Київський національний університет імені Тараса Шевченка
Факультет комп’ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем
Алгоритми та складність

Лабораторна робота №1

“ Побудова оберненої матриці методом Гауса-Жордана ”

Варіант №1

Виконала студентка 2-го курсу
Групи ІПС-21
Сенечко Дана Володимирівна

Київ - 2024

Завдання

Побудова оберненої матриці методом Гауса-Жордана. Стиль матриць T**. Тип даних: дійснозначні матриці.

Теорія

Метод Гауса — Жордана використовується для розв'язання систем лінійних алгебраїчних рівнянь, знаходження оберненої матриці, знаходження координат вектора у заданому базисі, знаходження рангу матриці. Метод є модифікацією методу Гаусса.

Метод Гауса (Гаусова елімінація*) — алгоритм розв'язування систем лінійних алгебраїчних рівнянь. Зазвичай під цим алгоритмом розуміють деяку послідовність операцій, що виконують над відповідною матрицею коефіцієнтів, для приведення її до трикутного вигляду, з наступним вираженням базисних змінних через небазисні. Цей метод також можливо використовувати для знаходження рангу матриці, для обчислення визначника матриці, а також для обчислення обернення невиродженої квадратної матриці.

***Елімінація** - виключення невідомої змінної з системи рівнянь.

Обернена матриця — матриця (позначається A^{-1}), яка існує для кожної невиродженої квадратної матриці A, розмірності $n \times n$, причому: $AA^{-1} = A^{-1}A = I_n$, де I_n одинична $n \times n$ матриця.

Якщо для матриці A існує A^{-1}, то така матриця називається **оборотною**, тобто кожна невироджена матриця є оборотною, і навпаки — кожна оборотна матриця є невиродженою.

Невироджена матриця — квадратна матриця, визначник якої не дорівнює нулю: $det(A) \neq 0$.

Квадратна матриця — це матриця з однаковою кількістю рядків і стовпців. $n \times n$-матриця — це квадратна матриця порядку n:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix}.$$

Алгоритм

- Для обчислення оберненої матриці, формується розширена матриця шляхом об'єднання початкової матриці з одиничною матрицею.
- Обирається перша зліва колонка, що містить хоч одне ненульове значення.
- Якщо верхнє число у цій колонці — нуль, то обмінюється увесь перший рядок матриці з іншим рядком матриці, де у цій колонці немає нуля.
- Усі елементи першого рядка діляться на верхній елемент обраної колонки.
- Від рядків, що залишились, віднімається перший рядок, помножений на перший елемент відповідного рядка, з метою отримання нуля в першому елементі кожного рядка (крім першого).
- Далі, повторюємо ці операції із матрицею, отриманою з початкової матриці після викреслювання першого рядка та першого стовпчика.
- Після повторення операцій $n - 1$ разів отримаємо верхню трикутну матрицю.

- Віднімаємо від передостаннього рядка останній рядок, помножений на відповідний коефіцієнт, щоб у передостанньому рядку залишилась лише 1 на головній діагоналі.
- Повторюємо попередній крок для наступних рядків. У результаті отримуємо одиничну матрицю і рішення на місці вільного вектора (над ним необхідно виконувати ті самі перетворення).

Складність алгоритму

1. Розширення матриці [*A*|*I*]:

   Створення розширеної матриці з розмірами $n \times 2n$ займає $O(n^2)$ часу, оскільки потрібно скопіювати всі елементи з початкової матриці *A* і заповнити одиничну матрицю *I*.
2. Гауссове елімінаційне перетворення:
   Основний етап алгоритму, де виконується Гауссова елімінація, складається з двох вкладених циклів:
   1) Зовнішній цикл (по рядках) виконується *n* разів.
   2) Внутрішній цикл (по стовпцях) для кожного рядка займає до *n* часу, оскільки потрібно обробити всі стовпці.

   Це призводить до того, що загальна складність цього етапу становить $O(n^3)$.
3. Копіювання результату:
   Копіювання оберненої матриці з розширеної матриці також займає $O(n^2)$.

Отже, алгоритм має кубічну часову складність $O(n^3)$, що є типовим для методів, які використовують елімінацію для вирішення систем лінійних рівнянь.

Мова реалізації алгоритму

C++

Модулі програми

- **`typedef double T;`**

Визначаємо тип T як double, щоб використовувати його для дійсних чисел в матриці.

- **`void printMatrix(T** matrix, int n)`**

Метод, який виводить матрицю на екран. Тут використовується форматування для красивого виводу (бібліотека iomanip). Якщо елемент менший за epsilon, він замінюється на 0.

- **`bool inverseMatrix(T** A, T** invMatrix, int n)`**

Алгоритм Гауса-Жордана для знаходження оберненої матриці.

- **`int main()`**

Головна функція, у якій відбувається взаємодія з користувачем та виклик інших функцій програми.

Інтерфейс користувача

Введення даних відбувається через консоль. Користувачеві спочатку пропонується ввести розмір матриці, а після цього всі її елементи. Далі програма виконує обчислення і виводить обернену заданій матрицю.

Вхідні дані:

- Розмір матриці:

  Користувач вводить ціле число, що визначає розмір квадратної матриці (n × n).

- Елементи матриці:

  Користувач вводить n*n елементів, розділених пробілами або переходами на новий рядок, для заповнення матриці.

Доступні дії:

1. Введення розміру матриці.
2. Введення елементів матриці.

Виведення результатів та завершення роботи програми відбувається автоматично після введення користувачем потрібних даних. У разі, якщо матриця не має оберненої матриці (негативний або нульовий визначник), програма повідомляє про це користувача.

Тестові приклади

1. Матриця 2x2, яка має обернену:

```
Enter the matrix size: 2
Enter the elements of matrix A:
4 7
2 6
Inverse matrix:
  0.6  -0.7
 -0.2   0.4
```

2. Матриця 3x3, яка має обернену:

```
Enter the matrix size: 3
Enter the elements of matrix A:
1 2 3
0 1 4
5 6 0
Inverse matrix:
  -24    18     5
   20   -15    -4
   -5     4     1
```

3. Матриця 3x3, яка не має оберненої (сингулярна):

```
Enter the matrix size: 3
Enter the elements of matrix A:
2 4 8
2 4 8
2 4 8
The matrix is degenerate and has no inverse.
Failed to inverse matrix.
```

4. Одинична матриця 3x3:

```
Enter the matrix size: 3
Enter the elements of matrix A:
1 0 0
0 1 0
0 0 1
Inverse matrix:
    1     0     0
    0     1     0
    0     0     1
```

5. Матриця з нулем на діагоналі (перестановка рядків):

```
Enter the matrix size: 3
Enter the elements of matrix A:
0 2 3
1 1 1
2 3 4
Inverse matrix:
   -1    -1     1
    2     6    -3
   -1    -4     2
```

6. Велика матриця 5x5:

```
Enter the matrix size: 5
Enter the elements of matrix A:
4 7 2 8 6
3 5 1 7 4
2 6 8 9 7
5 1 3 2 8
9 4 6 1 3
Inverse matrix:
  -0.27    0.37  -0.069   0.039     0.1
   0.69    -0.7  -0.042   -0.14   0.011
 -0.097  -0.055    0.13  -0.033   0.051
   -0.5    0.65   0.048   0.013  -0.012
   0.24   -0.29  -0.012    0.13   -0.08
```

Висновки

Отже, у даній роботі було описано та реалізовано алгоритм знаходження оберненої матриці методом Гауса-Жордана. Цей метод є ефективним для обчислення оберненої матриці за допомогою послідовних елементарних операцій над рядками з метою приведення початкової матриці до одиничної. Під час виконання алгоритму ми використовуємо розширену матрицю [A | I], де на виході отримуємо обернену матрицю до заданої квадратної матриці.

Було розроблено програму, яка приймає на вхід розмір матриці та її елементи, після чого виконує обчислення оберненої матриці, або виводить повідомлення про те, що матриця є виродженою (визначник дорівнює нулю, не має оберненої). Програма також реалізує функцію для форматованого виведення обчисленої оберненої матриці.

Часова складність алгоритму знаходження оберненої матриці методом Гауса-Жордана склала $O(n^3)$, де n — розмірність матриці.

Використані літературні джерела

- Метод Гаусса — Йордана — Вікіпедія
- Метод Гаусса — Вікіпедія
- Обернена матриця — Вікіпедія
- Невироджена матриця — Вікіпедія
- Квадратна матриця — Вікіпедія

Використання реалізації алгоритму в груповому проекті:

Цей алгоритм використовує Тесленко Назар для перевірки побудови оберненої матриці методом LU-розкладання.

Я, в свою чергу, використовую алгоритм Штрассена для множення матриць, написаний Міцкевичем Костянтином, для перевірки роботи мого алгоритму. Якщо помножити обернену матрицю на початкову, маємо отримати одиничну матрицю: $A \times A^{-1} = I$.

Ось, для прикладу, перевірка для тесту №2:

```
std::cout << std::endl << "Matrix with integers: " << std::endl;
A = matrix_initialize(n);
B = matrix_initialize(n);
A[0][0] = 1;  A[0][1] = 2;   A[0][2] = 3;
A[1][0] = 0;    A[1][1] = 1; A[1][2] = 4;
A[2][0] = 5;  A[2][1] = 6;   A[2][2] = 0;

B[0][0] = -24;  B[0][1] = 18;  B[0][2] = 5;
B[1][0] = 20;  B[1][1] = -15;  B[1][2] = -4;
B[2][0] = -5;   B[2][1] = 4; B[2][2] = 1;
```

```
Matrix with integers:
1 0 0
0 1 0
0 0 1
```