МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Благовещенский государственный педагогический университет»

(ФГБОУ ВПО «БГПУ»)

КУРСОВАЯ РАБОТА

на тему: «Решение систем линейных алгебраических уравнений методом LU разложений по алгоритму Краута»

по дисциплине: Математика

Исполнитель:

Студент группы 2 «А»\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.А. Пацура

дата подпись

Руководитель: \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_П.П. Алутин

к.ф-м.н, доцент, дата подпись

заведующий

кафедры математики

и МОМ

Благовещенск 2015

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

1 LU-разложение 4

1.1 Понятие LU-разложение 4

2 Применения 6

2.1 Решение систем алгебраических линейных уравнений 6

2.2 Обращение матриц 6

2.3 Вычисление определителя матрицы 6

2 Алгоритм Краута 8

ЗАКЛЮЧЕНИЕ 11

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 12

ПРИЛОЖЕНИЯ 13

# ВВЕДЕНИЕ

Одним из лучших методов решения систем линейных алгебраических уравнений общего вида является метод, основанный на разложении исходной матрицы на произведение треугольных, или метод LU-факторизации. Алгоритмы этого метода близки к алгоритмам метода Гаусса, хотя вычисления могут производиться в различной последовательности. Главным преимуществом метода LU-факторизации в сравнении с методом Гаусса является возможность более быстрого получения решений для различных векторов свободных членов, а также для транспонированной системы уравнений.

По числу операций, необходимых для решения конкретной системы, методы Гаусса и LU-факторизации практически неразличимы. Однако из всех операций решения в методе LU-разложения почти половина приходится на этап факторизации и следующая половина - на собственно решение двух треугольных систем. В силу этого факта при необходимости решения системы, отличающейся лишь вектором свободных членов, получаем экономию на операциях разложения. Аналогично, когда необходимо решать транспонированную систему после исходной, можно воспользоваться предыдущей факторизацией и лишь сменить порядок решения треугольных систем в силу операции транспонирования.

Объект курсового исследования – решение систем линейных алгебраических уравнений методом LU - разложения.

Предмет курсового исследования – LU разложений по алгоритму Краута.

Целью данной курсовой работы является изучение/решение систем линейных алгебраических уравнений методом Краута. А так же программная реализация данного метода.

# 1 LU-разложение

## 1.1 Понятие LU-разложение

**LU-разложение** (**LU-декомпозиция**, **LU-факторизация**) — представление матрицы A в виде произведения двух матриц, **A = L \* U**, где **L** — нижняя треугольная матрица, а **U** — верхняя треугольная матрица.

**Матрица** - математический объект, записываемый в виде прямоугольной таблицы элементов кольца или поля (например, целых, действительных или комплексных чисел), которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы.

В методе решения систем линейных уравнений, основанном на факторизации, предполагается, что исходная система может быть представлена в виде

т.е.

Причем для определенности полагается, что матрица L является нижнетреугольной, а **U**–верхнетреугольной, причем с единичной диагональю.

Забегая несколько вперед, сразу отметим, что определитель треугольной матрицы равен произведению диагональных элементов, а определитель произведения матриц равен произведению их определителей. Откуда следует, что определитель матрицы A равен произведению диагональных элементов нижнетреугольной матрицы L:

Ограничившись для наглядности порядком **n = 4**, запишем **L**- и **U**-матрицы:

,

,

# 2 Применения

## 2.1 Решение систем алгебраических линейных уравнений

Полученное LU-разложение матрицы **A** (матрица коэффициентов системы) может быть использовано для решения семейства систем линейных уравнений с различными векторами **b** в правой части:

Если известно LU-разложение матрицы **A**, , исходная система может быть записана как

,

Эта система может быть решена в два шага. На первом шаге решается система

,

Поскольку I - нижняя треугольная матрица, эта система решается непосредственно прямой подстановкой.

На втором шаге решается система

,

Поскольку U - верхняя треугольная матрица, эта система решается непосредственно обратной подстановкой.

## 2.2 Обращение матриц

Обращение матрицы **A** эквивалентно решению линейной системы

,

где **X** - неизвестная матрица, **I** - единичная матрица. Решение **X** этой системы является обратной матрицей .

Систему можно решить описанным выше методом **LU**-разложения.

## 2.3 Вычисление определителя матрицы

**Определитель** (или **детерминант**) – это многочлен, комбинирующий элементы квадратной матрицы таким образом, что его значение сохраняется при транспонировании и линейных комбинациях строк или столбцов.  
То есть, определитель характеризует содержание матрицы. В частности, если в матрице есть линейно-зависимые строки или столбцы, — определитель равен нулю.  
Определитель играет ключевую роль в решении в общем виде систем линейных уравнений, на его основе вводятся базовые понятия.

Имея LU-разложение матрицы **A**,

,

можно непосредственно вычислить её определитель,

где **n** — размер матрицы **A**,  и  - диагональные элементы матриц **L** и **U**.

# 2 Алгоритм Краута

Суть алгоритма Краута. Предполагая, что разложение **A = L \* U** существует, запишем произведение **L \* U**:

Сравнивая компоненты этого произведения с компонентами матрицы ***A***, видим, что первый столбец произведения ***L \* U*** равен первому столбцу матрицы ***A***, т.е. , при **i = 1,…,4**. Первая строка произведения может быть использована для определения первой строки матрицы ***U***.

Действительно, т.к.

,

при , получаем

.

Поскольку во втором столбце элементы и  известны, можем определить второй столбец матрицы ***L***:

**,**

где . Теперь, т.к. известны и, можно по второй строке произведения определить вторую строку матрицы **U**:

,

при. Далее, чередуя строки и столбцы, можно аналогичным образом найти остальные элементы матриц **L** и **U**.

Чтобы получить общие соотношения, запишем произвольный элемент произведения **L \* U**:

,

где верхний предел суммы учитывает наличие нулевых элементов в матрицах **L** и **U**. Рассмотрим произвольный элемент на или под главной диагональю матрицы **A**, для которого , и заменим индекс  на .

Учитывая при этом, что  , получим

откуда

, (6.20)

при и . Аналогичным образом, рассматривая произвольный элемент над главной диагональю, для которого , и используя  вместо , находим

откуда

, (6.21)

при и

Эти соотношения для  и  есть алгоритм разложения на треугольные матрицы – алгоритм Краута. Заметим, что текущие элементы матриц  и  определяются текущим элементом матрицы  и предыдущими элементами матриц  и . Отсюда, т.к. нулевые элементы и единичную диагональ матрицы  запоминать не нужно, в процессе вычислений матрицы  и могут быть записаны на месте матрицы , причем  расположена в нижнем треугольнике , а– соответственно в верхнем треугольнике  матрицы .

Коротко алгоритм Краута как вариант чередования столбцов и строк можно представить следующей последовательностью действий:

1) Положим и перейдем к пункту 3.

2) Используя выражение (6.20), рассчитываем **k**-ый столбец матрицы **L** и, если , закончим процедуру разложения.

3) Используя выражение (6.21), рассчитываем ***k***-ую строку матрицы ***U***.

4) Положим и перейдем к пункту 2.

# ЗАКЛЮЧЕНИЕ

В процессе курсового исследования нами была проделана следующая работа:

1. Изучены основные понятия теории решения СЛАУ.
2. Рассмотрены определение LU-разложения.
3. Приведены примеры программной реализация исследуемого метода.

В результате чего, считаем, что цель курсового исследования нами достигнута, аминь.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Г.С. Шевцов, О.Г. Крюкова, Б.И. Мызникова. Численные методы линейной алгебры. Изд-во "Лань". 2011.
2. Интернет-проект «Википедия» [Электрон. ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/LU-разложение
3. Интернет-проект «Студопедия» [Электрон. ресурс]. – Режим доступа: http://studopedia.ru/3\_67266\_LU-faktorizatsiya-algoritmom-krauta.html
4. Интернет-проект «Кибер гуру» [Электрон. ресурс]. – Режим доступа: http://www.cyberguru.ru/sources/php-sources/algorithms/algoritm-krauta-nizhne-verkhnyaya-lu-dekompozitsiya-matritsy.html

# ПРИЛОЖЕНИЯ

Приложение A

Программная реализация алгоритма

Так как задачей данной курсовой работы это исследование и программная реализация визуального решение систем линейных алгебраических уровнений методом LU-разложения по алгоритму Крата, то в текущем параграфе мною представлен программный код программы реализованный через язык программирования JavaScript и язык разметки HTML.

|  |
| --- |
| <html lang="en"> |
| <head> |
| <meta charset="UTF-8"> |
| <title>Document</title> |
| </head> |
| <body> |
|  |
| <script src="matrix.js"></script> |
| <script src="http://code.jquery.com/jquery-1.11.0.min.js"></script> |
|  |
| <script type="text/x-mathjax-config"> |
| MathJax.Hub.Config({ |
| extensions: ["tex2jax.js"], |
| jax: ["input/TeX","output/HTML-CSS"], |
| tex2jax: {inlineMath: [["$","$"],["\\(","\\)"]]} |
| }); |
|  |
| </script> |
| <script src="bower\_components/MathJax/MathJax.js"></script> |
|  |
| <style type="text/css"> |
| .not-valid { |
| border: 1px solid red; |
| } |
|  |
| form { |
| text-align: center; |
| padding-top: 50px; |
| } |
|  |
| #calc { |
| margin: 0 auto; |
| } |
| #calc td input { |
| text-align: right; |
| width: 50px; |
| } |
|  |
|  |
| #result { |
| width: 650px; |
| margin: 0 auto; |
| text-align: center; |
| } |
| .bvect { |
| border-left: 1px solid black; |
| background-color: black; |
| color: white; |
| } |
| </style> |
|  |
| <script> |
| $(document).ready(function () { |
| console.log("ready!"); |
|  |
| function printTable($selector, old, n) { |
| $selector.html(''); |
|  |
| var value, size = n || old.length; |
|  |
| for (i = 0; i < size; i++) { |
| row = $('<tr>'); |
|  |
| for (j = 0; j < size; j++) { |
| value = ''; |
|  |
| if (old[i] != undefined) { |
| if (old[i][j] != undefined && !isNaN(old[i][j])) { |
| value = old[i][j]; |
| } |
| } |
|  |
| row.append( |
| $('<td>').html( |
| $('<input>').attr({ |
| value: value, |
| type: "text", |
| name: "matrix[][]" |
| }) |
| ) |
| ); |
| } |
|  |
| $selector.append(row); |
| } |
| } |
|  |
|  |
| function printTableVector($selector, old, n) { |
| $selector.html(''); |
|  |
|  |
| var value, size = n || old.length; |
|  |
| for (i = 0; i < size; i++) { |
| row = $('<tr>'); |
|  |
| value = ''; |
|  |
| if (old[i] != undefined && !isNaN(old[i])) { |
| value = old[i]; |
| } |
|  |
| row.append( |
| $('<td>').html( |
| $('<input>').attr({ |
| value: value, |
| type: "text", |
| name: "matrix[][]" |
| }) |
| ) |
| ); |
|  |
| $selector.append(row); |
| } |
| } |
|  |
| MATRIX\_DEFAULT\_SIZE = 3; |
|  |
| $tableA = $('#calc'); |
| $tableB = $('#vectB'); |
|  |
| printTable($tableA, [[1,2,4],[5,6,1],[2,4,2]]); |
| printTableVector($tableB, [1,2,3], MATRIX\_DEFAULT\_SIZE); |
|  |
|  |
| function parseMatrix() { |
| var matrix = []; |
| $('#calc tr').each(function () { |
| var elements = $('td'); |
|  |
| var arrElementsRow = []; |
| $(this).find('td input').each(function (element) { |
| arrElementsRow.push(parseFloat($(this).val())); |
| }); |
|  |
| matrix.push(arrElementsRow); |
| }); |
|  |
| return matrix; |
| } |
|  |
| function parseVector() { |
| var tmp = []; |
|  |
| $tableB.find('td input').each(function() { |
| tmp.push(parseFloat($(this).val())); |
| }); |
|  |
| return tmp; |
| } |
|  |
| var $minusMatrixRow = $('#minusMatrixRow'); |
|  |
| $('#plusMatrixRow').click(function (e) { |
| e.preventDefault(); |
|  |
| if (MATRIX\_DEFAULT\_SIZE == 2) { |
| $minusMatrixRow.removeAttr('disabled'); |
| } |
|  |
| MATRIX\_DEFAULT\_SIZE++; |
| printTable($tableA, parseMatrix(), MATRIX\_DEFAULT\_SIZE); |
| printTableVector($tableB, parseVector(), MATRIX\_DEFAULT\_SIZE); |
| }); |
|  |
| $minusMatrixRow.click(function (e) { |
| e.preventDefault(); |
|  |
| if (MATRIX\_DEFAULT\_SIZE == 3) { |
| $minusMatrixRow.attr('disabled', 'true'); |
| } |
|  |
| MATRIX\_DEFAULT\_SIZE--; |
| printTable($tableA, parseMatrix(), MATRIX\_DEFAULT\_SIZE); |
| printTableVector($tableB, parseVector(), MATRIX\_DEFAULT\_SIZE); |
| }); |
|  |
| function buildStrMatrix(data) { |
| var str = '\\begin{bmatrix} '; |
|  |
|  |
| for (i = 0; i < data.length; i++) { |
| for (j = 0; j < data.length; j++) { |
| str += data[i][j]; |
|  |
| if (j + 1 < data.length) { |
| str += ' & '; |
| } |
| } |
|  |
| if (i + 1 < data.length) { |
| str += "\\\\ "; |
| } |
| } |
|  |
| return str + ' \\end{bmatrix}'; |
| } |
|  |
| function kraut(A) { |
| var L = []; |
| var U = []; |
|  |
| var tmp = []; |
| for (j = 0; j < A.length; j++) { |
| tmp.push(0); |
| } |
|  |
| for (i = 0; i < A.length; i++) { |
| L.push(tmp.slice()); |
| U.push(tmp.slice()); |
| } |
|  |
| var n = A.length; |
| var i, j, sum, k; |
|  |
| //U миноры = 1 |
| for (i = 0; i < n; i++) { |
| U[i][i] = 1; |
| } |
|  |
|  |
| //1 столбец |
| for (i = 0; i < n; i++) { |
| L[i][0] = A[i][0]; |
| } |
|  |
| $result.append('\\[ '); |
| $result.append("L = " + buildStrMatrix(L)); |
| $result.append("U = " + buildStrMatrix(U)); |
| $result.append('\\]'); |
|  |
| var m; |
| k = 0; |
| var go, print; |
|  |
| while (k < n) { |
| print = false; |
|  |
| for (j = 0; j < n; j++) { |
| if (j > k) { |
| sum = 0; |
| tmp = []; |
| go = false; |
|  |
| for (m = 0; m < k; m++) { |
| go = true; |
| tmp.push(L[k][m] + ' \* ' + U[m][j]); |
| sum += L[k][m] \* U[m][j]; |
| } |
|  |
| if (go) { |
| tmp = '-(' + tmp.join('+') + ')'; |
| } |
|  |
|  |
| U[k][j] = (A[k][j] - sum) / L[k][k]; |
| U[k][j] = Math.round(U[k][j]\*10000)/10000; |
|  |
| print = true; |
| $result.append('\\[ {U\_'+(k+1)+'}\_'+(j+1)+' = (' + A[k][j] + tmp + ')/' + L[k][k] + ' = ' + U[k][j] + '\\]'); |
| } |
| } |
|  |
| if (print) { |
| $result.append('\\[ '); |
| $result.append("U = " + buildStrMatrix(U)); |
| $result.append('\\]'); |
| } |
|  |
| k++; |
|  |
| print = false; |
| console.log("K = " + k); |
|  |
| for (i = 0; i < n; i++) { |
| if (i >= k) { |
| print = true; |
| console.log("L[" + i + "][" + k + "] = "); |
|  |
| sum = 0; |
| tmp = []; |
| go = false; |
|  |
| for (m = 0; m < k; m++) { |
| go = true; |
| tmp.push(L[i][m] + ' \* ' + U[m][k]); |
| sum += L[i][m] \* U[m][k]; |
| } |
|  |
|  |
| if (go) { |
| tmp = '-(' + tmp.join('+') + ')'; |
| } |
|  |
| L[i][k] = A[i][k] - sum; |
| L[i][k] = Math.round(L[i][k]\*10000)/10000; |
|  |
| $result.append('\\[ {L\_'+(i+1)+'}\_'+(k+1)+' = (' + A[i][k] + tmp + ') = ' + L[i][k] + '\\]'); |
| } |
| } |
|  |
| if (print) { |
| $result.append('\\[ '); |
| $result.append("L = " + buildStrMatrix(L)); |
| $result.append('\\]'); |
| } |
| } |
|  |
| return [L, U]; |
| } |
|  |
| var $result = $('#result'); |
|  |
| $('#calculate').click(function (e) { |
| e.preventDefault(); |
| $result.html(''); |
|  |
| var matrix = []; |
| var isValid = true; |
|  |
| $('#calc tr').each(function () { |
| var elements = $('td'); |
|  |
| var arrElementsRow = []; |
| $(this).find('td input').each(function (element) { |
| var $element = $(this); |
|  |
| if ($element.val() == "") { |
| $element.addClass('not-valid'); |
| isValid = false; |
| } else { |
| $element.removeClass('not-valid'); |
| } |
|  |
| arrElementsRow.push(parseFloat($element.val())); |
| }); |
|  |
| matrix.push(arrElementsRow); |
| }); |
|  |
|  |
| if (isValid) { |
| $result.append("\\[A = " + buildStrMatrix(matrix) + "\\]"); |
|  |
| $result.append('\\[A = L \* U\\]'); |
|  |
| var result = kraut(matrix); |
| var L = result[0]; |
| var U = result[1]; |
|  |
| $result.append('\\[ '); |
| $result.append("L = " + buildStrMatrix(L)); |
| $result.append("U = " + buildStrMatrix(U)); |
| $result.append('\\]'); |
|  |
| $result.append('<hr>'); |
| $result.append('\\[A \* X = B\\]'); |
| $result.append('\\[L \* U \* X = B \\]'); |
| $result.append('\\begin{equation\*}\\begin{cases} L\*Y=B \\\\ U\*Y=X \\end{cases}\\end{equation\*}'); |
|  |
|  |
| $result.append('<hr>'); |
| $result.append('\\[L \* Y = B\\]'); |
|  |
| var i, n, B, k, summ, Y, tmpSumm, X; |
|  |
| n = L.length; |
| B = parseVector(); |
|  |
| Y = []; |
| Y[0] = B[0]/L[0][0]; |
| $result.append('\\[ {Y\_1} = '+B[0]+'/'+L[0][0]+' = '+Y[0]+'; \\]'); |
|  |
| for (i = 1; i < n; i++) { |
| summ = 0; |
| tmpSumm = []; |
|  |
| for (k = 0; k < i; k++) { |
| summ += L[i][k] \* Y[k]; |
| tmpSumm.push(L[i][k]+' \* '+Y[k]); |
| } |
|  |
| Y[i] = (B[i]-summ)/L[i][i]; |
| Y[i] = Math.round(Y[i]\*10000)/10000; |
|  |
| $result.append('\\[ {Y\_'+(i+1)+'} = (' + B[i] + ' - ' + (tmpSumm.join('+')) + ') / ' + L[i][i] +' = ' + Y[k] + '\\]'); |
| } |
|  |
| $result.append('<hr>'); |
|  |
| X = []; |
|  |
| for (i = 0; i < n; i++) { |
| X[i] = Y[i]; |
| } |
|  |
| $result.append('\\[ {X\_'+n+'} = {Y\_'+n+'} = ' + Y[n-1] + '; \\]'); |
|  |
| for (i = n-2; i >= 0; i--) { |
| summ = 0; |
| tmpSumm = []; |
|  |
| for (k = i+1; k < n; k++) { |
| summ += U[i][k] \* X[k]; |
| tmpSumm.push(U[i][k]+' \* '+X[k]); |
| } |
|  |
| X[i] = (Y[i]-summ); |
| X[i] = Math.round(X[i]\*1000)/1000; |
|  |
| $result.append('\\[ {X\_'+(i+1)+'} = ' + Y[i] + ' - (' + (tmpSumm.join('+')) + ') = ' + X[i] + '\\]'); |
| } |
|  |
| MathJax.Hub.Queue(["Typeset", MathJax.Hub]); |
| } else { |
|  |
| } |
| }); |
| $('#calculate').click(); |
| }); |
| </script> |
|  |
| <form> |
| <table style="margin: 0 auto;"> |
| <tr> |
| <td colspan="2"> |
| <h1>A \* X = B, X = ?</h1> |
| </td> |
| </tr> |
| <tr> |
| <td><h1>A</h1></td> |
| <td><h1>B</h1></td> |
| </tr> |
| <tr> |
| <td colspan="2"> |
| <button id="plusMatrixRow">+1</button> |
| <button id="minusMatrixRow">-1</button> |
| </td> |
| </tr> |
| <tr> |
| <td> |
| <table id="calc"></table> |
| </td> |
| <td> |
| <table id="vectB"></table> |
| </td> |
| </tr> |
| </table> |
|  |
| <button id="calculate">Посчитать</button> |
| <div id="result"> |
|  |
| </div> |
| </form> |
|  |
| </body> |
| </html> |

Результат выполнения программы представлен ниже (Рис. А.1, Рис. А.2, Рис. А.3).

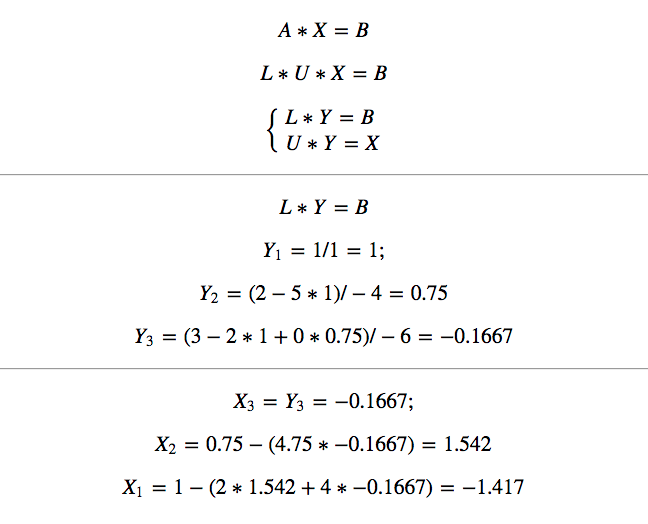
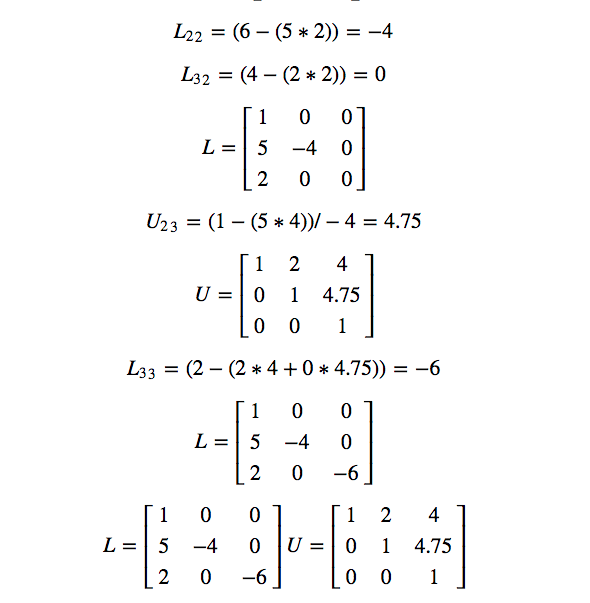
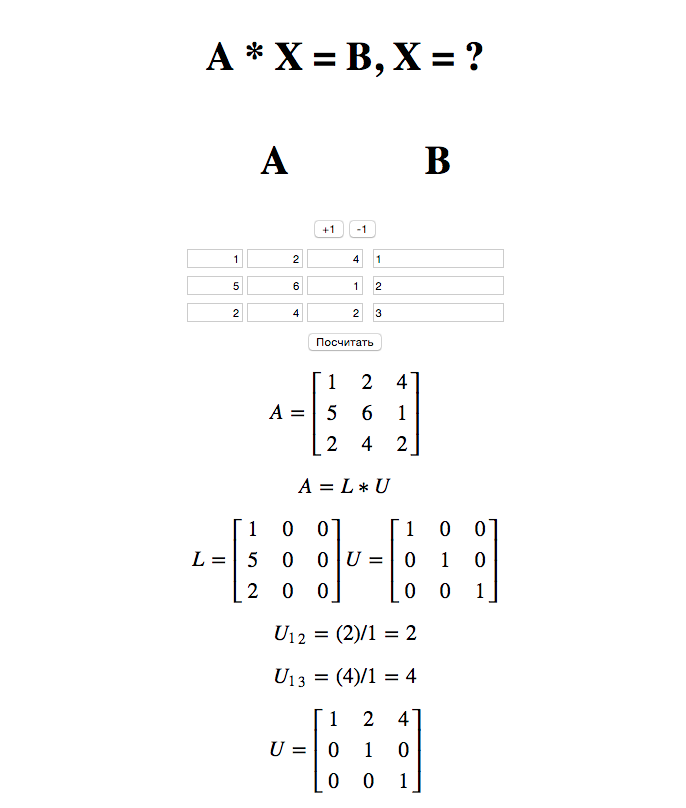


Рис.A.1

Рис.A.2

Рис.A.3