МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа №3

Выполнил студент группы №M32111

Чу Тхи Фыонг Тхао

Преподаватель

Москаленко М. А.

САНКТ-ПЕТЕРБУРГ

2021

Цель работы: реализовать LU разложение, итерационный метод, нахождение обратной матрицы. Проанализировать алгоритмы на разных матрицах

1. LU разложение

Это представление матрицы А в виде произведения матриц LU, где L – нижняя треугольная матрица, U – верхняя треугольная матрица. Важно, чтобы матрица А была обратима, а все ведущие главные миноры матрицы невырождены.

Нахождение разложения

1. Зануляем матрицу U

2. Заполняем матрицу L как единичную

3. Проходимся циклом по i от 1 до n и по j от 1 до n проверяя условие:

Если i <= j то выполяем $u_{ij} = a_{ij} - \sum_{k=1}^{i-1}(l_{ik} * u_{kj})$

иначе выполняем $l_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik} * u_{kj}}{u_{jj}}$

Решение СЛАУ

Пусть $Ax = b$ и $A = L$U тогда решение СЛАУ получаем из следующей системы уравнений:

$$\begin{cases} Ly = b \\ Ux = y \end{cases}$$

Обратная матрица

Обратная матрица через LU разложение находится при решении СЛАУ вида

$Ly = E$, где E - единичная матрица. $UA^{-1} = y$

Хранение разреженных матриц

Для хранения матриц где преобладают нулевые элементы бывает эффективнее использовать особые форматы хранения. В нашем случае матрица будет в разреженно-строчном формате (CSR). В таком случае схема хранения состоит из таких частей:

- Массив значений — содержит подряд все ненулевые значения матрицы в порядке их расположения справа налево по строкам и сверху вниз по столбцам
- Массив индексов столбцов — массив размера равного количеству элементов, хранит индексы столбцов значений в том же порядке что и сами значения
- Массив индексации строк — массив размера на 1 больший числа строк исходной матрицы, в каждой i-той ячейке хранит количество ненулевых элементов в строках до i-1 включительно. Стоит отметить что первый элемент в таком случае всегда 0, а последний равен числу всех ненулевых элементов исходной матрицы.

2. Примеры работы

Матрица A

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

L – нижняя треугольная матрица

```
The L matrix is:
[[ 1.          0.          0.        ]
 [-0.5         1.          0.        ]
 [ 0.         -0.66666667  1.        ]]
```

U – верхняя треугольная матрица

```
The U matrix is:
[[ 2.         -1.          0.        ]
 [ 0.          1.5        -1.        ]
 [ 0.          0.          1.33333333]]
```

Нахождения обратной матрицы A с использованием LU-разложения

```
Matrix inverse is:
[[0.75 0.5  0.25]
 [0.5  1.   0.5 ]
 [0.25 0.5  0.75]]
```

Решения системы с использованием LU-разложения

$$A = \begin{bmatrix} 1 & 5 & 5 \\ 6 & 9 & 22 \\ 32 & 5 & 5 \end{bmatrix}, \ b = \begin{bmatrix} 1 \\ 2 \\ 7 \end{bmatrix}$$

```
Result:
[ 0.19354839  0.20843672 -0.0471464 ]
```

3. Решение СЛАУ

$$\begin{cases} 10x_1 - x_2 + 2x_3 = 6 \\ -x_1 + 11x_2 - x_3 + 3x_4 = 25 \\ 2x_1 - x_2 + 10x_3 - x_4 = -11 \\ 3x_2 - x_3 + 8x_4 = 15 \end{cases}$$

Метод Якоби

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

Порядок решения СЛАУ методом Якоби:

1) Приведение системы уравнений к виду, в котором на каждой строчке выражено какое-либо неизвестное значение системы.

2) Произвольный выбор нулевого решения, в качестве него можно взять вектор-столбец сводных чисел.

3) Производим подстановку произвольного нулевого решения в систему уравнений, полученную под пунктом №1

4) Осуществление дополнительных итераций, для каждой из которых используется решение, полученное на предыдущем этапе

```
System:
10.0*x1 + -1.0*x2 + 2.0*x3 + 0.0*x4 = 6.0
-1.0*x1 + 11.0*x2 + -1.0*x3 + 3.0*x4 = 25.0
2.0*x1 + -1.0*x2 + 10.0*x3 + -1.0*x4 = -11.0
0.0*x1 + 3.0*x2 + -1.0*x3 + 8.0*x4 = 15.0

Iteration 1: [ 0.6         2.27272727 -1.1         1.875     ]
Iteration 2: [ 1.04727273  1.71590909 -0.80522727  0.88522727]
Iteration 3: [ 0.93263636  2.05330579 -1.04934091  1.13088068]
Iteration 4: [ 1.01519876  1.95369576 -0.96810863  0.97384272]
Iteration 5: [ 0.9889913   2.01141473 -1.0102859   1.02135051]
Iteration 6: [ 1.00319865  1.99224126 -0.99452174  0.99443374]
Iteration 7: [ 0.99812847  2.00230688 -1.00197223  1.00359431]
Iteration 8: [ 1.00062513  1.9986703  -0.99903558  0.99888839]
Iteration 9: [ 0.99967415  2.00044767 -1.00036916  1.00061919]
Iteration 10: [ 1.0001186   1.99976795 -0.99982814  0.99978598]
Iteration 11: [ 0.99994242  2.00008477 -1.00006833  1.0001085 ]
Iteration 12: [ 1.00002214  1.99995896 -0.99996916  0.99995967]
Iteration 13: [ 0.99998973  2.00001582 -1.00001257  1.00001924]
Iteration 14: [ 1.00000409  1.99999268 -0.99999444  0.9999925 ]
Iteration 15: [ 0.99999816  2.00000292 -1.0000023   1.00000344]
Iteration 16: [ 1.00000075  1.99999868 -0.99999899  0.99999862]
Iteration 17: [ 0.99999967  2.00000054 -1.00000042  1.00000062]
Iteration 18: [ 1.00000014  1.99999976 -0.99999982  0.99999975]
Iteration 19: [ 0.99999994  2.0000001  -1.00000008  1.00000011]
Iteration 20: [ 1.00000003  1.99999996 -0.99999997  0.99999995]
Iteration 21: [ 0.99999999  2.00000002 -1.00000001  1.00000002]
Iteration 22: [ 1.          1.99999999 -0.99999999  0.99999999]
Iteration 23: [ 1.  2. -1.  1.]
Iteration 24: [ 1.  2. -1.  1.]
Iteration 25: [ 1.  2. -1.  1.]
Iteration 26: [ 1.  2. -1.  1.]
Iteration 27: [ 1.  2. -1.  1.]
Iteration 28: [ 1.  2. -1.  1.]
Iterations: 28
Solution:
[ 1.  2. -1.  1.]
```

Метод Зейлея

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \dots, n. \text{[4]}$$

```
System of equations:
[ 10*x1 +  -1*x2 +   2*x3 +   0*x4] = [  6]
[ -1*x1 +  11*x2 +  -1*x3 +   3*x4] = [ 25]
[  2*x1 +  -1*x2 +  10*x3 +  -1*x4] = [-11]
[  0*x1 +   3*x2 +  -1*x3 +   8*x4] = [ 15]
Iteration 1: [0. 0. 0. 0.]
Iteration 2: [ 0.6         2.32727273 -0.98727273  0.87886364]
Iteration 3: [ 1.03018182  2.03693802 -1.0144562   0.98434122]
Iteration 4: [ 1.00658504  2.00355502 -1.00252738  0.99835095]
Iteration 5: [ 1.00086098  2.00029825 -1.00030728  0.99984975]
Iteration 6: [ 1.00009128  2.00002134 -1.00003115  0.9999881 ]
Iteration 7: [ 1.00000836  2.00000117 -1.00000275  0.99999922]
Iteration 8: [ 1.00000067  2.00000002 -1.00000021  0.99999996]
Iteration 9: [ 1.00000004  1.99999999 -1.00000001  1.        ]
Iteration 10: [ 1.  2. -1.  1.]
Iterations: 10
Solution: [ 1.  2. -1.  1.]
```

4. Исследование методов на матрицах, число обусловленности которых регулируется за счет изменения диагонального преобладания

```
*)
A = [[-1.99 -1.99]
     [-2.99 -2.99]]

B = [-5.97 -8.97]

Solution with method Jacobi:
[-0.  0.]

*)
A = [[-3.998 -2.999 -0.999]
     [-2.999 -4.998 -1.999]
     [-0.999 -0.999 -1.998]]

B = [-12.993 -18.992  -8.991]

Solution with method Jacobi:
[-0.8875831  0.1124169  1.1124169]
```

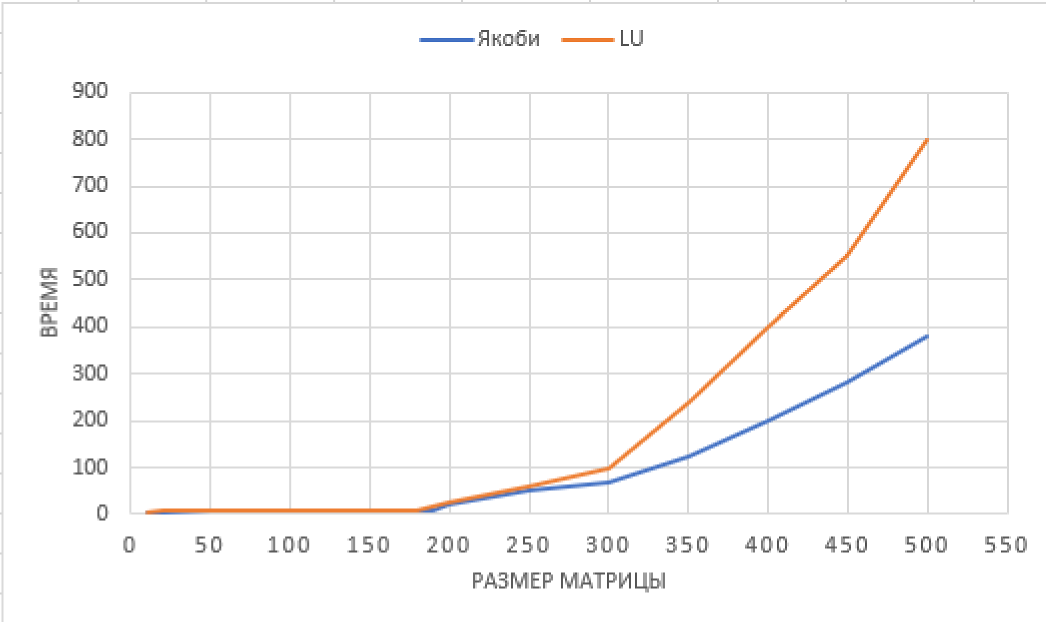
```
*)
A = [[ -4.9997  -0.9999  -0.9999  -2.9999]
     [ -0.9999  -5.9997  -3.9999  -0.9999]
     [ -2.9999  -0.9999  -7.9997  -3.9999]
     [ -3.9999  -2.9999  -3.9999 -10.9997]]

B = [-21.9988 -28.9986 -44.9984 -65.9982]

Solution with method Jacobi:
[-1.65454701 -0.65454701  0.34545299  1.34545299]

*)
A = [[ -9.99996  -3.99999  -1.99999  -2.99999  -0.99999]
     [ -1.99999 -10.99996  -3.99999  -2.99999  -1.99999]
     [ -3.99999  -2.99999 -12.99996  -1.99999  -3.99999]
     [ -1.99999  -0.99999  -2.99999  -6.99996  -0.99999]
     [ -1.99999  -1.99999  -1.99999  -0.99999  -6.99996]]

B = [-40.99982 -57.99979 -76.99976 -45.99973 -50.9997 ]

Solution with method Jacobi:
[-1.86901855 -0.86901855  0.13098145  1.13098145  2.13098145]


*)
A = [[ -8.999995  -0.999999  -2.999999  -1.999999  -0.999999  -1.999999]
     [ -0.999999 -13.999995  -2.999999  -3.999999  -1.999999  -3.999999]
     [ -3.999999  -2.999999 -12.999995  -1.999999  -0.999999  -2.999999]
     [ -3.999999  -3.999999  -3.999999 -14.999995  -0.999999  -1.999999]
     [ -2.999999  -1.999999  -2.999999  -0.999999 -10.999995  -1.999999]
     [ -3.999999  -0.999999  -2.999999  -2.999999  -0.999999 -11.999995]]

B = [ -44.999975  -87.999971  -79.999967 -100.99963   -86.999959 -103.999955]

Solution with method Jacobi:
[-2.31524496 -1.31524321 -0.31524854  0.68472564  1.68475034  2.6847556 ]
```

5. Аналогично на матрицах Гильберта

```
*)
A = [[1.         0.5       ]
     [0.5        0.33333333]]

B = [2.        1.6666667]

Solution with method Jacobi:
[-2.00000036  8.00000072]
```

```
*)
A = [[1.         0.5        0.33333333]
     [0.5        0.33333333 0.25      ]
     [0.33333333 0.25       0.2       ]]

B = [3.         1.91666667 1.4333333 ]

Solution with method Jacobi:
[-inf -inf   0.]

*)
A = [[1.         0.5        0.33333333 0.25      ]
     [0.5        0.33333333 0.25       0.2       ]
     [0.33333333 0.25       0.2        0.16666667]
     [0.25       0.2        0.16666667 0.14285714]]

B = [4.         2.71666667 2.1        1.72142857]

Solution with method Jacobi:
[-inf -inf   0.   0.]

*)
A = [[1.         0.5        0.33333333 0.25       0.2       ]
     [0.5        0.33333333 0.25       0.2        0.16666667]
     [0.33333333 0.25       0.2        0.16666667 0.14285714]
     [0.25       0.2        0.16666667 0.14285714 0.125     ]
     [0.2        0.16666667 0.14285714 0.125      0.11111111]]

B = [5.         3.55       2.81428571 2.34642857 2.01746032]

Solution with method Jacobi:
[-inf -inf   0.   0.   0.]

*)
A = [[1.         0.5        0.33333333 0.25       0.2        0.16666667]
     [0.5        0.33333333 0.25       0.2        0.16666667 0.14285714]
     [0.33333333 0.25       0.2        0.16666667 0.14285714 0.125     ]
     [0.25       0.2        0.16666667 0.14285714 0.125      0.11111111]
     [0.2        0.16666667 0.14285714 0.125      0.11111111 0.1       ]
     [0.16666667 0.14285714 0.125      0.11111111 0.1        0.09090989]]

B = [6.         4.40714286 3.56428571 3.01309524 2.61746032 2.31727994]

Solution with method Jacobi:
[-inf -inf   0.   0.   0.   0.]
```

6. Анализ эффективности методов на матрицах разных размеров



Вывод

В ходе выполнения лабораторной работы, мы реализовали LU разложение, с помощью LU реализовали алгоритм решения СЛАУ, так же был сделан итерационный алгоритм Якоби и проанализирован на разных матрицах, в ходе чего установили, что данный алгоритм не на всех матрицах может корректно работать. Данный алгоритм предназначен только для матриц со строгим диагональным преобладанием.