МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Лабораторна робота №6

з дисципліни: «Чисельні методи»

16 Варіант

СПЕЦІАЛЬНОСТІ 121 – Інженерія програмного забезпечення

Виконав: Левак О. О.

Група: ІТ-91

Перевірила: Тимофєєва Ю.С.

Київ 2020

**Лабораторна робота №6. Розв'язування систем лінійних рівнянь**

**Мета:** навчитися розв’язувати системи нелінійних рівнянь лінійними методами, а саме методом Гауса та прогону, та ітераційними методами, до яких відносяться метод простих ітерацій та метод Зейделя. Дізнатися та засвоїти на практиці методи обчислення визначника матриці та знаходження оберненої матриці.

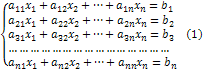
**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Серед чисельних методів алгебри існують прямі методи, у яких розв’язок обчислюється за скінченне фіксоване число операцій та ітераційні методи, у яких результат досягається в процесі послідовних наближень.

**2.1 Метод Гауса**

Розв'язок системи лінійних алгебраїчних рівнянь за методом Гаусса (метод послідовного виключення змінних) знаходиться за два етапи. На першому етапі вихідну систему рівнянь зводять до рівносильної їй системи трикутної форми — прямий хід методу Гаусса. На другому етапі, використовуючи систему трикутної форми, знаходимо значення невідомих величин (обернений хід методу Гаусса).

Прямий хід методу Гаусса: Нехай дано систему лінійних алгебраїчних рівнянь виду:



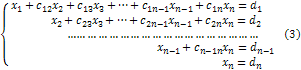
Нехай http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa23.gif (ведучий елемент), цього можна досягнути перестановкою рівнянь. Поділивши коефіцієнти першого рівняння системи (1) на http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa24.gif отримаємо:

http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa25.gif

де http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa26.gif.

Користуючись рівнянням (2), легко виключити із другого рівняння системи (1) невідому http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa27.gif. Для цього достатньо від другого рівняння системи (1) відняти рівняння (2), помножене на http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa28.gif; від третього рівняння системи (1) , відняти рівняння (2), помножене на http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa29.gif, і так далі.

Таким чином, ми отримуємо систему трикутної форми, яка має вигляд:



Обернений хід методу Гаусса**:** Цей етап полягає у знаходженні значень невідомих із системи, яку ми отримали на попередньому кроці. Його називають оберненим ходом тому, що спочатку з останнього рівняння знаходимо http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa31.gif:

http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa32.gif

Потім, підставляємо цю величину у http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa33.gif-ше рівняння — знаходимо http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa34.gif:

http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa35.gif

Далі підставляємо http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa31.gif і http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa34.gif в http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa36.gif-ге рівняння системи (3) — знаходимо http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa37.gifПродовжуючи даний процес далі, знайдемо шуканий розв'язок системи (1). Очевидно, що даний процес визначений однією формулою:

http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_gaussa39.gif

**2.2 Метод Зейделя**

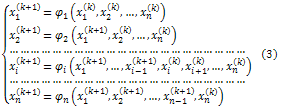
Нехай потрібно знайти розв'язок системи нелінійних алгебраїчних рівнянь (СЛАР) виду (1), використовуючи при цьому метод Зейделя.

Розв'язок нелінійних системи методом Зейделя

Для застосування даного методу систему (1), аналогічно, як і у методі простої ітерації, за допомогою еквівалентних перетворень необхідно привести до наступного вигляду (один із способів приведення системи (1) до виду (2) можна знайти за посиланням [Розв'язок систем нелінійних рівнянь методом ітерації](http://www.mathros.net.ua/rozvjazok-systemy-nelinijnyh-rivnjan-metodom-yteracii-poslidovnyh-nablyzhen.html)):

metod_zejdelja_snr2

Далі, задавши початкове наближення metod_zejdelja_snr3, реалізується ітераційний процес обчислення наближень до розв'язку системи за наступними формулами:



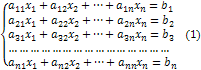
Даний ітераційний процес продовжується до тих пір, поки не буде виконуватись умова metod_zejdelja_snr8, де metod_zejdelja_snr91 - задана точність розв'язку. Тобто бачимо, що метод Зейделя, як і для випадку лінійних систем, відрізняється від методу простої ітерації лише тим, що обчислені на (*k+1*) -й ітерації значення невідомих metod_zejdelja_snr6, використовується для обчислення на тій самій ітерації нового значення невідомої metod_zejdelja_snr7. Такий прийом дозволяє збільшити швидкість збіжності ітераційного процесу, тобто забезпечити виконання умови зупинки при меншій кількості ітерацій. Швидкість збіжності методу Зейделя тим більша, чим більший порядок розглядуваної системи.

Також відмітимо, що в методі Зейделя успіх також залежить від вибору початкових значень невідомих: вони повинні бути досить близькі до точного рішення. В іншому випадку ітераційний процес може розходитися.

**2.3 Метод простої ітерації**

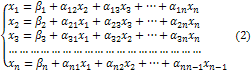
При великому числі невідомих у системі лінійних рівнянь, схема [методу Гауса](http://www.mathros.net.ua/metod-gaussa-rozvjazok-systemy-linijnyh-rivnjan-metodom-gaussa.html), яка дає точний розв'язок, стає досить складною. У цих умовах, для розв'язку системи лінійних рівнянь, доцільніше використовувати наближені чисельні методи. Одним з таких методів є метод простої ітерації (метод послідовних наближень).

Нехай дано систему лінійних рівнянь виду:



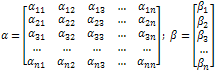
Припускаючи, що діагональні коефіцієнти http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii22.gif, розв'яжемо перше рівняння системи відносно http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii23.gif, друге — відносно http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii24.gif і так далі.

Тоді отримуємо систему, яка прийме наступного вигляд:



де http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii26.gif при http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii27.gif і http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii28.gif при http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii29.gif.

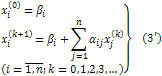
Увівши матриці:



систему (2) можна записати у матричній формі http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii31.gif. Систему (2') будемо розв'язувати методом послідовних наближень. За початкове (нульове) наближення приймемо стовпець вільних членів http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii32.gif. Далі, послідовно будуємо матриці-стовпці http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii33.gif (перше наближення), http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii34.gif (друге наближення) і так далі.

Таким чином http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii35.gif-ше наближення обчислюють за формулою http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii361.gif.  Якщо послідовність наближень http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii37.gif має границю http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii38.gif, то ця межа є розв'язком системи (2).

Запишемо формули наближень у розгорнутому вигляді:



Метод послідовних наближень, визначений формулою (3) чи (3'), має назву методу простої ітерації. При використанні даного методу немає необхідності за нульове наближення http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii40.gif брати стовпець вільних членів. Бо збіжність процесу ітерації залежить тільки від властивостей матриці http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii411.gif, якщо цей процес збігається при виборі будь-якого початкового наближення, то він буде сходитись до того ж граничного вектора і при любому іншому виборі цього початкового наближення. Тому початковий вектор http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii40.gif в процесі ітерації може бути вибраний довільно.

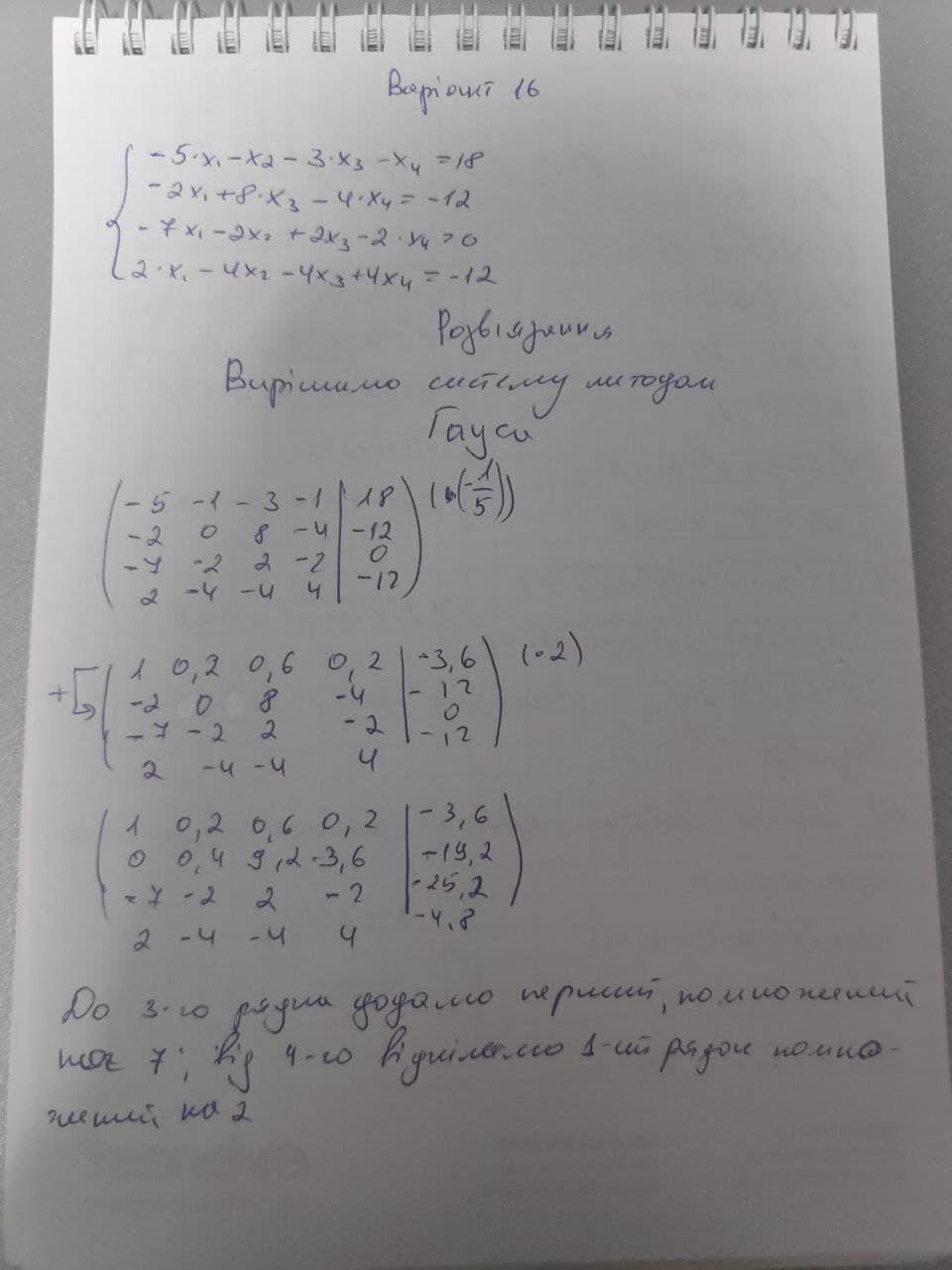
Достатня умова збіжності процесу ітерації наступна: якщо для приведеної системи (2) виконується хоча б одна з умов:

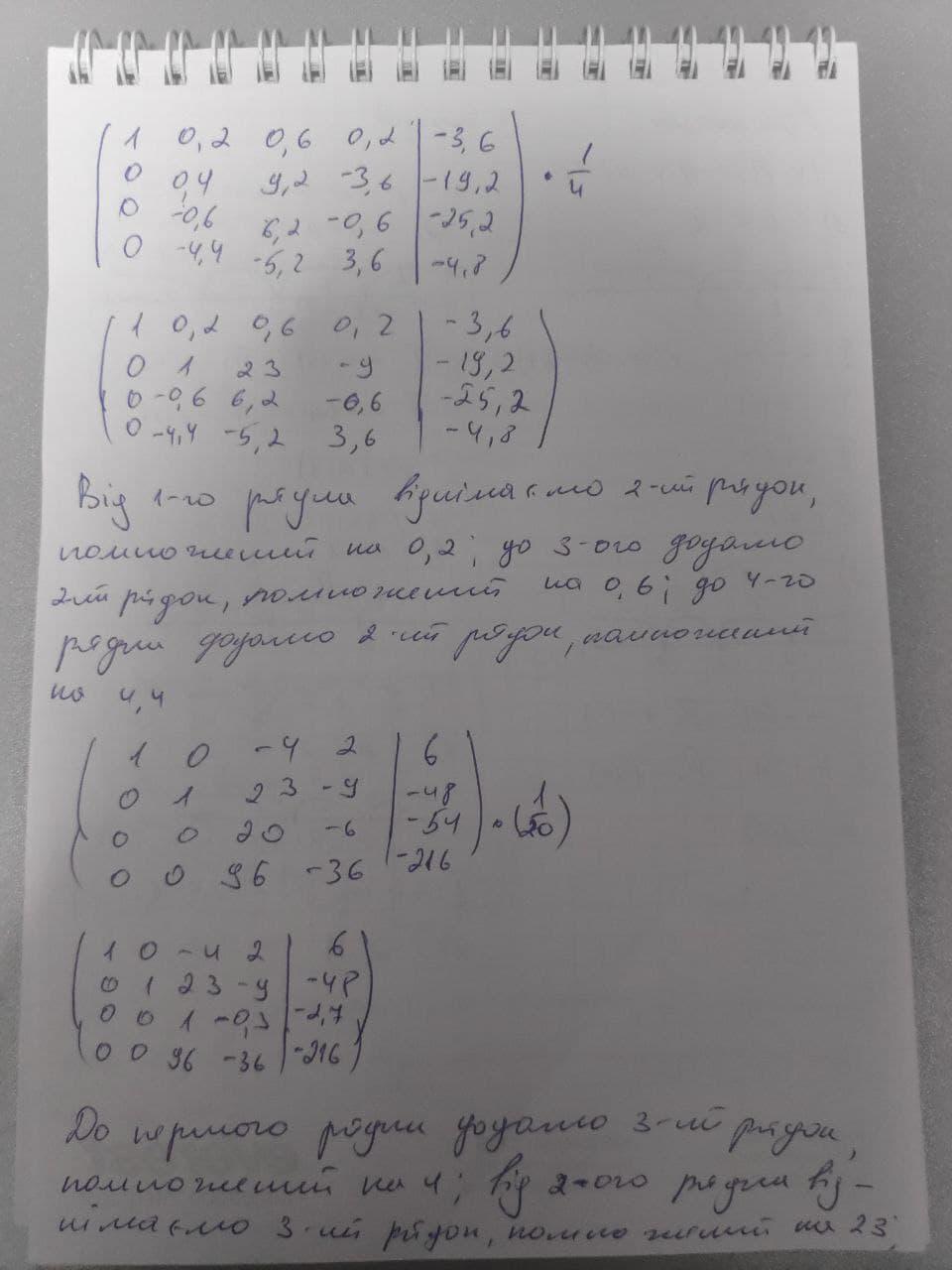
http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii42.gif або http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii43.gif

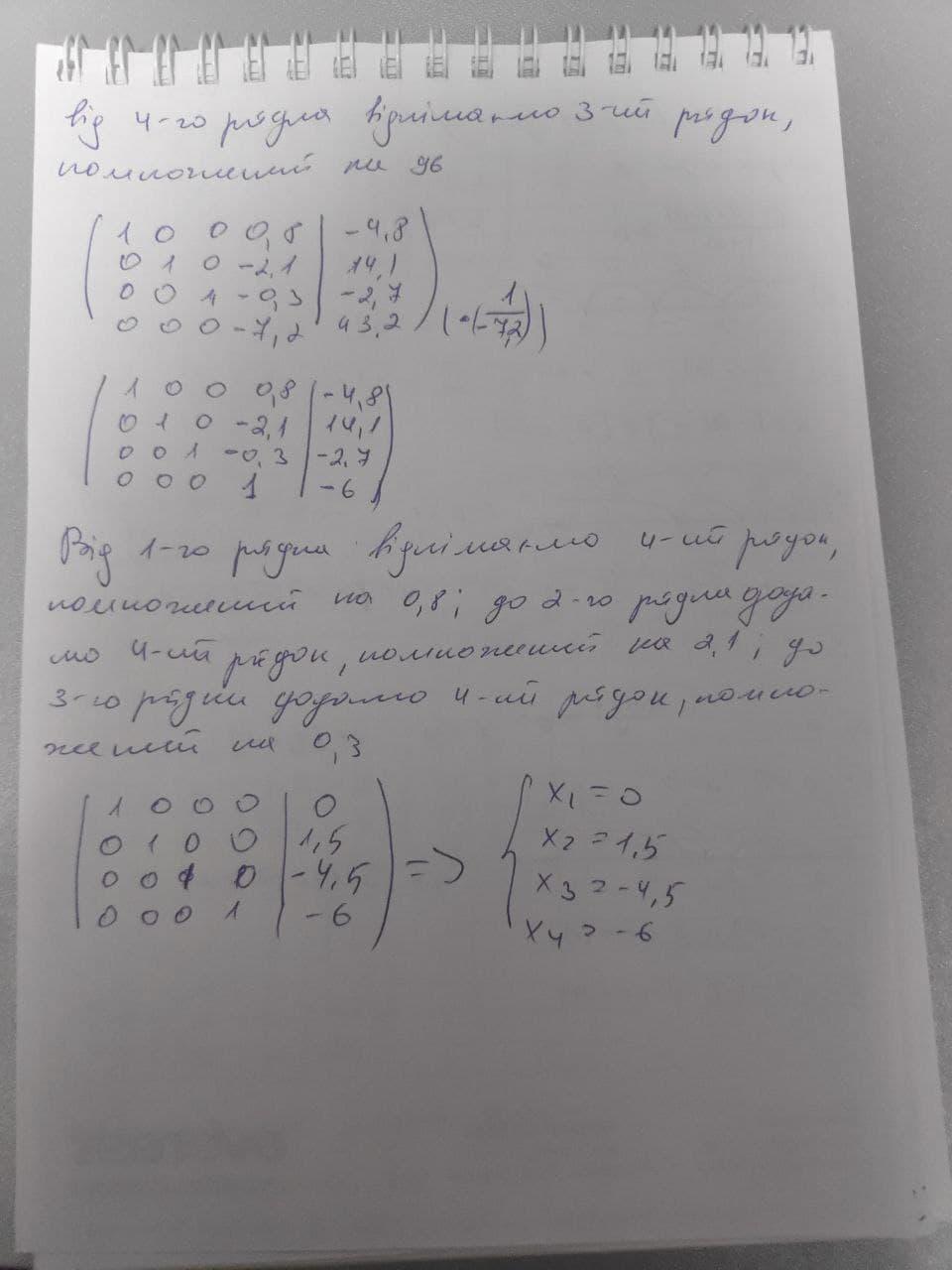
то процес ітерації (3) збігається до єдиного розв'язку цієї системи, незалежно від початкового наближення.

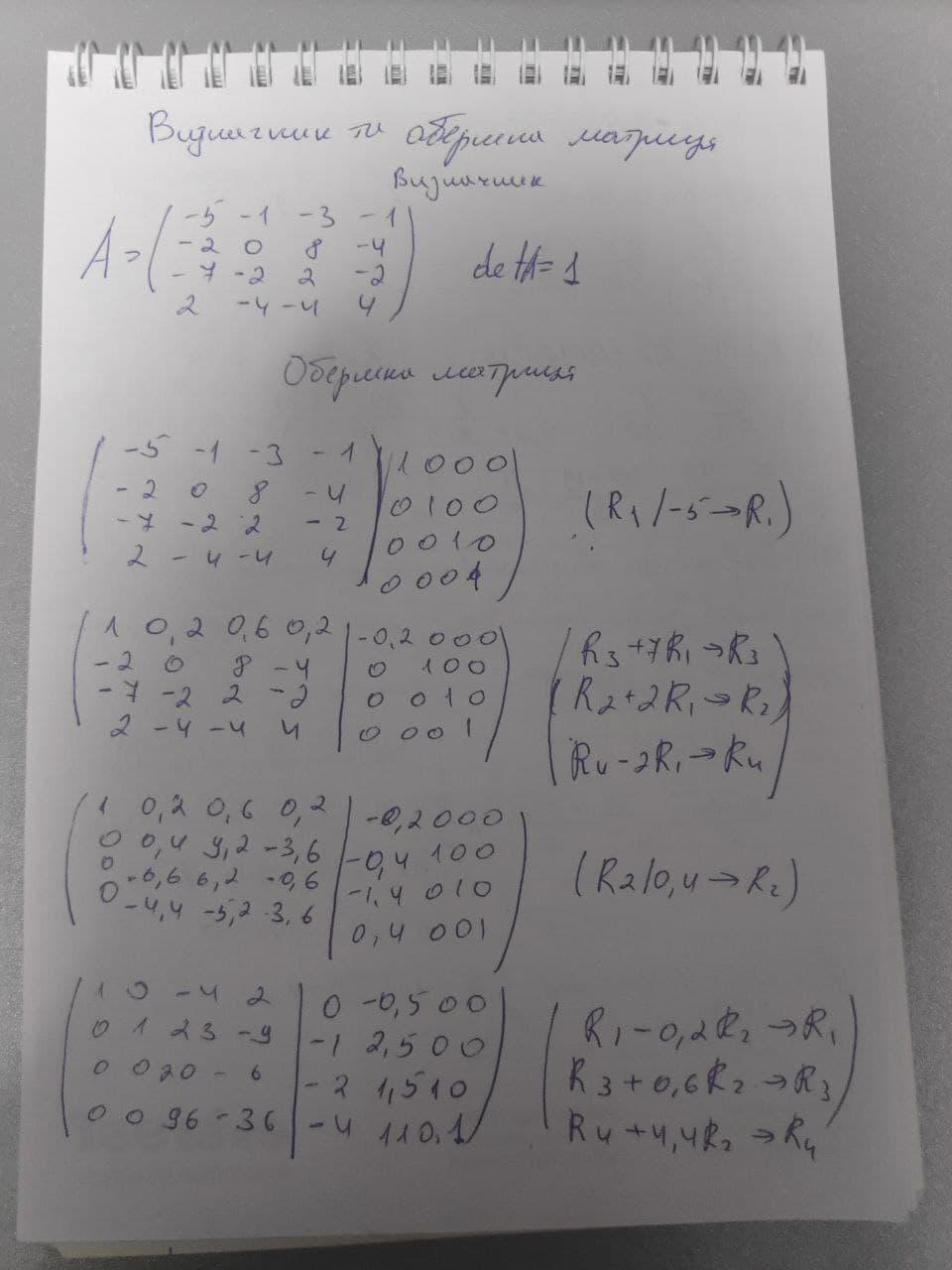
Зауваження**:** ітераційний процес методу простої ітерації необхідно продовжувати до тих пір, поки не буде виконуватись умова http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii13.gif, де http://www.mathros.net.ua/wp-content/uploads/2012/07/metod_iteracii2.gif — задана точність обчислювального процесу.

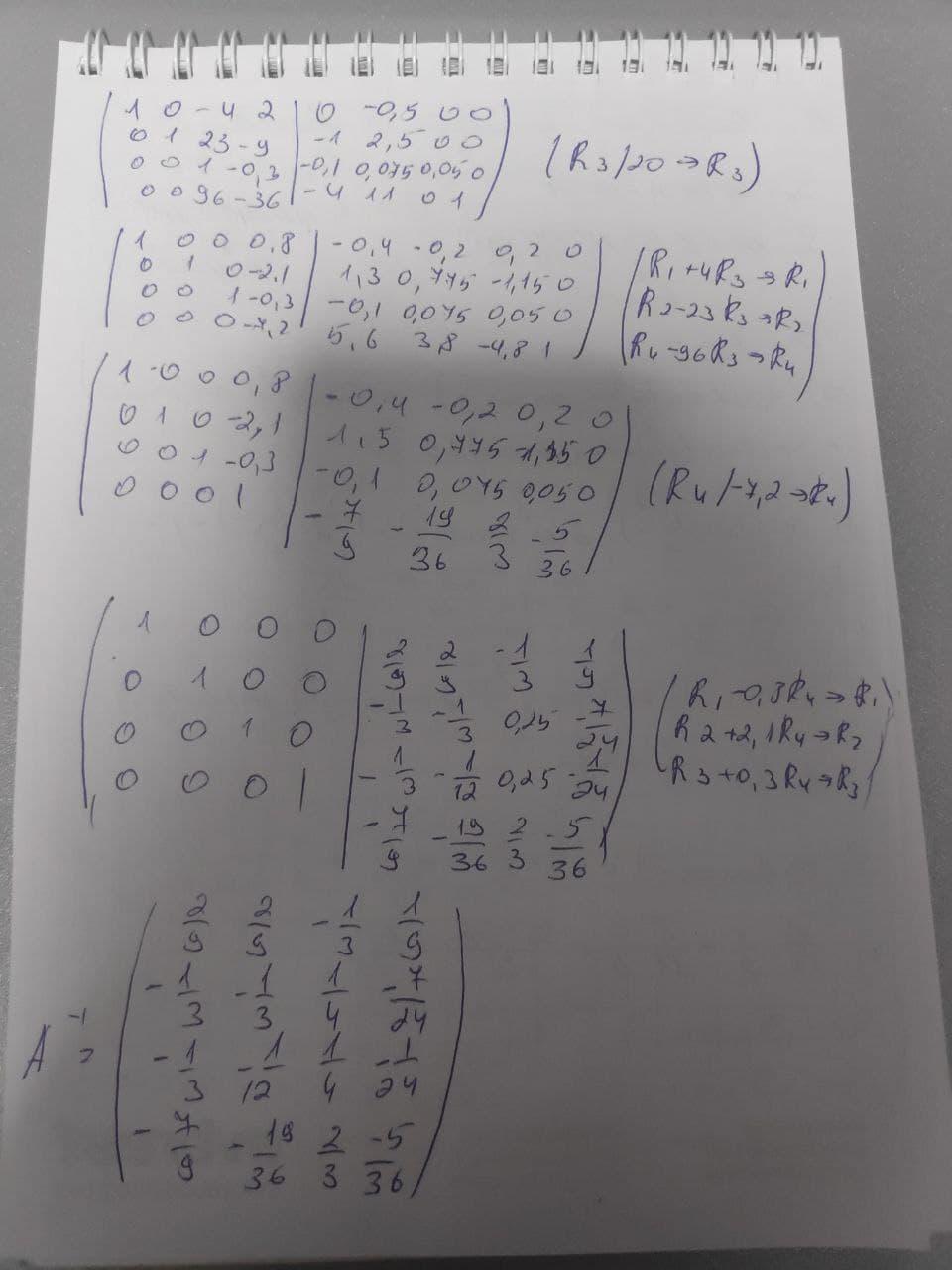
**РОЗВ’ЯЗОК ЗАДАЧІ В АНАЛІТИЧНІЙ ФОРМІ**

****

****

****

****

****

**ЛІСТИНГ ПРОГРАМИ РОЗВ’ЯЗУВАННЯ ЗАДАЧІ**

**import numpy.linalg**

**from math import sqrt**

**import numpy as np**

**np.set\_printoptions(suppress=True)**

**Array1 = [**

**[-5., -1., -3., -1., 18.],**

**[-2., 0., 8., -4., -12.],**

**[-7., -2., 2., -2., 0.],**

**[2., -4., -4., 4., -12.]]**

**Array1\_2 = [**

**[-5., -1., -3., -1.],**

**[-2., 0., 8., -4.],**

**[-7., -2., 2., -2.],**

**[2., -4., -4., 4.]]**

**Array2 = np.array([**

**[18, -9, 0, 0, 0],**

**[2, -9, -4, 0, 0],**

**[0, -9, 21, -8, 0],**

**[0, 0, -4, -10, 5],**

**[0, 0, 0, 7, 12]])**

**result2 = np.array([-81, 71, -39, 64, 3])**

**Array2\_2 = np.array([**

**[8, -2, 0, 0],**

**[-1, 6, -2, 0],**

**[0, 2, 10, -4],**

**[0, 0, -1, 6]])**

**# a = [-1, 2, -1, 0]**

**# b = [8, 6, 10, 6]**

**# c = [-2, -2, -4, 0]**

**# d = [6, 3, 8, 5]**

**a = [2, -9, -4, 7, 0]**

**b = [18, -9, 21, -10, 12]**

**c = [-9, -4, -8, 5, 0]**

**d = [-81, 71, -39, 64, 3]**

**Array3 = [**

**[10, 1, 1],**

**[2, 10, 1],**

**[2, 2, 10]]**

**result3 = [12, 13, 14]**

**# Array3 = [**

**# [21, -6, -9, -4],**

**# [-6, 20, -4, 2],**

**# [-2, -7, -20, 3],**

**# [4, 9, 6, 24]]**

**# result3 = [127, -144, 236, -5]**

**e = 0.01**

**def Gauss(m):**

**n = len(m)**

**# прямий хід (результат - верхня трикутна матриця)**

**for k in range(n - 1): # обираємо провідний рядок**

**m = MaxRowChange(m, k) # рядок з найбільшим по модулю елементом на верх**

**for i in range(k + 1, n):**

**div = m[i][k] / m[k][k] # An1/A11**

**for j in range(k, n + 1):**

**m[i][j] -= div \* m[k][j] # m[k][j] - елемент провідного рядка**

**if IsSingular(m):**

**print("Визначник дорівнює нулю => система має безліч розв'язків")**

**return**

**# зворотній хід**

**x = [0 for i in range(n)] # [0, 0, 0, 0]**

**for k in range(n - 1, -1, -1):**

**b = m[k][-1]**

**sumOfRowOnX = sum([m[k][j] \* x[j] for j in range(k + 1, n)])**

**x[k] = (b - sumOfRowOnX) / m[k][k]**

**if abs(x[k] == 0.0): x[k] = 0.0**

**return x**

**def MaxRowChange(m, col):**

**max\_element = m[col][col]**

**max\_row = col**

**for i in range(col + 1, len(m)):**

**if abs(m[i][col]) > abs(max\_element):**

**max\_element = m[i][col]**

**max\_row = i**

**if max\_row != col:**

**m[col], m[max\_row] = m[max\_row], m[col]**

**return m**

**def IsSingular(m):**

**for i in range(len(m)):**

**if not m[i][i]:**

**return True**

**return False**

**def Determinant(m):**

**d = 0**

**for i in range(0, len(m)):**

**for j in range(0, len(m)):**

**if (i == j):**

**d += m[i][j]**

**return d**

**def GaussInverse(M):**

**n = len(M)**

**I = [] # створити одиничну матрицю**

**for i in range(n): # заповнити одиничну матрицю**

**L = []**

**for j in range(n):**

**if i == j: L.append(1)**

**else: L.append(0)**

**I.append(L)**

**for i in range(n):**

**j = i**

**while M[j][i] == 0: # шукаємо перший рядок із ненульовим значенням**

**j += 1**

**# поміняти місцями рядок i та рядок j**

**M[i], M[j] = M[j], M[i]**

**I[i], I[j] = I[j], I[i]**

**const = M[i][i]**

**for j in range(n): # ділимо на const, щоб мати 1 по діагоналі**

**I[i][j] = I[i][j] / const**

**M[i][j] = M[i][j] / const**

**for j in range(i+1,n): # операція, щоб мати 0 в i-му стовпці всіх рядків нижче i-го**

**const = M[j][i]**

**for k in range(n):**

**I[j][k] -= const \* I[i][k]**

**M[j][k] -= const \* M[i][k]**

**# маємо матрицю верхнього трикутника в M з одиницями по діагоналі**

**for i in range(n):**

**for j in range(i):**

**const = M[j][i]**

**for k in range(n):**

**I[j][k] -= const \* I[i][k]**

**M[j][k] -= const \* M[i][k]**

**return I**

**def Progonka(a,b,c,d):**

**M = len(d)**

**p = []**

**q = []**

**x = []**

**for i in range(M):**

**x.append(0)**

**p.append((-1)\*c[0]/b[0])**

**q.append(d[0]/b[0])**

**for i in range(1, M):**

**p.append(((-1)\*c[i]) / (b[i]+a[i-1]\*p[i-1]))**

**q.append((d[i]-a[i-1]\*q[i-1]) / (b[i]+a[i-1]\*p[i-1]))**

**x[M-1] = q[M-1]**

**for i in reversed(range(M-1)):**

**x[i] = p[i]\*x[i+1]+q[i]**

**print("P:", p)**

**print("Q:", q)**

**return x**

**def Zeidel(M, b, e):**

**n = len(M)**

**x = [.0 for i in range(n)]**

**converge = False**

**while not converge:**

**xNew = np.copy(x)**

**for i in range(n):**

**sum1 = sum(M[i][j] \* xNew[j] for j in range(i))**

**sum2 = sum(M[i][j] \* x[j] for j in range(i + 1, n))**

**xNew[i] = (b[i] - sum1 - sum2) / M[i][i]**

**converge = abs(sum(xNew[i] - x[i] for i in range(n))) <= e**

**x = xNew**

**return x**

**def Iteration(M, results, tolerance):**

**diag = np.diag(M)**

**div = M - np.diagflat(diag) # по діагоналі нулі**

**xNew = np.zeros\_like(diag)**

**x = np.ones\_like(diag)**

**while np.any(np.abs(x-xNew) > tolerance\*np.abs(x+xNew)):**

**x = xNew**

**xNew = (results - np.dot(div, x)) / diag**

**return x**

**print("Гаус: ")**

**print(Gauss(Array1))**

**print("Визначник матриці: ")**

**print(Determinant(Array1))**

**print("Обернена матриця: ")**

**print(np.array(GaussInverse(Array1\_2)))**

**print("Метод Прогону: ")**

**print(Progonka(a, b, c, d))**

**print(np.linalg.solve(Array2, result2))**

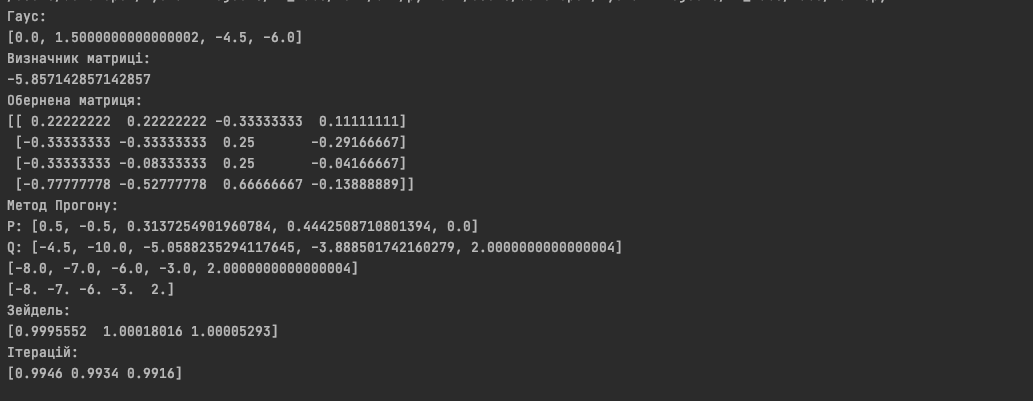
**print("Зейдель: ")**

**print(Zeidel(Array3, result3, e))**

**print ("Ітерацій: ")**

**print(Iteration(Array3, result3, e))**

**РЕЗУЛЬТАТИ ВИКОНАННЯ ПРОГРАМИ**

****

**Висновки :**

У ході лабораторної роботи я навчилася розв’язувати системи лінійних рівнянь за допомогою методів Гаусса, Зейделя та методу простих ітерацій, результати виконання програми збігаються з аналітичними розв’язками.