МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №1

з дисципліни
«Чисельні методи»

Виконав:
Студент групи ШІ-22
Михальчук Антон
Перевірила:
доцент кафедри СШІ
Гентош Леся Ігорівна

Львів - 2025 р.

Лабораторна робота № 1.

Тема: Метод Гауса для розв’язування систем лінійних алгебричних рівнянь.

Мета – засвоїти основні способи практичного використання методу Гауса.

Варіант 15

Постановка завдання

- Скласти програму, яка реалізує метод Гауса для розв’язування СЛАР з постовпцевим вибором головного елемента.
- Програма повинна обчислювати визначник.
- Додати модифікацію з вибором головного елемента з усієї невиключенної матриці.
- Перевірити виконання наданних завдань.
- Перевірити виконання для матриць n>1000.
- Знайти обчислювальну, просторову та часову складність.
- Перевірити на достовірність.
- Перевірити на точність.
- Перевірити на стійкість.

Львів - 2025 р.

Код програмної реалізації

```python
import numpy as np

n = 10000
zero_percentage = 0.1
input_id = 6

random_mask = np.random.rand(n, n + 1)
zero_mask = random_mask < zero_percentage
matrix = np.random.uniform(-1000, 1000, size=(n, n + 1))
matrix[zero_mask] = 0

if n > 1000:
    np.savez_compressed(f"inputs/input{input_id}.npz", a=matrix[:, :n], b=matrix[:, n])
    print(f"Matrices saved to file inputs/input{input_id}.npz")
else:
    with open(f"inputs/input{input_id}.txt", 'w') as f:
        f.write(f"{n}\n")
        np.savetxt(f, matrix)
    print(f"Matrix saved to file inputs/input{input_id}.txt")
```

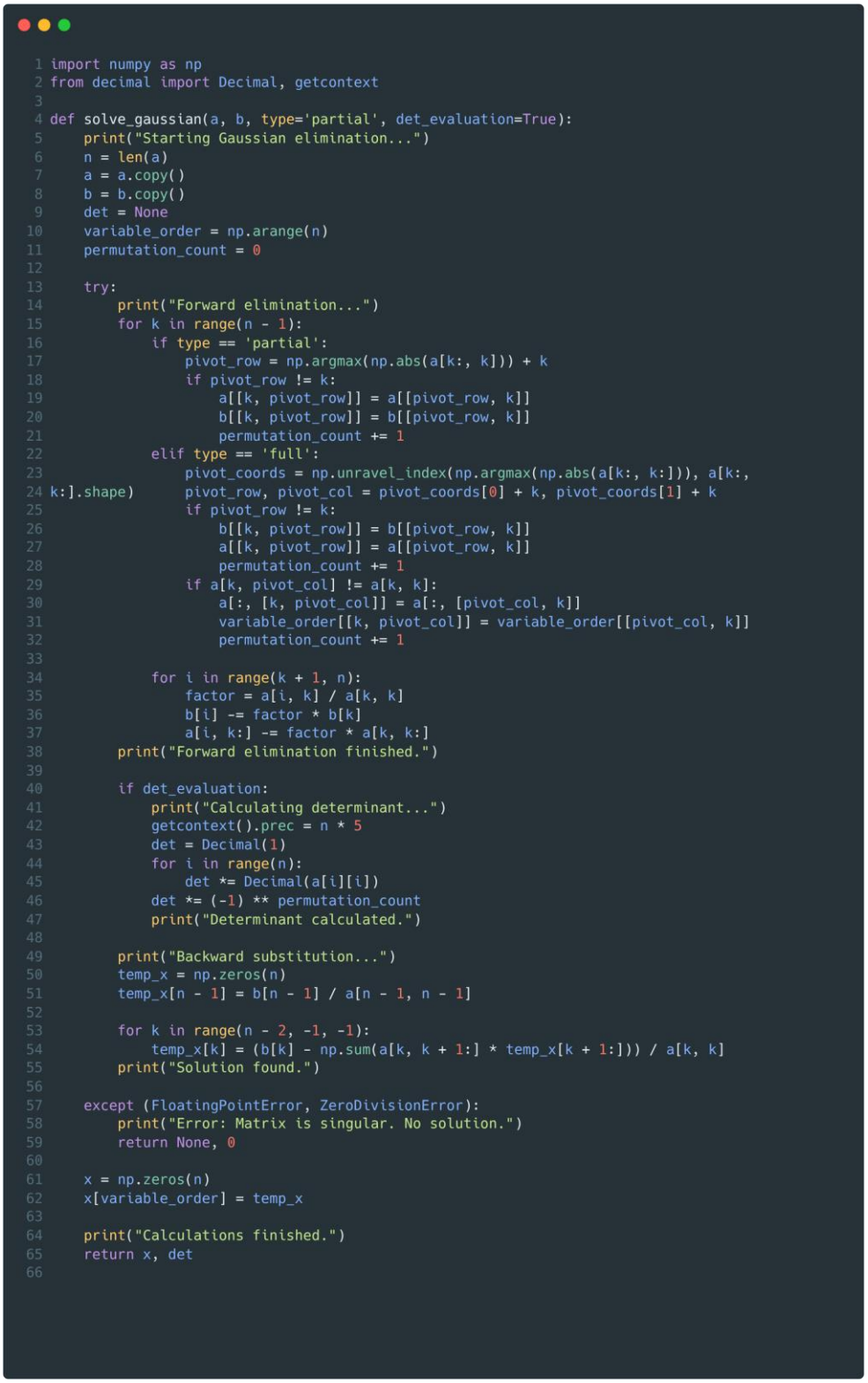
create_inputs.py

Генератор випадкових СЛАР з довільним параметром n та визначенням відсотку нульових елементів:

```python
import warnings
import time
import numpy as np
from gaussian import solve_gaussian
from data_handler import read_sole_data, method_evaluation,
save_solution

np.seterr(divide='raise', invalid='raise')
warnings.simplefilter('error', RuntimeWarning)
input_id = 6
decimal_places = 60
pivoting_type = "full"
is_evaluate = True

def main():
    a, b = read_sole_data(input_id)
    if a is None or b is None:
        return

    start_time = time.time()
    x, det = solve_gaussian(a, b, type=pivoting_type,
det_evaluation=is_evaluate)
    end_time = time.time()
    execution_time = end_time - start_time

    if x is not None:
        if is_evaluate:
            method_evaluation(a, x, b, det, execution_time, input_id,
pivoting_type, decimal_places)
        else:
            print(f"Execution time: {execution_time}\n")

    save_solution(x, input_id, pivoting_type, decimal_places)

if __name__ == '__main__':
    main()
```

main.py

У цьому файлі відбувається налаштування чисельного експерименту. Існує можливість обрати: метод (постовпцевий, чи повний вибір головного елементу) та чи проводити оцінку, чи лише видати обрахунок.



```python
import numpy as np
from decimal import Decimal, getcontext

def solve_gaussian(a, b, type='partial', det_evaluation=True):
    print("Starting Gaussian elimination...")
    n = len(a)
    a = a.copy()
    b = b.copy()
    det = None
    variable_order = np.arange(n)
    permutation_count = 0

    try:
        print("Forward elimination...")
        for k in range(n - 1):
            if type == 'partial':
                pivot_row = np.argmax(np.abs(a[k:, k])) + k
                if pivot_row != k:
                    a[[k, pivot_row]] = a[[pivot_row, k]]
                    b[[k, pivot_row]] = b[[pivot_row, k]]
                    permutation_count += 1
            elif type == 'full':
                pivot_coords = np.unravel_index(np.argmax(np.abs(a[k:, k:])), a[k:,
k:].shape)
                pivot_row, pivot_col = pivot_coords[0] + k, pivot_coords[1] + k
                if pivot_row != k:
                    b[[k, pivot_row]] = b[[pivot_row, k]]
                    a[[k, pivot_row]] = a[[pivot_row, k]]
                    permutation_count += 1
                if a[k, pivot_col] != a[k, k]:
                    a[:, [k, pivot_col]] = a[:, [pivot_col, k]]
                    variable_order[[k, pivot_col]] = variable_order[[pivot_col, k]]
                    permutation_count += 1

            for i in range(k + 1, n):
                factor = a[i, k] / a[k, k]
                b[i] -= factor * b[k]
                a[i, k:] -= factor * a[k, k:]
        print("Forward elimination finished.")

        if det_evaluation:
            print("Calculating determinant...")
            getcontext().prec = n * 5
            det = Decimal(1)
            for i in range(n):
                det *= Decimal(a[i][i])
            det *= (-1) ** permutation_count
            print("Determinant calculated.")

        print("Backward substitution...")
        temp_x = np.zeros(n)
        temp_x[n - 1] = b[n - 1] / a[n - 1, n - 1]

        for k in range(n - 2, -1, -1):
            temp_x[k] = (b[k] - np.sum(a[k, k + 1:] * temp_x[k + 1:])) / a[k, k]
        print("Solution found.")

    except (FloatingPointError, ZeroDivisionError):
        print("Error: Matrix is singular. No solution.")
        return None, 0

    x = np.zeros(n)
    x[variable_order] = temp_x

    print("Calculations finished.")
    return x, det

```

gaussian.py

Файл з функцією розв’язання СЛАР методом Гаусса з вибором режиму: постовпцевий та повний

Львів - 2025 р.

```python
import numpy as np
from decimal import Decimal
from gaussian import solve_gaussian
import os

def read_sole_data(input_id):
    npz_path = f"inputs/input{input_id}.npz"
    txt_path = f"inputs/input{input_id}.txt"

    try:
        if os.path.exists(npz_path):
            print(f"Reading data from file {npz_path}...")
            data = np.load(npz_path)
            if 'a' in data and 'b' in data:
                a = data['a']
                b = data['b']
                print("Data loaded successfully.")
                return a, b
            else:
                print("Error: NPZ file does not contain keys 'a' and 'b'.")
                return None, None

        elif os.path.exists(txt_path):
            print(f"Reading data from file {txt_path}...")
            full_data = np.genfromtxt(txt_path, skip_header=1, delimiter=None,
filling_values=0)
            if full_data.ndim < 2:
                print("Error: Not enough data for matrix and vector.")
                return None, None
            a = full_data[:, :-1].copy()
            b = full_data[:, -1].copy()
            if a.shape[0] != a.shape[1] or a.shape[0] != b.shape[0]:
                print("Error: dimensions do not match.")
                return None, None
            print("Data loaded successfully.")
            return a, b

        else:
            print("Error: Data file not found.")
            return None, None

    except Exception as error:
        print(f"Error reading file: {error}")
        return None, None
```

data_handler.py
read_sole_data - функція для зчитування матриці A та вектора b з файлів формату .txt або .npz

Львів - 2025 р.

```python
def method_evaluation(a, x, b, det, execution_time, input_id, pivoting_type, decimal_places):
    print("Starting method evaluation...")
    print("Step 1: Calculating benchmark determinant (np.linalg.det)...")
    try:
        det_benchmark = np.linalg.det(a)
        print("Benchmark determinant successfully calculated.")
    except RuntimeWarning as e:
        det_benchmark = None
        print(f"Error calculating benchmark determinant: {e}")

    print("Step 2: Calculating benchmark solution (np.linalg.solve)...")
    try:
        x_benchmark = np.linalg.solve(a, b)
        print("Benchmark solution successfully calculated.")
    except np.linalg.LinAlgError:
        x_benchmark = None
        print("Error: Matrix is singular. Cannot calculate benchmark solution.")

    print(f"Step 3: Opening file to write results: evaluations/evaluation{input_id}_{pivoting_type}.txt")
    with open(f"evaluations/evaluation{input_id}_{pivoting_type}.txt", 'w') as f:
        f.write(f"Execution time: {execution_time}\n")
        if det is not None:
            f.write(f"Determinant: {det:.{decimal_places}g}\n")
        if det_benchmark is not None:
            det_benchmark = Decimal(str(det_benchmark))
            f.write(f"Benchmark determinant: {det_benchmark:.{decimal_places}g}\n")
            abs_error_det = abs(det - det_benchmark)
            rel_error_det = abs_error_det / abs(det_benchmark)
            f.write(f"Absolute error of determinant: {abs_error_det}\n")
            f.write(f"Relative error of determinant: {rel_error_det}\n")

        if x is not None:
            print("Step 4: Calculating stability error...")
            epsilon = 1e-8
            b_perturbed = b + epsilon * np.random.randn(*b.shape)
            x_perturbed, det_perturbed = solve_gaussian(a, b_perturbed, type=pivoting_type,
det_evaluation=False)
            stability_error = np.linalg.norm(x_perturbed - x)
            f.write(f"Stability error: {stability_error}\n")
            print("Stability error calculated.")

        print("Step 5: Calculating condition number of matrix Cond(A)...")
        cond_a = np.linalg.cond(a.copy())
        f.write(f"Cond A: {cond_a}\n")
        print("Condition number calculated.")
        if x is not None and x_benchmark is not None:
            print("Step 6: Calculating solution errors (absolute and relative)...")
            abs_error_solution = np.linalg.norm(x - x_benchmark)
            rel_error_solution = abs_error_solution / np.linalg.norm(x_benchmark)
            f.write(f"Absolute error of solution: {abs_error_solution}\n")
            f.write(f"Relative error of solution: {rel_error_solution}\n")
            print("Solution errors calculated.")
```

data_handler.py

method_evaluation - функція, що проводить оцінку метода, завдяки порівнянню похибки розв’язку з “золотим стандартом” відповідними методами бібліотеки numpy.

У файл evaluation записується дискримінант, абсолютна та відносна похибка дискримінанту, стійкість методу, абсолютна та відносна похибка розв’язків.

Львів - 2025 р.

```python
def save_solution(x, input_id, pivoting_type, decimal_places):
    n = len(x)
    output_dir = "outputs"
    os.makedirs(output_dir, exist_ok=True)
    if x is not None:
        if n > 1000:
            npz_path = f"{output_dir}/output{input_id}_{pivoting_type}.npz"
            np.savez_compressed(npz_path, x=x)
            print(f"Solution saved to {npz_path}")
        else:
            txt_path = f"{output_dir}/output{input_id}_{pivoting_type}.txt"
            np.savetxt(txt_path, x, fmt=f'%.{decimal_places}g')
            print(f"Solution saved to {txt_path}")
    else:
        txt_path = f"{output_dir}/output{input_id}_{pivoting_type}.txt"
        with open(txt_path, 'w') as f:
            f.write("Matrix is singular.")
        print(f"Matrix is singular. Saved to {txt_path}")
```

data_handler.py
save_solution – зберігає розв’язок у форматі .txt або .npz залежно від розміру СЛАР.

Аналіз чисельних експериментів

Перше завдання

input1.txt

```
4
1 -4 -3  8  11.55
2 -6 -4  2  7.35
3 -8  1 -4 -3.35
4  2  2 -6 -4.55
```

Розв’язок:

output1_partial.txt

```
0.830303030303030542569331373670138418674468994140625
0.10227272727272733454650932571894372813403606414794921875
-1.0833333333333332593184650249895639717578887939453125
0.9848484848484849729643997306993696838617324829101562 5
```

output1_full.txt

```
0.830303030303030542569331373670138418674468994140625
0.10227272727272733454650932571894372813403606414794921875
-1.0833333333333330372738600999582558870315551757 8125
0.9848484848484850839867021932150237262248992919921875
```

Оцінка:

evaluation1_partial.txt

```
Execution time: 0.00013685226440429688
Determinant: 1056.0000000000000427
Benchmark determinant: 1056.0
Absolute error of determinant: 4.27E-14
Relative error of determinant: 4.0435606060606060606E-17
Stability error: 5.721207306939229e-09
Cond A: 5.94223498672021
Absolute error of solution: 2.2247786310271853e-16
Relative error of solution: 1.3193735948546014e-16
```

evaluation1_full.txt

```
Execution time: 0.00017642974853515625
Determinant: 1056.0000000000000853
Benchmark determinant: 1056.0
Absolute error of determinant: 8.53E-14
Relative error of determinant: 8.077651515151515515E-17
Stability error: 3.874126434583939e-09
Cond A: 5.94223498672021
Absolute error of solution: 3.3335590258932494e-16
Relative error of solution: 1.9769201727823663e-16
```

Друге завдання

Розв’язок:

output2_partial.txt

```
0
2.000000000000000444089209850062616169452667236328125
0.3333333333333332593184650249895639717578887939453125
-1.5
```

output2_full.txt

```
-1.714028529245855862902556800821207178785634785622357667556 06e-16
2
0.3333333333333333703407674875052180141210556030273437 5
-1.5
```

Оцінка:

Львів - 2025 р.

evaluation2_partial.txt

```
Execution time: 0.0001380443572998047
Determinant: -954.0000000000005861
Benchmark determinant: -954.0000000000013
Absolute error of determinant: 1.24139E-12
Relative error of determinant: 1.3012473794549248515E-15
Stability error: 3.409069923942645e-09
Cond A: 6.893516361135211
Absolute error of solution: 1.4802973661668753e-16
Relative error of solution: 5.869248202606501e-17
```

evaluation2_full.txt

```
Execution time: 0.000209808349609375
Determinant: -953.99999999999985980
Benchmark determinant: -954.0000000000013
Absolute error of determinant: 1.44020E-12
Relative error of determinant: 1.5096436058700189072E-15
Stability error: 3.6060069175711164e-09
Cond A: 6.893516361135211
Absolute error of solution: 5.581920433062216e-16
Relative error of solution: 2.2131821090567051e-16
```

Третє завдання

Розв’язок:

output3_partial.txt

```
0.66666666666666662965923251249478198587894439697265625
-2
-0.75
```

output3_full.txt

```
0.66666666666666662965923251249478198587894439697265625
-2
-0.75
```

Оцінка:

evaluation3_partial.txt

```
Execution time: 0.0001232624053955078
Determinant: 12
Benchmark determinant: 12.0
Absolute error of determinant: 0.0
Relative error of determinant: 0
Stability error: 1.591319166056226e-08
Cond A: 14.3633871971338
Absolute error of solution: 0.0
Relative error of solution: 0.0
```

evaluation3_full.txt

```
Execution time: 0.00013637542724609375
Determinant: 12
Benchmark determinant: 12.0
Absolute error of determinant: 0.0
Relative error of determinant: 0
Stability error: 3.6500861034149777e-09
Cond A: 14.3633871971338
Absolute error of solution: 0.0
Relative error of solution: 0.0
```

Четверте завдання

Розв’язок:

output4_partial.txt

```
Matrix is singular.
```

output4_full.txt

```
Matrix is singular.
```

Метод визначив матрицю, як вироджену, після отримання ділення на 0. Вірно стверджувати, що якщо абсолютно найбільшим елементом у стовпці, або в невиключенній матриці є нуль, то така матриця є виродженою без єдиного розв’язку.

Обчислювальна складність методу Гауса:

Прямий хід:

$$\sum_{k=0}^{n-2}\sum_{i=k+1}^{n-1} 2(n-k) = \frac{2}{3}n(n+1)(n-1) = \frac{2}{3}n^3 - \frac{2}{3}n$$

Постовпцевий вибір:

$$\sum_{k=0}^{n-2}(n-k-1) = \frac{n^2}{2} - \frac{n}{2}$$

Повний вибір:

$$\sum_{k=0}^{n-2}(n-k-1)^2 = \frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6}$$

Зворотний хід:

$$\sum_{k=0}^{n-2}\big((n-k)+(n-k-1)\big) = n^2 - 1$$

Обрахунок детермінанту:

$$n + 2$$

Обчислювальна складність методу Гауса з постовпцевим вибором:

$$\frac{2}{3}n^3 + \frac{3n^2}{2} - \frac{n}{6}$$

Обчислювальна складність методу Гауса з повним вибором:

$$n^3 + \frac{n^2}{2} + \frac{n}{2}$$

Просторова складність методу Гауса:

Оригінальні матриця та вектор:

$$O(n^2) + O(n)$$

Копіювання матриці та вектора:

$$O(n^2) + O(n)$$

Львів - 2025 р.

Допоміжні вектори: variable_order, temp_x, x:

$$O(n) + O(n) + O(n)$$

Загалом у байтах:

$$8n^2 + 8n + 8n^2 + 8n + 4n + 8n + 8n = 16n^2 + 36n$$

Часова складність методу Гауса:

$$T(n) = \tau f(n)$$

τ можна знайти експериментально, порахуємо час виконання алгоритмів при n=1000:

При частковому виборі:

$\tau = \frac{1.595668077468872}{\frac{2}{3}1000^3 + \frac{3 \cdot 1000^2}{2} - \frac{1000}{6}}$= 2.3881294220360823e-09

При повному виборі:

$\tau = \frac{2.378239870071411}{1000^3 + \frac{1000^2}{2} + \frac{1000}{2}}$= 2.377050156468099e-09

Львів - 2025 р.

Обчислення для n>1000:

Задля оптимізації вводу та виводу данних, було прийнято рішення, для усіх n>1000 генерувати вхідні данні в спеціальному форматі npz для коректної роботи з бібліотеками numpy. Це дозволило розширити клас СЛАР які може розв’язати метод на доступній обчислювальній техніці, адже обробка данних з txt формату витрачала більше оперативної пам’яті, аніж було доступно.

Після оптимізації, можна зосередитися на тому, які ресурси потрібні саме для методу Гауса. Наразі доступно близько 16 гігабайтів оперативної пам’яті.

Прогнозоване обмеження на n:

$$16n^2 + 36n < 16000000000$$

З чого виводимо, що приблизним обмеженням доступної оперативної пам’яті повинна бути СЛАР з розміром 31600.

Завдячуючи, часовій складності можна спрогнозувати, що обчислення такого СЛАР займе з частковим повним вибором близько 9 годин та 16 годин відповідно.

Проведемо експеримент з n=10000:

Прогнозоване використання пам’яті: $16 \cdot 10000^2 + 36 \cdot 10000 = 1600360000$ байт. $\approx$ 1.6 гб.

Прогнозаваний час (частковий): 17 хвилин.

Прогнозований час (повний): 31 хвилин.

Львів - 2025 р.

Результати експерименту

evaluation6_full.txt

```
Execution time: 1217.6422336101532
Determinant: -1.902342753791177039242058819761047797874534841089621669
Stability error: 3.036118267069972e-09
Cond A: 94557.24950780396
Absolute error of solution: 1.182704764871899e-08
Relative error of solution: 2.2689640955740917e-11
```

evaluation6_partial.txt

```
Execution time: 535.5828998088837
Determinant: -1.902342753757701535801704696386221242580075605077523
Stability error: 4.345994486380594e-09
Cond A: 94557.24950780396
Absolute error of solution: 3.878990641780384e-09
Relative error of solution: 7.441663172990496e-12
```

Реальний час виконання виявився меншим за прогнозований 8 хвилин проти 17 та 20 хвилин проти 31. Однак, вдалося приблизно спрогнозувати використану оперативну пам’ять. Протягом всього виконання в середньому спостерігалася витрата 1.6 ГБ оперативної пам’яті:

Name	ID	CPU	Memory	Disk Re
pycharm	6967	0.2 %	1.6 GB	
python	13848	6.3 %	1.6 GB	
gnome-shell	3164	0.5 %	255.0 MB	
chrome	4154	0.1 %	217.4 MB	
chrome --type=renderer --crashpad-handler-pi	4723	0.0 %	169.6 MB	

Детермінант виявився більшим, аніж може дозволити утримувати тип float64, тому було прийнято рішення використати бібліотеку decimal для обрахунку чисел з довільною точністю. Однак, отриманий детермінант виявився астрономічної величини $\sim 10^{45212}$, через що зникає будь-яка можливість перевірити його на точність, адже бібліотека numpy не може обрахувати такого розміру числа. Проте варто вказати те, що детермінанти отримані від двох методів майже збігаються.

Для аналізу на достовірність та точність було використано бібліотеку мови програмування python - numpy. Вона є достовірним “золотим стандартом”, що має підтверджену подвійну точність згідно стандарту IEEE 754.

Аналіз на достовірність:

Абсолютні та відносні похибки були обчислені через другу норму різниці між нашим розв’язком та розв’язком, отриманим за допомогою NumPy. Такі похибки свідчать про достовірність чисельного розв’язку. Варто зауважити, що вдвічі швидший частковий метод дав менші помилки, аніж складніший повний метод.

Аналіз на точність:

Відносні похибки знаходяться на рівні очікуваної точності для precision і повністю узгоджуються з оцінкою через умову матриці.

$$\delta a < cond(A)\varepsilon_{mach} \approx 9.45 \cdot 10^{4} \cdot 1.11 \times 10^{-16} \approx 1.049 \cdot 10^{-11}$$

Відносна похибка часткового методу: $7.442 \cdot 10^{-12} < 1.049 \cdot 10^{-11}$
Відносна похибка повного методу: $2.269 \cdot 10^{-11} \approx 1.049 \cdot 10^{-11}$

Це підтверджує, що метод Гаусса дав високоточний розв’язок.

Аналіз на стійкість:

Стійкість оцінювалася як друга норма різниці між нашим розв’язком та розв’язком для модифікованих вхідних даних з доданим малим шумом.

Метод повного вибору виявився дещо стійкішим, аніж метод часткового вибору.

Різниця між розв’язками у 9 цифр після коми демонструє, що метод зберігає числову стійкість щодо невеликих варіацій у вхідних даних, і його похибки відповідають передбачуваним оцінкам за умовою числової задачі.

Висновки

1. Досягнення мети та реалізація

Мета роботи (засвоєння способів практичного використання методу Гауса) успішно досягнута шляхом реалізації програми, що розв'язує СЛАР з використанням двох модифікацій: постовпцевого та повного вибору головного елемента.

Львів - 2025 р.

Реалізація: Успішно створено модульний програмний комплекс (create_inputs.py, main.py, gaussian.py, data_handler.py) для генерації, розв'язання та оцінки СЛАР.

Визначник: Програма коректно обчислює визначник, навіть для матриць великого розміру, використовуючи бібліотеку decimal для обробки чисел астрономічної величини (напр., ∼1045212).

2. Аналіз складності та продуктивності

Було проведено детальний аналіз складності та чисельні експерименти на великих матрицях (n=10000).

Обчислювальна складність: Теоретично підтверджено, що обидві модифікації методу Гауса мають кубічну складність

Просторова складність: Визначено як O(n^2), а прогнозована потреба в пам'яті (напр., ≈1.6 ГБ для n=10000) підтвердилася експериментально.

Часова складність: Експерименти з n=10000 показали, що реальний час виконання виявився меншим за прогнозований, підтверджуючи ефективність реалізації:

Постовпцевий вибір: ≈8 хвилин (прогноз: 17 хвилин).

Повний вибір: ≈20 хвилин (прогноз: 31 хвилина).

3. Аналіз достовірності, точності та стійкості

Оцінка якості розв'язку проводилась із застосуванням бібліотеки NumPy як "золотого стандарту".

Достовірність та Точність:

Метод Гауса дав високоточний розв'язок, оскільки відносні похибки знаходяться на рівні очікуваної точності. Парадоксально, але швидший постовпцевий метод показав меншу відносну похибку (7.442·10−12) порівняно з повним методом (2.269·10−11). Це свідчить про високу якість реалізації часткового вибору для тестових матриць.

Стійкість:

Обидва методи демонструють числову стійкість щодо невеликих варіацій у вхідних даних (шуму).

Метод повного вибору виявився дещо стійкішим, що відповідає теоретичним очікуванням, хоча й поступається частковому методу в точності для даного набору тестів.

Львів - 2025 р.