Реализация решения СЛАУ прямым методом (LU-, LR- разложения) (1.1.2г(а) – решение матричного уравнения)

Группа: ПМ-2001

Студент: Иксанов Марат Васильевич

1. Суть метода и алгоритм решения:

Для решения матричного уравнения $Ax = f$ матрица A представляется в виде произведения двух матриц: $LUx = f$, где L – нижняя треугольная матрица, а U – верхняя треугольная матрица, где

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}, a_{11} \neq 0, \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \neq 0, \dots, |\mathrm{A}| \neq 0,$$

$$L = \begin{pmatrix} l_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ l_{n1} & \cdots & l_{nn} \end{pmatrix}, R = \begin{pmatrix} 1 & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix},$$

$$f = \begin{pmatrix} f_{11} & \cdots & f_{1m} \\ \vdots & \ddots & \vdots \\ f_{n1} & \cdots & f_{nm} \end{pmatrix}, x = \begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix}$$

Данный метод является разновидностью метода Гаусса, поскольку можно разбить решение на 2 части, каждая из которых решается прямым и обратным ходом метода Гаусса.

Как было описано выше, мы представляем матричное уравнение $Ax = f$ как $LRx = f$. Далее можно представить это уравнение как $Lg = f$, где $Rx = g$, где $g = \begin{pmatrix} g_{11} & \cdots & g_{1m} \\ \vdots & \ddots & \vdots \\ g_{n1} & \cdots & g_{nm} \end{pmatrix}$

Поскольку L, R – нижняя и верхняя треугольные матрицы, на первом шаге решается уравнение $Lg = f$ прямым ходом. На втором шаге решается уравнение $Rx = g$ обратным ходом. В конечном итоге, матрица x – является искомым решением матричного уравнения $Ax = f$

2. Построение L, R, g матриц и прямой ход решения матричного уравнения:

Первый столбец L матрицы равен первому столбцу матрицы A, в то время, как первая строка R матрицы равна первой строке матрицы A, а первая строка матрицы g равна произведению l_{11}^{-1} на каждый элемент 1ой строки матрицы f. В дальнейшем, обратный элементом a^{-1} будем считать $\frac{1}{a}$.

Остальные столбцы и строки строятся следующим образом:

Последовательно строятся столбец k матрицы L и строка матрицы k матрицы $R.$ Столбец L равен разности соответствующего столбца матрицы A и суммы всех $k-1$ столбцов матрицы L до, каждый из которых соответственно умножен на r_{ik}, где i – индекс соответствующего столбца матрицы L, который меньше искомого индекса k. Строка R равна произведению l_{11}^{-1} на разность соответствующей строки k и суммы всех i строк матрицы R, умноженных соответственно на l_{ki} , где $i < k$. Описанный в этом абзаце процесс повторяется (n-k) раз.

Правая часть матричного уравнения, матрица g строится следующим образом: k строка матрицы g равна произведению l_{11}^{-1} на разность k строки матрицы f и суммы $k-1$ предыдущих строк, соответственно умноженных на l_{ki}, где i индекс одной из этих строк.

В данном пункте k идет от 2 до n последовательно .

3. Обратный ход решения матричного уравнения:

В данном пункте k идет от $n-1$ до 1 последовательно.

В начале автоматически присваиваем n строке матрицы x строку матрицы g. Далее каждая k строка матрицы x равна разности k строки матрицы g и суммы всех строк, индекс которых между k и n, умноженных на r_{kj} соответственно, где i – индекс текущей строки матрицы x в сумме.

Полученная матрица x является искомым решением матричного уравнения.

4. Программная реализация:

Входные данные: матрица A и матрица правой части f.

Выходные данные: матрица x.

Метод LU-, LR- разложения реализован с помощью Wolfram Mathematica:

```
input = {{1.00, 0.42, 0.54, 0.66}, {0.42, 1.00, 0.32, 0.44}, {0.54, 0.32, 1.00, 0.22}, {0.66, 0.44, 0.22, 1.00}};
f = Normal[SparseArray[{i_, i_} → 1, Dimensions[input]]];
n = Length[input];


g = ConstantArray[0, Dimensions[f]];
left = ConstantArray[0, {n, n}];
right = left;
x = ConstantArray[0, Dimensions[f]];

left⟦All, 1⟧ = input⟦All, 1⟧;
right⟦1, All⟧ = left⟦1, 1⟧^-1 * input⟦1, All⟧;
g⟦1, All⟧ = left⟦1, 1⟧^-1 * f⟦1, All⟧;

Do[
  Do[

    left⟦All, k⟧ = input⟦All, k⟧ - Sum[left⟦All, j⟧ * right⟦j, k⟧, {j, k - 1}];
    right⟦k, All⟧ = left⟦k, k⟧^-1 * (input⟦k, All⟧ - Sum[left⟦k, j⟧ * right⟦j, All⟧, {j, k - 1}]);

    g⟦k, All⟧ = left⟦k, k⟧^-1 * (f⟦k, All⟧ - Sum[left⟦k, j⟧ * g⟦j, All⟧, {j, k - 1}])
    , {i, k, n}];
  , {k, 2, n}];

x⟦n, All⟧ = g⟦n, All⟧;
Do[
  x⟦k, All⟧ = g⟦k, All⟧ - Sum[right⟦k, j⟧ * x⟦j, All⟧, {j, k + 1, n}]

  , {k, n - 1, 1, -1}];
x // MatrixForm
```

5. Результат вычисления на предоставленных входных данных и оценка точности решения:

Для оценки точности решения рассмотрим входные данные, тип данных входных матриц – числа с плавающей точкой.

Входные данные: $A = \begin{pmatrix} 1.00 & 0.42 & 0.54 & 0.66 \\ 0.42 & 1.00 & 0.32 & 0.44 \\ 0.54 & 0.32 & 1.00 & 0.22 \\ 0.66 & 0.44 & 0.22 & 1.00 \end{pmatrix}$, $f = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Необходимо решить матричное уравнение $Ax = f$, иначе говоря найти обратную матрицу матрицы A

Решение с помощью своей функции:

$$\begin{pmatrix} 2.50759 & -0.123039 & -1.01149 & -1.37834 \\ -0.123039 & 1.33221 & -0.261427 & -0.447454 \\ -1.01149 & -0.261427 & 1.53183 & 0.445609 \\ -1.37834 & -0.447454 & 0.445609 & 2.00855 \end{pmatrix}$$

Решение (нахождение обратной матрицы) с помощью встроенной функции $Inverse[A]$:

$$\begin{pmatrix} 2.50759 & -0.123039 & -1.01149 & -1.37834 \\ -0.123039 & 1.33221 & -0.261427 & -0.447454 \\ -1.01149 & -0.261427 & 1.53183 & 0.445609 \\ -1.37834 & -0.447454 & 0.445609 & 2.00855 \end{pmatrix}$$

Поскольку результатами вычислений являются матрицы, составленные из чисел с плавающей точкой, полученные значения могут отличаться на довольно малую величину, следовательно, для определения точности решения построим матрицу невязки и определим ее норму. Матрица невязки – разность левой и правой части матричного уравнения с найденной матрицей x, то есть $Ax - f$

Вычисленная норма матрицы невязки собственной функции:

```
Norm[input.x - f]
```

3.7904×10^{-16}

Вычисленная норма матрицы невязки встроенной функции:

```
Norm[input.Inverse[input] - f]
```

3.4701×10^{-16}

Можно заметить, что нормы практически совпадают и достаточно малы, отсюда следует, что обе функции вычисляют довольно точно.