Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Завдання №3

“Побудова оберненої матриці методом LU-розкладання”

Варіант №1

Виконав студент 2-го курсу

Групи ІПС-21

Тесленко Назар Олександрович

Київ – 2024

Завдання

Побудова оберненої матриці методом LU-розкладання.

Теорія

Алгоритм LU-розкладання дозволяє ефективно знайти обернену матрицю для квадратної матриці. Цей метод є особливо корисним для великих матриць, оскільки розкладання зменшує кількість необхідних обчислень.

LU-розкладання — це процес розбиття матриці A на добуток двох трикутних матриць L та U, де:

- **L** — нижня трикутна матриця з одиничними елементами на діагоналі.
- **U** — верхня трикутна матриця.

$$\boldsymbol{A = LU}$$

Обернена матриця — це матриця, яка, помножена на початкову матрицю, дає одиничну матрицю. Обернена матриця існує тільки для квадратних невироджених матриць.

$$\boldsymbol{A * A^{-1} = I}$$

Пряма підстановка — це метод для обчислення матриці Y у системі LY=I, який базується на послідовному обчисленні значень із використанням значень нижніх елементів у трикутній матриці.

Зворотна підстановка — це метод для знаходження A^{-1} із рівняння $U * A^{-1} = Y$, який використовується для розв'язання системи рівнянь з верхньою трикутною матрицею.

Умови застосування методу:

- Матриця A повинна бути квадратною та невиродженою (тобто мати ненульовий визначник).
- Матриця U не повинна мати нульових елементів на головній діагоналі, інакше неможливо розв'язати систему рівнянь за допомогою зворотної підстановки.

Алгоритм

Алгоритм знаходження оберненої матриці за допомогою LU-розкладу використовує два основні етапи: розклад матриці на дві трикутні матриці (LU-розклад) і розв’язування систем лінійних рівнянь для кожного стовпця оберненої матриці. Алгоритм працює з квадратною матрицею розміру n x n і використовує пряму та зворотню підстановки для знаходження відповідних елементів оберненої матриці.

LU-розклад:

- Початкова матриця A розкладається на дві трикутні матриці: L (нижня трикутна матриця з одиницями на головній діагоналі) і U (верхня трикутна матриця).
- Це здійснюється шляхом проходження по кожному рядку і та:
    - Обчислення елементів матриці U, використовуючи елементи з уже обчисленої матриці L.
    - Обчислення елементів матриці L, використовуючи елементи з матриці U.
- Після цього розклад матриці на L та U завершено.

Обчислення оберненої матриці:

- Для кожного стовпця і оберненої матриця, формується одиничний вектор e[i].
- Використовуючи цей вектор, спочатку виконується **пряма підстановка** для матриці L, щоб отримати проміжний вектор y, розв’язуючи систему рівнянь L * y = e[i].
- Після цього виконується **зворотна підстановка** для матриці U, щоб знайти відповідний стовпець оберненої матриці x, розв’язуючи систему рівнянь U * x = y.
- Цей процес повторюється для кожного стовпця оберненої матриці.

Результат:

- В результаті алгоритм повертає обернену матрицю inverse, яка була обчислена за допомогою LU-розкладу та підстановок.

Псевдокод виконання програми:

1 Виконання LU-розкладу для A

for кожного рядка i від 1 до n:

for кожного стовпця k від i до n:

U[i][k] = A[i][k] - Σ (L[i][j] * U[j][k]) для всіх j < i

for кожного стовпця k від i до n:

if i = k, то L[i][i] = 1

else L[k][i] = (A[k][i] - Σ (L[k][j] * U[j][i]) для всіх j < i) / U[i][i]

2 Обчислення оберненої матриці:

for кожного стовпця i від 1 до n: одиничний вектор e[i] розміром n

ForwardSubstitution(L, e[i], y, n) для знаходження вектора y

for кожного i від 0 до n-1:

y[i] = b[i] - Σ (L[i][j] * y[j]) для всіх j < i

BackwardSubstitution(U, y, x, n) для знаходження вектора x

for кожного i від n-1 до 0 (в зворотному напрямку):

x[i] = (y[i] - Σ (U[i][j] * x[j]) для всіх j > i) / U[i][i]

записати x як i-й стовпець Inverse

Складність алгоритму:

Допоміжні функції:

allocateMatrix(double**& matrix, int n) **– O(n^2)**

freeMatrix(double**& matrix, int n) **– O(n)**

printMatrix(double** matrix, int n) - **O(n^2)**

Головні функції:

luDecomposition(double** A, double** L, double** U, int n)

Часова складність:

Зовнішній цикл по рядках i: O(n)

Внутрішній цикл для заповнення елементів U[i][k]: O(n) для кожного рядка (сумарно O(n^2))

Внутрішній цикл для заповнення елементів L[k][i]: O(n) для кожного рядка (сумарно O(n^2))

- **Загальна складність: O(n^3)** — через вкладені цикли для кожного елемента матриць L і U.

forwardSubstitution(double L, double* I, double* y, int n)**

Часова складність:

Зовнішній цикл по кожному елементу i: O(n)

Вкладений цикл для обчислення суми: O(i), але в найгіршому випадку це O(n)

- **Загальна складність:** O(n^2)

backwardSubstitution(double U, double* y, double* x, int n)**

Часова складність:

Зовнішній цикл по кожному елементу i (від кінця до початку): O(n)

Вкладений цикл для обчислення суми: O(n - i), але в найгіршому випадку це також O(n)

- **Загальна складність:** O(n^2)

invertMatrix(double A, double** L, double** U, double** inverse, int n)**

Часова складність:

Виклик функції luDecomposition: O(n^3)

Цикл по кожному стовпцю (O(n)):

- Виклик функції forwardSubstitution: O(n^2) для кожного стовпця
- Виклик функції backwardSubstitution: O(n^2)для кожного стовпця

Обчислення оберненої матриці для кожного стовпця: O(n^3) (оскільки функції прямої і зворотної підстановки викликаються для кожного стовпця)

- **Загальна складність:** O(n^3)

Отже загальна складність алгоритму: O(n^3) + O(n^3) + O(n^3) = **O(n^3)**

Мова реалізації: C++

Модулі програми:

Допоміжні функції:

- allocateMatrix(double**& matrix, int n)

Виділяє динамічну пам'ять для матриці розміром n x n.

- freeMatrix(double**& matrix, int n)

Звільняє виділену пам'ять для матриці розміром n x n.

- printMatrix(double** matrix, int n)

Виводить матррицю розміром n x n до консолі

Головні функції:

- luDecomposition(double** A, double** L, double** U, int n)

Функція, що виконує розкладання матриці A на L та U

- forwardSubstitution(double** L, double* I, double* y, int n)

Функція, для обчислення матриці Y у системі LY=I, який базується на послідовному обчисленні значень із використанням значень нижніх елементів у трикутній матриці.

- backwardSubstitution(double** U, double* y, double* x, int n)

Функція, для знаходження A^{-1} із рівняння $U * A^{-1} = Y$, який використовується для розв'язання системи рівнянь з верхньою трикутною матрицею.

- invertMatrix(double** A, double** L, double** U, double** inverse, int n)

Функція, яка узагльнює усі вище описані функції, для зручного виклику

Інтерфейс користувача:

Введення даних користувачем відбувається через консоль. Спочатку вводить розмір матриці - n, після цього вводиться сама матриця A

За допомогою функції *printMatrix()* у консолі виводиться матриці *A, L, U* для переконання того, що розбиття матриці A на L та U було виконано вірно, і після цього виводиться обернена матриця *inversed*

Тестові приклади:

Однією умовою з завдань було реалізація алгоритму для дійсних чисел, отже проведемо тестування для чисел різних множин:

- Для додаткової перевірки коректності знаходження оберненої матриці використаємо функцію Сенечко Дани, що знаходить обернену матрицю методом Гаусса-Жордана

Натуральні числа N:

```
Enter the size n of matrix: 3
Enter matrix A:
2 3 1
4 7 2
6 18 5

Matrix A:
2 3 1
4 7 2
6 18 5

Matrix U:
2 3 1
0 1 0
0 0 2

Matrix L:
1 0 0
2 1 0
3 9 1
```

```
Inversed matrix using LU method:
-0.25 0.75 -0.25
-2 1 0
7.5 -4.5 0.5

Inversed by Gaus Jordan method:
-0.25 0.75 -0.25
-2 1 0
7.5 -4.5 0.5
```

Для перевірки корректної роботи LU розкладу обрахуємо на калькуляторі для порівняння добуток матриць, а також переконаємось що алгоритм правильно обраховує обернену матрицю A:

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 9 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 3 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 \\ 4 & 7 & 2 \\ 6 & 18 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 3 & 1 \\ 4 & 7 & 2 \\ 6 & 18 & 5 \end{bmatrix}^{-1} = \begin{bmatrix} -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} \\ -2 & 1 & 0 \\ \frac{15}{2} & -\frac{9}{2} & \frac{1}{2} \end{bmatrix}$$

Отже можемо засвідчитись в коректності роботи одразу декількох функцій, а саме: LU-розкладання, знаходження оберненої матриці методом LU-розкладання та знаходження оберненої матриці методом Гаусса-Жордана.

Цілі Z:

```
Enter the size n of matrix: 4
Enter matrix A:
-2  3 -1  5
 4 -7  2  1
 6  0 -3 -8
-1  9  4 -6

Matrix A:
-2 3 -1 5
4 -7 2 1
6 0 -3 -8
-1 9 4 -6

Matrix U:
-2 3 -1 5
0 -1 0 11
0 0 -6 106
0 0 0 153.5

Matrix L:
1 0 0 0
-2 1 0 0
-3 -9 1 0
0.5 -7.5 -0.75 1
```

```
Inversed matrix using LU method:
0.253529 0.193811 0.133008 0.0662324
0.167752 0.0211726 0.0537459 0.0716612
-0.0184582 0.140065 -0.0803474 0.115092
0.197068 0.0928339 0.00488599 0.00651466

Inversed by Gaus Jordan method:
0.253529 0.193811 0.133008 0.0662324
0.167752 0.0211726 0.0537459 0.0716612
-0.0184582 0.140065 -0.0803474 0.115092
0.197068 0.0928339 0.00488599 0.00651466
```

Раціональні Q:

```
Enter the size n of matrix: 5
Enter matrix A:
1.5  -2.3  3.0  4.2  5.1
6.5   7.0 -8.5 -9.3 10.4
2.7   1.8  0.9  3.3 -2.1
4.1  -5.0  6.6  7.2  8.3
9.4  -3.1  2.2  1.0 -4.5

Matrix A:
1.5 -2.3 3 4.2 5.1
6.5 7 -8.5 -9.3 10.4
2.7 1.8 0.9 3.3 -2.1
4.1 -5 6.6 7.2 8.3
9.4 -3.1 2.2 1 -4.5

Matrix U:
1.5 -2.3 3 4.2 5.1
0 16.9667 -21.5 -27.5 -11.7
0 0 3.02711 5.3677 -7.18385
0 0 0 -2.24854 -4.68046
0 0 0 0 -27.8513

Matrix L:
1 0 0 0 0
4.33333 1 0 0 0
1.8 0.350098 1 0 0
2.73333 0.075835 0.0100597 1 0
6.26667 0.666798 -0.747858 1.32032 1
```

```
Inversed matrix using LU method:
0.0216144 0.0292745 0.0357818 -0.00133367 0.0729948
-0.967735 0.0418236 0.279162 0.515521 -0.179534
-1.89667 0.00872461 0.156687 1.07609 -0.217736
1.03699 -0.0301079 0.0613603 -0.543412 0.0747385
0.0149922 0.0299141 -0.0273288 0.0474061 -0.035905

Inversed by Gaus Jordan method:
0.0216144 0.0292745 0.0357818 -0.00133367 0.0729948
-0.967735 0.0418236 0.279162 0.515521 -0.179534
-1.89667 0.00872461 0.156687 1.07609 -0.217736
1.03699 -0.0301079 0.0613603 -0.543412 0.0747385
0.0149922 0.0299141 -0.0273288 0.0474061 -0.035905
```

Ірраціональні І:

```
Enter the size n of matrix: 6
Enter matrix A:
1.414  1.732  2.236  3.162  2.645  1.618
2.718  3.141  1.618  2.236  1.732  2.828
1.732  2.236  3.141  1.414  2.645  3.605
2.236  3.141  1.414  1.618  3.605  2.645
1.414  2.236  3.162  2.645  3.141  1.732
2.828  1.618  2.645  1.732  2.236  3.141

Matrix A:
1.414 1.732 2.236 3.162 2.645 1.618
2.718 3.141 1.618 2.236 1.732 2.828
1.732 2.236 3.141 1.414 2.645 3.605
2.236 3.141 1.414 1.618 3.605 2.645
1.414 2.236 3.162 2.645 3.141 1.732
2.828 1.618 2.645 1.732 2.236 3.141
```

```
Matrix U:
1.414 1.732 2.236 3.162 2.645 1.618
0 -0.188262 -2.68005 -3.84202 -3.35224 -0.28213
0 0 -1.22763 -4.79548 -2.63337 1.45156
0 0 0 19.0622 9.09345 -9.79409
0 0 0 0 -1.56527 -1.0386
0 0 0 0 0 -5.23954

Matrix L:
1 0 0 0 0 0
1.92221 1 0 0 0 0
1.22489 -0.60811 1 0 0 0
1.58133 -2.13605 6.39166 1 0 0
1 -2.67712 5.09017 0.713834 1 0
2 9.8055 -19.9183 -3.27543 -4.56731 1
```

```
Inversed matrix using LU method:
-0.434027 0.062894 -0.67798 0.0610526 0.321787 0.716234
-0.480614 0.491459 0.0264971 -0.0127556 0.458964 -0.46766
-0.546147 0.0580918 -0.0168131 -0.331515 0.735866 0.121723
0.745515 0.102902 0.0469581 -0.16749 -0.419029 -0.158473
0.132715 -0.484341 -0.0969088 0.457961 -0.0604715 0.126638
0.59269 -0.0706577 0.654022 -0.00288735 -0.8717 -0.190856

Inversed by Gaus Jordan method:
-0.434027 0.062894 -0.67798 0.0610526 0.321787 0.716234
-0.480614 0.491459 0.0264971 -0.0127556 0.458964 -0.46766
-0.546147 0.0580918 -0.0168131 -0.331515 0.735866 0.121723
0.745515 0.102902 0.0469581 -0.16749 -0.419029 -0.158473
0.132715 -0.484341 -0.0969088 0.457961 -0.0604715 0.126638
0.59269 -0.0706577 0.654022 -0.00288735 -0.8717 -0.190856
```

Отже, виконавши даний алгоритм, використовуючи різний розмір матриць та числа різних множин, можна запевнитись, що алгоритм працює коректно для знаходження оберненої матриці методом LU-розкладання, що було додатково доведено знаходженням оберненої матриці методом Гаусса-Жордана.

Висновки:

У даній роботі було розглянуто алгоритм для знаходження оберненої матриці за допомогою LU-розкладу. Алгоритм включає в себе розділення матриці на дві трикутні матриці, L (нижню трикутну) та U (верхню трикутну), а також використання прямої та зворотної підстановок для обчислення оберненої матриці. У процесі роботи було реалізовано програму, що виконує LU-розклад вхідної матриці, а також обчислює та повертає обернену матрицю.

Також у програмі реалізовані функції для виділення та звільнення пам'яті під матриці, що оптимізує використання пам'яті і робить алгоритм більш ефективним. Під час обчислення оберненої матриці використовується LU-розклад, що знижує обчислювальну складність операції до O(n^3), де n — розмір матриці.

Використання учасниками групи:

Даний алгоритм був використаний у роботі Дани Сенечко для перевірки коректності роботи алгоритму знаходження оберненої матриці методом Гаусса-Жордана.

Використані джерела:

- LU-розклад матриці
- Метод LU Розкладу в Деталях: Повний Огляд Розв’язання Систем Лінійних Рівнянь
- Відео: LU-розклад матриці