МИНОБРНАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕДЖЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет прикладной математики, информатики и механики

Отчет на тему:

Решение плоской задачи теории упругости методом конечных элементов

Выполнили студенты 1 курса магистратуры: Зобов В.В., Никуленков Е.С, Хвостова А.В., Борисова Е.С., Билай Н.А., Фролова О.В. Казьмин Д.В., Половинкин М.В.

Проверила к.т.н., доцент: Корзунина В.В.

Воронеж 2012г.

Оглавление

Постановка задачи

Численно реализовать Метод Конечных Элементов для решения плоской задачи теории упругости. Разработать средство визуализации решения.

Требования к визуализации:

1. Отображение на экране недеформированной сетки конечных элементов.
2. Отображение граничных условий (перемещений граничных узлов), если они заданы в файлах исходных данных.
3. Возможность перемещать граничные узлы на экране с целью задания граничных условий.
4. Отображение деформированной сетки.

Программа должна читать входные данные из текстовых файлов. В этих текстовых файлах должна содержаться следующая информация:

1. Матрица смежности.
2. Таблица координат узлов.
3. Множество граничных узлов: внешний контур и не более 3-х внутренних контуров.
4. Граничные условия.
5. Параметры материала E и v.

Анализом задачи и составлением математических алгоритмов решения занималась Хвостова А.В.

Разработкой архитектуры программы занимались Зобов В.В. и Никуленков Е.С.

Визуализация

Реализовал Зобов В.В.

Реализованы все требования к визуализации.

Написанные функции:

- int CoordXtoScreenX(double x), int CoordYtoScreenY(double y), double ScreenXtoCoordX(double x), double ScreenYtoCoordY(double y) Функции преобразования координат вершин в экранные и обратно.
- void CreateGrid(PaintEventArgs e). Функция создания сетки.
- void Draw(PaintEventArgs e, bool withMoves). Функция отрисовки области. Значение параметра withMoves показывает, будет ли область отрисована с перемещениями или без них.

Алгоритм Катхилла-Макки

Реализовали Половинкин М.В. и Билай Н.А.

В теории матриц алгоритм Катхилла-Макки – это алгоритм уменьшения ширины ленты разреженной матрицы симметричных матриц. Обратный алгоритм Катхилла-Макки (RCM) – это тот же самый алгоритм с обратной нумераций индексов. На практике это, как правило, лучшее решение.

Исходная симметричная матрица $n \times n$ рассматривается как матрица смежности графа (V, E). Алгоритм Катхилла — Макки перенумеровывает вершины графа таким образом, чтобы в результате соответствующей перестановки столбцов и строк исходной матрицы уменьшить ширину её ленты.

Алгоритм строит упорядоченный кортеж вершин R, представляющий новую нумерацию вершин. Для связного графа алгоритм выглядит следующим образом:

1. Выбрать периферийную вершину (или псевдопериферийную вершину) v для начального значения кортежа $R := (v)$;
2. Для $i = 1, 2, ...$, пока выполнено условие $|R| < n$, выполнять шаги 3 -5.
3. Построить множество смежности $\mathrm{Adj}(R_i)$ для R_i, где R_i — i-ая компонента R, и исключить вершины, которые уже содержатся в R, то есть: $A_i := \mathrm{Adj}(R_i) \setminus R$.
4. Отсортировать A_i по возрастанию степеней вершин.
5. Добавить A_i в кортеж результата R.

Другими словами, алгоритм нумерует вершины в ходе поиска в ширину, при котором смежные вершины обходятся в порядке увеличения их степеней.

Для несвязного графа алгоритм можно применить отдельно к каждой компоненте связности.

Временная вычислительная сложность алгоритма RCM при условии, что для упорядочения применена сортировка вставками, $O(m|E|)$, где m — максимальная степень вершины, $|E|$ — количество ребер графа.

Функции, реализующие алгоритм Катхилла-Макки:

- `int[] lastLevel(int[][] connMatrix, int N, int beginTop, out int levelNum)`. Функция, определяющая список вершин, находящихся на последнем уровне графа для вершины beginTop, и количество уровней levelNum.
- `int[] pairedTops(int[][] connMatrix, int N, int top, List<int> passed)`. Функция, определяющая список соседних не посещенных вершин для вершины top.
- `int countLink(int[][] connMatrix, int N, int top)`. Функция, определяющая количество связей у узла top.
- `int[] CuthillMcKee(int[][] connMatrix, out bool ok)`. Функция, реализующая алгоритм Катхилла-Макки.
- `void applyCuthillMcKee(CommonData c, int[] newTopsNum)`. Функция, применяющая изменения нумерации вершин, полученной в ходе алгоритма Катхилла-Макки, к исходным данным.

Входные данные

Подготовил Казьмин Д.В.

Пример текстового файла, содержащего входные данные, описанные в постановке и необходимые для решения задачи, представлен на рис. 1.

```
16
#Матрица смежности
1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0
1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0
0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0
1 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0
1 1 0 0 1 1 1 0 0 1 0 0 0 0 0 0
0 1 1 0 0 1 1 1 0 0 1 1 0 0 0 0
0 0 1 1 0 0 1 1 0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0
0 0 0 0 1 1 0 0 1 1 1 0 0 1 1 0
0 0 0 0 0 0 1 0 0 1 1 1 0 0 1 1
0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1
0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1
#Координаты
0 0
1 0
2 0
3 0
0 1
1 1
2 1
3 1
0 2
1 2
2 2
3 2
0 3
1 3
2 3
3 3
#Внешний контур
12
0 1 2 3 4 7 8 11 12 13 14 15
#Внутренние контуры
4
5 6 9 10
#Граничные условия
0
#Параметры материала
1000
0,35
```

Рис. 1. Пример текстового файла.

Чтение входных данных

Реализовал Никуленков Е.С.

Для хранения входных параметров программы, считанных из файла, был создан класс CommonData. Основным методом для загрузки данных является метод со следующей сигнатурой:

```
void Load(String fileName).
```

Работа с глобальной матрицей жесткости

Реализовал Никуленков Е.С.

Матрица жесткости представляется в виде ленточной матрицы размером (2*N)x(2*L), где N – общее количество узлов, а L – ширина ленты в матрице смежности после применения алгоритма Катхилла-Макки.

Основная процедура работы с матрицей – её генерация:

```
void GetK(double[][] K, int Ntr, Point[] Coords, int[][] Tr, int L, double E, double v).
```

Параметры:

K – матрица жесткости;

Ntr – общее количество конечных треугольных элементов;

Coords – массив координат узлов;

Tr – массив конечных элементов. Каждый элемент представлен номерами узлов его вершин;

L – ширина ленты в матрице смежности после применения алгоритма Катхилла-Макки;

E, v – параметры материала.

При генерации глобальной матрицы жесткости для каждого конечного элемента формируется его матрица жесткости, а затем значения элементов этой матрицы жесткости добавляются к соответствующим элементам глобальной матрицы жесткости.

Для генерации матрицы жесткости для одного конечного элемента используется следующая функция:

`double[][] Get_Ke(Point[] Coor, double E, double v),`

где Coor – координаты вершин конечного элемента.

Матрица жесткости ($[K]^e$) вычисляется по формулам:

$[K]^e = [B]^T[D][B] * S_\Delta$,

$$[B] = \frac{1}{2*\Delta}\begin{bmatrix} b_i & 0 & b_j & 0 & b_k & 0 \\ 0 & c_i & 0 & c_j & 0 & c_k \\ c_i & b_i & c_j & b_j & c_k & b_k \end{bmatrix},$$

$$[D] = \frac{E}{1-v^2}\begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix}.$$

Подготовка и решение системы линейных уравнений

Реализовали Никуленков Е.С. (подготовка системы) и Фролова О.В. (решение системы методом Холецкого).

После генерации глобальной матрицы жесткости строится система линейных уравнений в виде:

$K * x = f$.

Правая часть системы вычисляется на основе граничных условий, заданных пользователем. Для этого реализована функция:

`bool SetBoundaryConditions(double[][] K, int N, int L, double[] f, double[] u).`

Граничные условия заданы одномерным массивом размером 2*N (N – общее количество узлов), т.е. для каждого узла задано его перемещение в двумерном пространстве.

Учет граничных условий заключается в применении следующих формул для каждого узла с *i*-ым ненулевым граничным условием:

$f_k = f_k - K_{ki} * u_i,$

$K_{ki} = K_{ik} = 0,$

$K_{ii} = 0,$

$f_i = u_i.$

Полученная таким образом система линейных уравнений решается методом Холецкого. Реализация метода заключена в следующей функции:

```
private double[] Cholesky(double[][] A, int N, int L, double[] f).
```

Вспомогательные функции и процедуры

Реализовала Борисова Е.С.

Для поддержания работы основной логики программы были реализованы такие вспомогательные функции как:

```
int[][] GetTriangles(int N, Point[]Coords, int[][]M);
int getBandWidthOfConnectivityMatrix(int[][] connMatrix);
```

Первая функция создает массив конечных элементов на основе координат вершин, а также матрицы смежности. Вторая же считает ширину ленты матрицы смежности.

Основная процедура решения

Реализовал Никуленков Е.С.

После считывания входных данных из файла и изменения граничных условий с помощью графического интерфейса пользователя запускается основная процедура решения:

```
double[] Solve(CommonData cd).
```

Её логика работы может быть записана следующим образом:

1. Применить алгоритм Катхилла-Макки к матрице смежности с помощью функций `CuthillMcKee()` и `applyCuthillMcKee()`.
2. Определить ширину ленты полученной матрицы смежности с помощью функции `getBandWidthOfConnectivityMatrix()`.
3. Сформировать массив конечных элементов (треугольников) с помощью функции `GetTriangles()`.
4. Сгенерировать глобальную матрицу жесткости с помощью функции `GetK()`.

5. Сформировать систему линейных уравнений на основе граничных условий и глобальной матрицы жесткости с помощью функции `SetBoundaryConditions()`.
6. Решить полученную систему линейных уравнений с помощью метода Холецкого, используя функцию `Cholesky()`.

После этого происходит отрисовка полученной области с учетом полученных перемещений вершин.

Тестирование

Тест 1. Граничные условия на внешнем контуре.

Цель: проверить работу программы при наличии граничных условий на внешнем контуре.

Исходные данные представлены на рис 2.

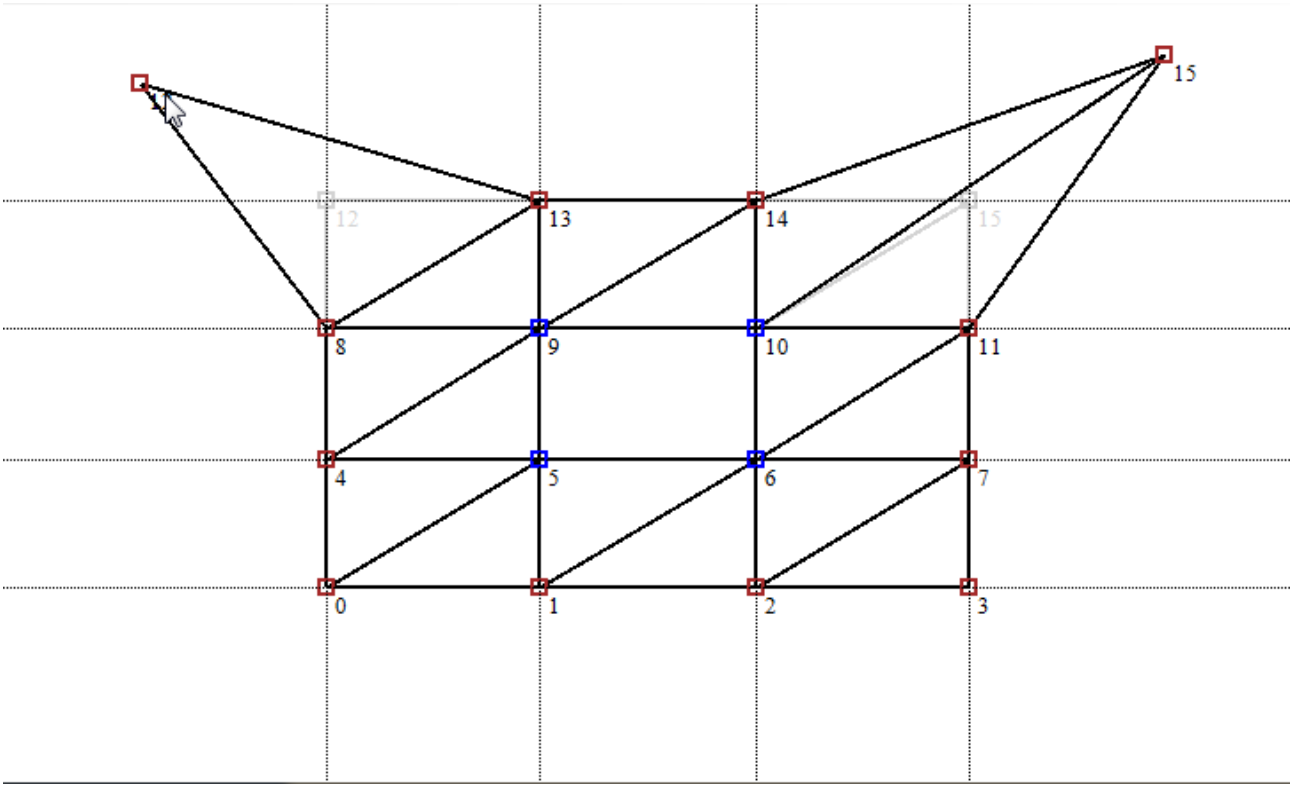

Рис. 2. Исходная область теста 1.

Ожидаемый результат: смещение всех точек области по Y вверх и растяжение области по X.

Полученный результат представлен на рис. 3.

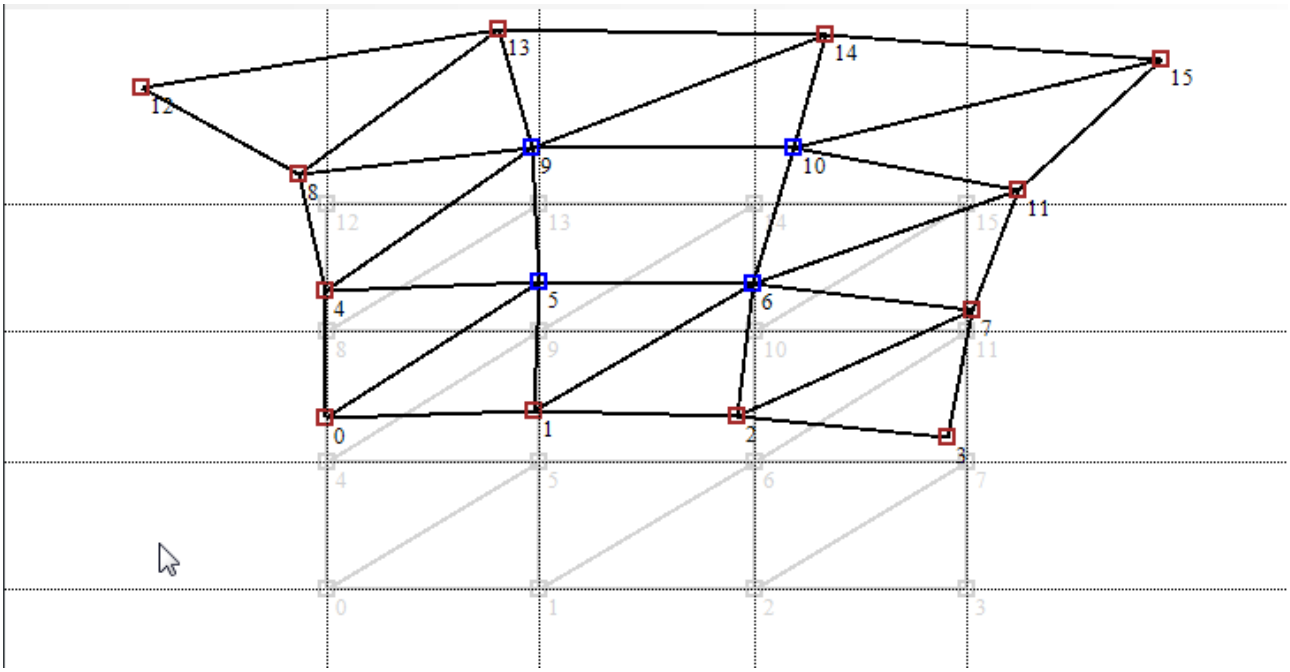

Рис. 3. Деформированная область.

Тест 2. Граничные условия на внутреннем контуре.

Цель: проверить работу программы при наличии граничных условий на внутреннем контуре.

Исходные данные представлены на рис 4.

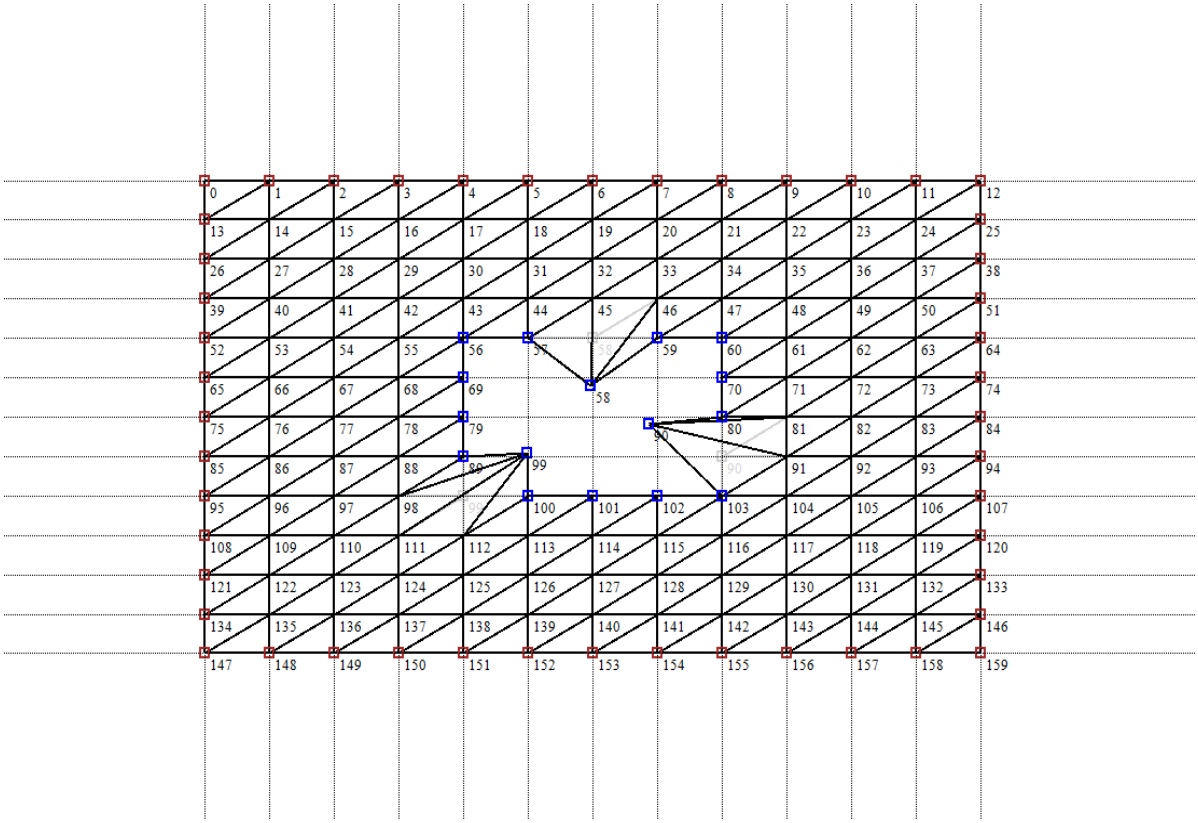

Рис. 4. Исходная область теста 2.

Ожидаемый результат: деформация области, смещение точек к центру области.

Полученный результат представлен на рис. 5.

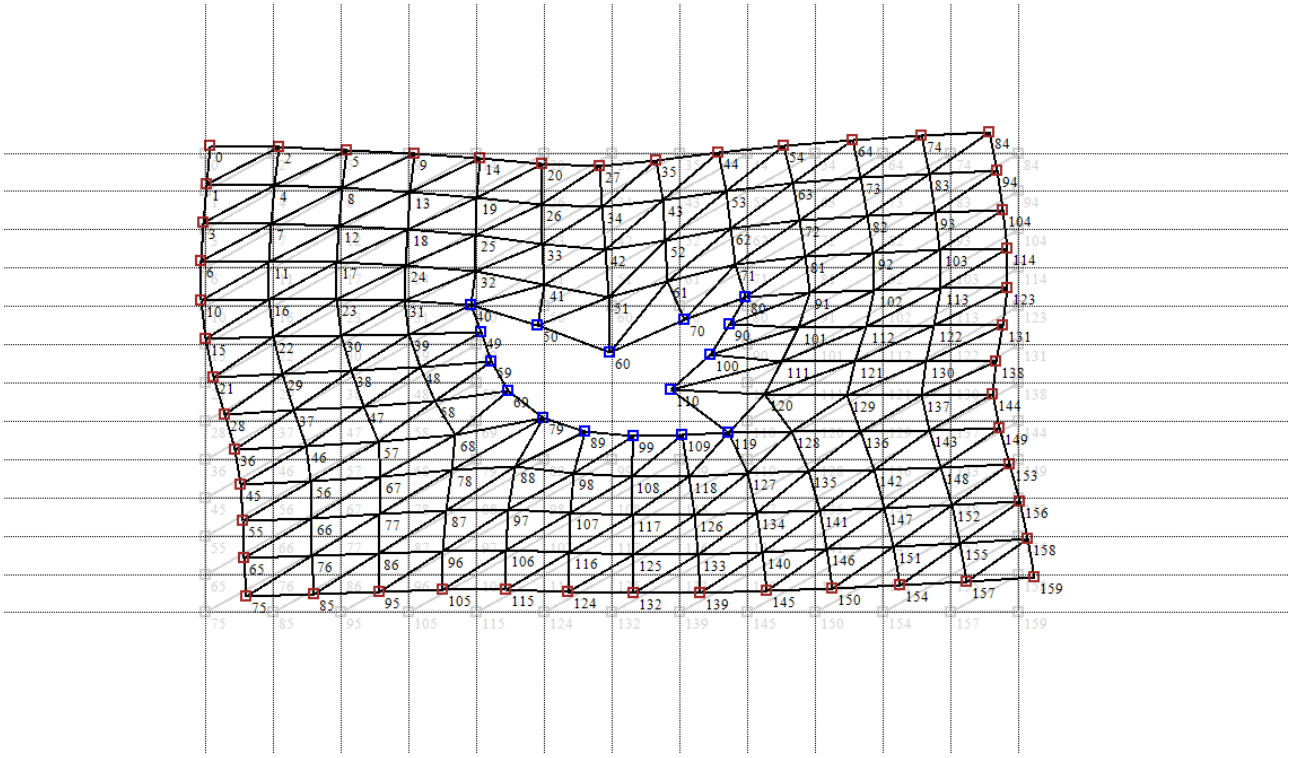

Рис. 5. Деформированная область.

Тест 3. Граничные условия на внешнем и внутреннем контуре.

Цель: проверить работу программы при наличии граничных условий на обоих контурах.

Исходные данные представлены на рис 6.

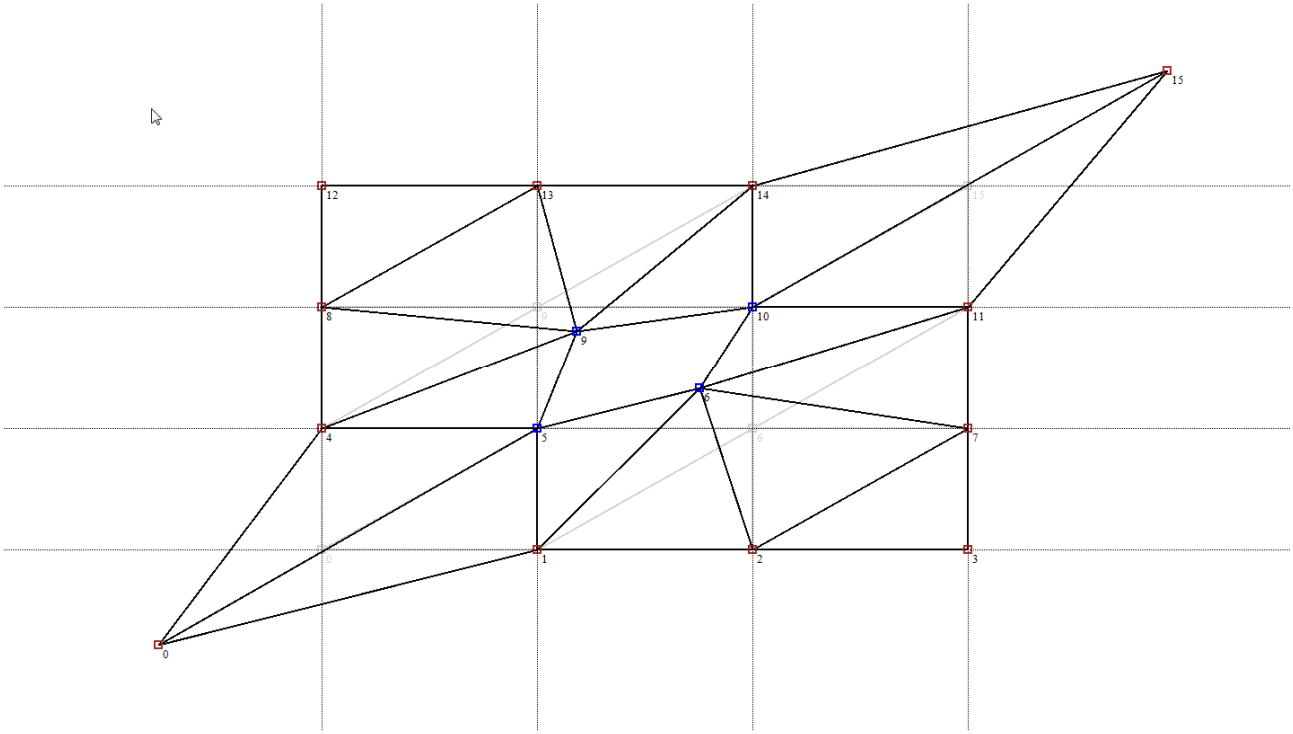

Рис. 6. Исходная область теста 3.

Ожидаемый результат: деформация области, растяжение области по линии, соединяющей точки с граничными условиями на внешней границе.

Полученный результат представлен на рис. 7.

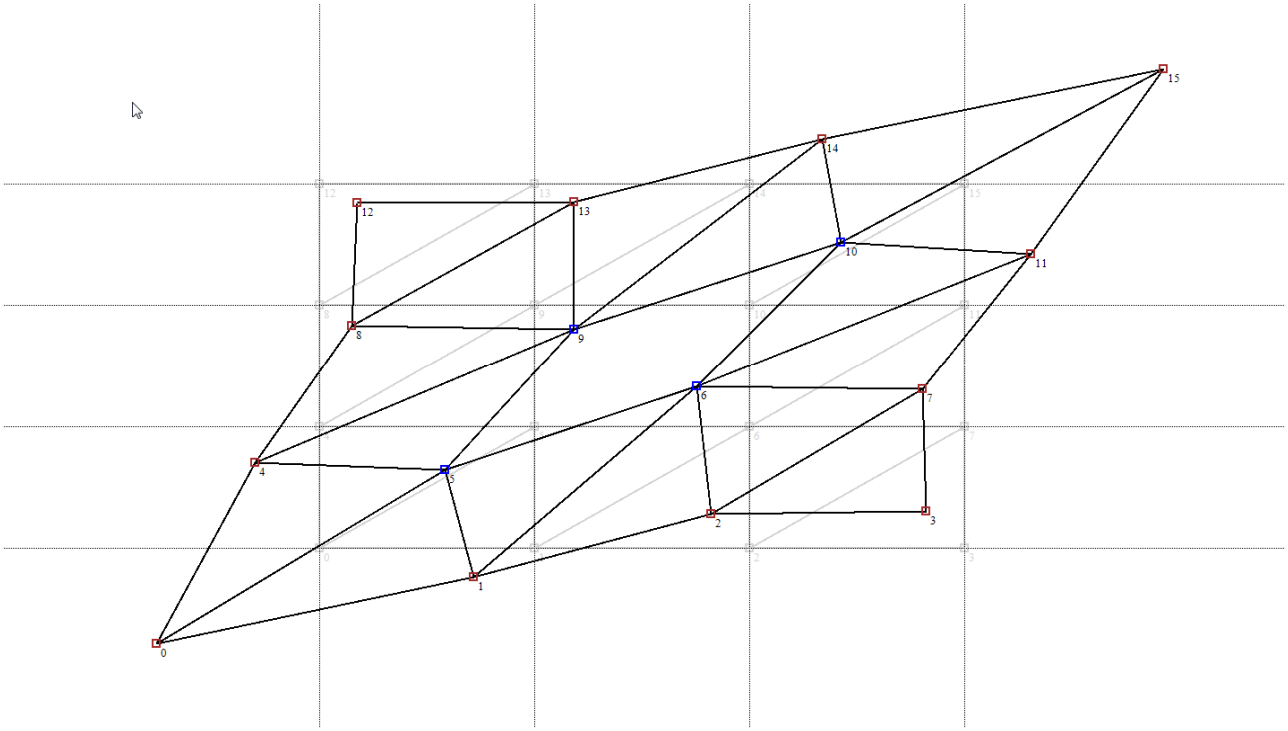

Рис. 7. Деформированная область.