Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій



**ЗВІТ**

**Про виконання лабораторної роботи № 5**

«Наближені методи розв’язування систем лінійних алгебраїчних рівнянь»

**з дисципліни «Чисельні методи»**

**Лектор:**

доцент кафедри ПЗ

Мельник Н.Б.

**Виконав:**

студ. групи ПЗ-15

Марущак А. С.

**Прийняв:**

асистент кафедри ПЗ

Гарматій Г.Ю

«\_\_\_» \_\_\_\_\_\_ 2022 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2022

**Тема роботи:** Наближені методи розв’язування систем лінійних алгебраїчних рівнянь

**Мета роботи:** ознайомлення на практиці з методами Якобі та Зейделя розв’язування систем лінійних алгебраїчних рівнянь

**Теоретичні відомості**

Розглянемо систему лінійних алгебраїчних рівнянь

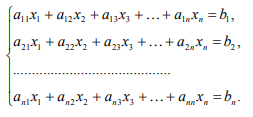
Наближені методи розв’язування СЛАР полягають у тому, що вектор знаходять як границю послідовних наближень , де n – номер ітерації. Навіть в припущенні, що обчислення виконують без заокруглень, ітераційні методи дають змогу визначити розв’язок тільки з заданою точністю

||||

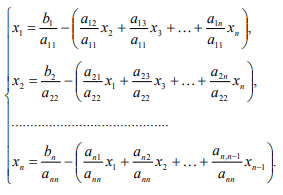
Для розв’язування СЛАР наближеними методами розглянемо метод простої ітерації (Якобі) та метод Зейделя.

**Метод Якобі**

Розглянемо систему лінійних алгебраїчних рівнянь виду:



Припустивши, що коефіцієнти , розв’яжемо i -те рівняння системи відносно . У результаті отримаємо таку систему:



Введемо позначення



Тоді систему можна переписати у вигляді:

Нехай перше наближення системи рівне стовпцю вільних членів, тобто

.

Тоді наближення розв’язку можна описати рекурентною формулою:

.

Якщо послідовність є збіжною, то тоді – розв'язок системи.

**Алгоритм методу Якобі:**

1. Повторюючи вищенаведені кроки записуємо систему в зведеному вигляді.
2. На основі отриманої СЛАР записуємо матрицю та **.**
3. Встановалюємо початкове наближення рівним вектору-стовпцю : .
4. Допоки не досягнемо потрібної точності, використовуючи рекурентну формулу , уточнюємо розв’язок.

**Метод Зейделя**

Даний метод є модифікацією методу простої ітерації. Основна його ідея полягає в тому, що при обчисленні чергового k-го наближення розв’язку використовують вже знайдені значення k-го наближеного розв’язку .

Ітераційна формула методу Зейделя має вигляд



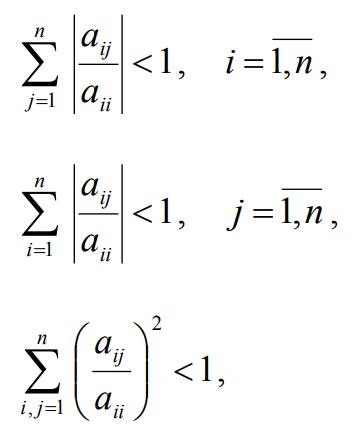
Алгоритм методу Зейделя аналогічний до алгоритму методу Якобі за винятком формули, що використовується у кроці (4).

**Збіжність ітераційного процесу**

Збіжність ітераційного процесу залежить від величини коефіцієнтів матриці α. Тому не обов’язково за нульове наближення розв’язку вибирати стовпець вільних членів. За початкове наближення можна також вибирати

* вектор, усі координати якого ;
* вектор, усі координати якого ;
* вектор , отриманий у результаті аналізу особливостей об’єкту дослідження та задачі, яку розв’язують.

Якщо елементи матриці задовольняють одну з умов

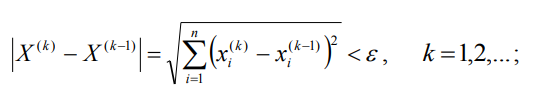


то система рівнянь має єдиний розв’язок , який не залежить від початкового наближення.

**Критерії припинення ітераційного процесу**

Якщо задана похибка ε наближеного розв’язку, то критерієм припинення ітераційного процесу вважають виконання однієї з умов:

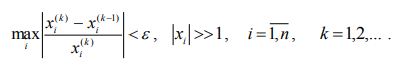
- модуль різниці між наступним та попереднім наближенням розв’язку повинен бути меншим за ε



- максимальне значення модуля різниць між відповідними компонентами наступного та попереднього наближення розв’язку повинно бути меншим за ε

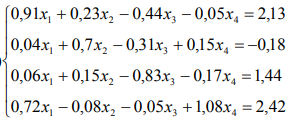


- максимальне значення модуля відносних різниць між відповідними компонентами наступного та попереднього наближення розв’язку повинно бути меншим за ε



**Індивідуальне завдання**

1. Ознайомитися з теоретичним матеріалом.
2. Скласти програму розв’язування системи лінійних алгебраїчних рівнянь методом Якобі та Зейделя з точністю . Порівняти кількість ітерацій для обох методів.



**Хід роботи**

Проведемо деякі обчилення вручну, щоб потім мати змогу порівняти

результат виконання програми з дійсним.

По-перше, поділимо кожен і-й рядок на коефіцієнт, що стоїть біля . Тоді отримаємо наступне:

Перенесемо з лівої частини і-го рядка всі члени, що не містять , у праву сторону з протилежним знаком. Тоді наша СЛАР набуде наступного вигляду:

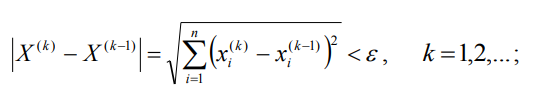
Нехай , тоді запишемо систему у компактному вигляді:

Обчислимо хоча б одну канонічну норму матриці , щоб переконатись у збіжності ітераційного процесу. Нехай це буде максимум суми модулів елементів кожного рядка:

,

Отже, ітераційний процес буде збіжним.

Нехай початкове наближення . Тоді проведемо обчислення до точності за рекурентною формулою методом Якобі та Зейделя, обчислюючи різницю між векторами-стовпцями та , як відстань між відповідними векторами у просторі, тобто



Результати обчислень подамо у вигляді таблиці:

**Метод Якобі**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ ітерації** |  |  |  |  |  |
| **0** | **2,341** | **-0,257** | **1,735** | **2,241** | **-** |
| **1** | **1,690** | **-1,639** | **-2,071** | **0,581** | **2,281** |
| **2** | **1,785** | **-1,395** | **-2,028** | **0,897** | **0,413** |
| **3** | **1,762** | **-1,449** | **-2,042** | **0,853** | **0,075** |
| **4** | **1,767** | **-1,445** | **-2,044** | **0,864** | **0,013** |
| **5** | **1,765** | **-1,449** | **-2,045** | **0,861** | **0,005** |
| **6** | **1,765** | **-1,448** | **-2,046** | **0,862** | **0,00096** |

**Метод Зейделя**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ ітерації** |  |  |  |  |  |
| **0** | **2,341** | **-0,257** | **1,735** | **2,241** | **-** |
| **1** | **1,690** | **-1,602** | **-2,361** | **0,886** | **2,112** |
| **2** | **1,653** | **-1,587** | **-2,084** | **0,925** | **0,283** |
| **3** | **1,785** | **-1,480** | **-2,063** | **0,846** | **0,189** |
| **4** | **1,764** | **-1,453** | **-2,043** | **0,863** | **0,043** |
| **5** | **1,767** | **-1,448** | **-2,046** | **0,861** | **0,007** |
| **6** | **1,765** | **-1,448** | **-2,045** | **0,862** | **0,003** |
| **7** | **1,765** | **-1,449** | **-2,046** | **0,862** | **0,0006** |

Отож, в обох методах отримали однаковий результат:

Наступна мета – реалізувати подані методи програмно і переконатися в достовірності результату.

Для подальшої роботи я буду використовувати структуру даних Matrix, яка відповідає матриці заданого розміру, як зрозуміло з назви. Серед її функціональних можливостей – розклад на матриці та , якщо матриця представляє собою матрицю системи, а також множення і додавання матриць, перевірку на збіжність за допомогою канонічних норм. Покажемо це у коді:

public void Deconstruct(out Matrix alpha, out Matrix beta)

{

if (Columns != Rows + 1)

throw new Exception("Not acceptable matrix to decompose!");

Matrix A = new(Rows, Rows);

Matrix B = new(Rows, 1);

for(int i = 0; i < Rows; i++)

{

A.data[i] = this.data[i].Take(rows).Select((a, j) => j == i ? 0 : -a / this.data[i][i]).ToArray();

B[i, 0] = this.data[i].Last() / this.data[i][i];

}

alpha = A;

beta = B;

}

public static Matrix operator\*(Matrix A, Matrix B)

{

if (A.Columns != B.Rows)

throw new ArgumentException();

Matrix result = new(A.Rows, B.Columns);

for(int i = 0; i < result.Rows; i++)

{

for(int j = 0; j < result.Columns; j++)

{

double sum = 0;

for(int k = 0; k < A.Columns; k++)

{

sum += A[i, k] \* B[k, j];

}

result[i, j] = sum;

}

}

return result;

}

public static Matrix operator+(Matrix A, Matrix B)

{

if (A.Columns != B.Columns || A.Rows != B.Rows)

throw new ArgumentException();

Matrix result = new(A.Rows, A.Columns);

for (int i = 0; i < result.Rows; i++)

{

for (int j = 0; j < result.Columns; j++)

{

result[i, j] = A[i, j] + B[i, j];

}

}

return result;

}

public bool TestNorms()

{

if (Rows != Columns) throw new Exception();

double firstNorm = this.Select(row => row.Sum(elem => Math.Abs(elem))).Max();

double secondNorm = this.Transposed().Select(row => row.Sum(elem => Math.Abs(elem))).Max();

double thirdNorm = Math.Sqrt(this.Sum(row => row.Sum(elem => elem \* elem)));

Console.WriteLine($"Перша норма: {firstNorm:0.000}");

Console.WriteLine($"Друга норма: {secondNorm:0.000}");

Console.WriteLine($"Третя норма: {thirdNorm:0.000}");

if (firstNorm < 1) return true;

if (secondNorm < 1) return true;

if (thirdNorm < 1) return true;

return true;

}

Тоді, подані методи можемо реалізувати наступним чином.

**Метод Якобі:**

static Matrix YakobiMethod(Matrix alpha, Matrix beta, out int iterations)

{

iterations = 0;

//Початкове наближення - матриця бета

Matrix X = beta.Clone();

Matrix X\_prev = new Matrix(X.Rows, X.Columns);

do

{

//Зберігаємо попереднє наближення

X\_prev = X.Clone();

//Використовуємо рекурентну формулу

X = alpha \* X + beta;

iterations++;

} while (X.DistanceTo(X\_prev) > 1e-3); // Допоки не досягнемо потрібної точності

//Повертаємо результат

return X;

}

**Метод Зейделя:**

static Matrix ZeidelMethod(Matrix alpha, Matrix beta, out int iterations)

{

iterations = 0;

//Початкове наближення - матриця бета

Matrix X = beta.Clone();

Matrix X\_prev = new Matrix(X.Rows, X.Columns);

do

{

//Зберігаємо попереднє наближення

X\_prev = X.Clone();

//Обчислюємо Xi

for (int i = 0; i < X.Rows; i++)

{

double sum = 0;

// використовуючи вже обчислені члени поточного наближення

for (int k = 0; k < i; k++)

{

sum += X[k, 0] \* alpha[i, k];

}

//і ще не обчислені на даній ітерації члени з попереднього наближення

for (int j = i+1; j < X.Rows; j++)

{

sum += X\_prev[j, 0] \* alpha[i, j];

}

X[i, 0] = sum + beta[i,0];

}

iterations++;

} while (X.DistanceTo(X\_prev) > 1e-3); // Допоки не досягнемо потрібної точності

//Повертаємо результат

return X;

}

Тоді, з умови дістанемо розширенну матрицю системи

Подамо цю матрицю на вхід програми і подивимось на результат.

**Результат** виконання програми:

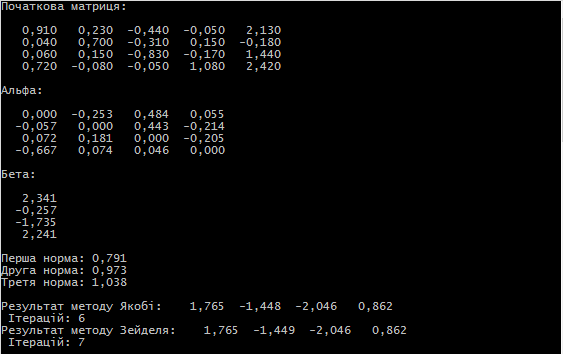


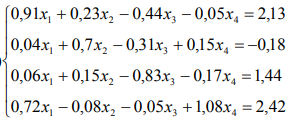
Рис 5.1 Результат виконання програми.

**Аналіз результатів:**

Як бачимо, результати обчислень на комп’ютері повністю співпали з результатами обчислень, здійснених вручну. Також, співпали матриці та . Тому, можемо судити про достовірність як і ручних обчислень, так і програми, складеної на основі вивчених алгоритмів. К-ть ітерацій обох методів при різній точності обчислень приблизно рівна.

**Висновок:**

Виконуючи цю лабораторну роботу ми ознайомились з 2-ма ітеративними методами розв’язування СЛАР: методом Якобі та Зейделя. За допомогою цих знань ми реалізували програму, що розв’язала наступну систему:



Перед цим, ми провели деякі обчислення вручну, і проміжні результати обчислень повністю аналогічні до результату виконання програми.

Ми отримали вектор-стовпець дійсних чисел, що відображає розв' язок цієї системи: