056, -0.09622641509433955, 4.074716981132076]])

b = np.array([11.172, -2.0983207547169807, -7.05254716981132, 4.7115660377358495])

#A, b = A\_step.copy(), b\_step.copy()

b = np.array([[11.172], [-2.0983207547169807], [-7.05254716981132], [4.7115660377358495]])

tolerance = 1e-6

Ab = np.column\_stack([A, b])

print("="\*50)

print("Вхідна матриця")

print("----------------------------------------------------------------------------------------")

print(A)

print("="\*50)

print("Зведена матриця")

print("----------------------------------------------------------------------------------------")

print(np.column\_stack([A, b]))

print("="\*50)

print("Метод Зейделя")

print("----------------------------------------------------------------------------------------")

print("Відповідь:", sei := seidel(A.copy(), b.copy(), tolerance=tolerance), sep="\n")

print("="\*50)

print("Метод простих ітерацій")

print("----------------------------------------------------------------------------------------")

print("Відповідь:", jac := jacobi(A.copy(), b.copy(), tolerance=0.1), sep="\n")

print("="\*50)

print(f"MSE SEIDEL: {np.mean((jacobi\_b) /4):.1f}")

print(f"MSE Jacoby: {np.mean((jacobi\_b) /4):.1f}")

# **4 Висновок**

Під час виконання лабораторної роботи навчився використовувати два алгоритми ітераційним способом для розв’язування систем алгебраїчних рівнянь СЛАР.