**СОДЕРЖАНИЕ**

Введение 4

1 Обзор численных методов моделирования в механике 5

1.1 Численные методы решения задачи 5

1.2 Программные средства конечно-элементного анализа 10

2 Алгоритмический анализ задачи 12

2.1 Постановка задачи 12

2.2 Описание математической модели 12

2.3 Графическая схема алгоритма и ее описание 16

3 Программная реализация поставленной задачи 17

3.1 Моделирование задачи в пакете ANSYS 17

3.2 Разработка программного комплекса 20

3.3 Верификацияполученных результатов 23

3.4 Решение поставленной задачи 24

Заключение 26

Список использованных источников 27

Приложение А. Листинг класса “Node” 28

Приложение Б. Листинг класса “Element” 29

Приложение В. Листинг класса “MatrixHelper” 33

Приложение Г. Руководство пользователя 36

Приложение Д. Log-файл, созданной модели из ANSYS 41

Приложение Е. Презентация 44

**ВВЕДЕНИЕ**

Экстремальные условия работы элементов современных конструкций, сложность их формы и большие габариты делают исключительно трудным и дорогим осуществление натурного или полунатурного эксперимента, особенно, если речь идет об установлении предельных (разрушающих) нагрузок. Создание конструкций такого типа невозможно без совершенствования и автоматизации процесса проектирования, применения новых мате­риалов и технологии.

В основе любого расчета на прочность лежит расчетная схема, включающая в себя геометрию конструкции и действующие на нее нагрузки (механические и температурные). Естественно, что при создании расчетной схемы сложной конструкции прибегают к некоторой идеализации ее формы, при этом степень этой идеализации влияет на досто­верность результатов расчета.

Теории упругости и пластичности, теория пластин и оболочек и другие аналитические тео­рии решают большое количество технических за­дач, связанных с исследованием напряженно-деформированного состояния твердых тел.

Задачи со сложной геометрией обычно решаются числен­ными методами, к которым относится, в частно­сти, и метод конечных элементов.

Применение численных методов в механике деформируемого твердого тела позво­ляет решать самые разнообразные и сложные задачи теории упругости. В настоящее время для облегчения решения подобных задач разработано ряд программных ком­плексов. Среди них выделяют: *PLAXIS 3D Foundation*, *COSMOSWORKS*, *ANSYS* и др.

**1 ОБЗОР ЧИСЛЕННЫХ МЕТОДОВ МОДЕЛИРОВАНИЯ В МЕХАНИКЕ**

**1.1 Численные методы решения задачи**

При выполнении инженерных расчетов, на практике используют как аналитические, так и численные методы. Применение аналитических методов требует высокого уровня математической подготовки инженера. Кроме того, как правило, аналитические расчеты позволяют получить решение задач для тел, имеющих достаточно простую геометрическую форму и схему нагружения.

Однако применение численных методов, к которым относятся методы конечных разностей, конечных элементов, граничных элементов и другие, не ограничено ни сложностью геометрии тела, ни способами приложения нагрузок.

К наиболее хорошо известным методам относятся метод конечных разностей (МКР) и метод конечных элементов (МКЭ) [1].

*1.1.1 Метод конечных разностей.* Метод конечных разностей ‒ метод численного решения краевых задач для дифференциальных уравнений называют также методом сеток. Идея метода конечных разностей (метода сеток) известна давно, с соответствующих трудов Эйлера. Однако практическое применение этого метода было тогда весьма ограничено из-за огромного объема ручных вычислений, связанных с размерностью получаемых систем алгебраических уравнений, на решение которых требовались годы. В настоящее время, с появлением быстродействующих компьютеров, ситуация в корне изменилась. Этот метод стал удобен для практического использования и является одним из наиболее эффективных при решении различных задач математической физики.

Основная идея метода конечных разностей (метода сеток) для приближенного численного решения краевой задачи для двумерного дифференциального уравнения в частных производных состоит в том, что:

1. на плоскости в области *А*, в которой ищется решение, строится сеточная область *Аs* (рисунок 1.1), состоящая из одинаковых ячеек размером *s* (*s* – шаг сетки) и являющаяся приближением данной области *A*;
2. заданное дифференциальное уравнение в частных производных заменяется в узлах сетки *Аs* соответствующим конечно-разностным уравнением;
3. с учетом граничных условий устанавливаются значения искомого решения в граничных узлах области *Аs*.

Решая полученную систему конечно-разностных алгебраических уравнений, получим значения искомой функции в узлах сетки *Аs*, т.е. приближенное численное решение краевой задачи. Выбор сеточной области *Аs* зависит от конкретной задачи, но всегда надо стремиться к тому, чтобы контур сеточной области *Аs* наилучшим образом аппроксимировал контур области *А*.

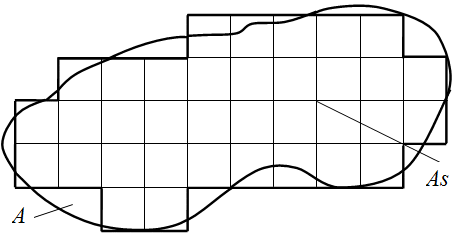


Рисунок 1.1 - Построение сеточной области

Большим преимуществом метода конечных элементов является слабая зависимость от граничных условий задачи, геометрии конструкций и характера исходного напряженного состояния. Недостатком является высокий порядок систем алгебраических уравнений. Для МКР также характерны затруднения при учете смешанных граничных условий, рассмотрении многосвязных областей и стыковок областей, описываемых различными дифференциальными уравнениями [2].

*1.1.2 Метод конечных элементов.* Метод конечных элементов (МКЭ) – основной метод современной строительной механики, лежащий в основе подавляющего большинства современных программных комплексов, предназначенных для выполнения расчетов строительных конструкций на ЭВМ. Но диапазон его применения чрезвычайно широк: строительство и машиностроение, гидро- и аэродинамика, горное дело и новейшая техника, а также различные задачи математической физики – теплопроводности, фильтрации, распространения волн и т. д.

Метод конечных элементов, как и многие другие численные методы, основан на представлении реальной континуальной конструкции ее дискретной моделью и замене дифференциальных уравнений, описывающих НДС сплошных тел, системой алгебраических уравнений. Вместе с тем МКЭ допускает ясную геометрическую, конструктивную и физическую интерпретацию. Суть метода заключается в том, что область (одно-, двух- или трехмерная), занимаемая конструкцией, разбивается на некоторое число малых, но конечных по размерам подобластей (рисунок 1.2). Последние носят название конечных элементов (КЭ), а сам процесс разбивки – дискретизацией.

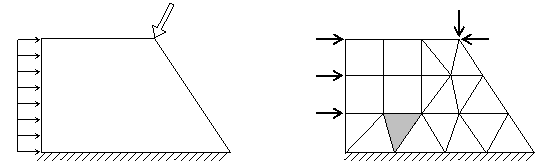


Рисунок 1.2 – Конструкция, разбитая на конечные элементы

МКЭ – это вариационный метод. Функционал энергии для всей рассматриваемой области здесь представляется в виде суммы функционалов отдельных ее частей – конечных элементов. По области каждого элемента, независимо от других, задается свой закон распределения искомых функций. Такая кусочно-непрерывная аппроксимация выполняется с помощью специально подобранных аппроксимирующих функций, называемых также координатными или интерполирующими. С их помощью искомые непрерывные величины (перемещения, напряжения и т.д.) в пределах каждого КЭ выражаются через значения этих величин в узловых точках, а произвольная заданная нагрузка заменяется системой эквивалентных узловых сил.

Метод конечных элементов позволяет практически полностью автоматизировать расчет стержневых систем, хотя, как правило, требует выполнения значительно большего числа вычислительных операций по сравнению с классическими методами строительной механики. Однако, в современных условиях большой объем вычислений не является серьезной проблемой, и, в связи с этим, при внедрении ЭВМ в инженерную практику МКЭ получил широчайшее распространение. Поэтому, знание основ метода конечных элементов и современных программных средств, позволяющих на его основе решать разнообразные задачи, в наше время для инженера является абсолютно необходимым.

Основа физической концепции МКЭ – это разбиение математической модели конструкции на непересекающиеся компоненты (подобласти) простой геометрии, называемые конечными элементами. Множество элементов, на которые разбита конструкция, называется конечно-элементной сеткой. Механическое поведение каждого элемента выражается с помощью конечного числа степеней свободы или значений искомых функций во множестве узловых точек. Поведение математической модели, таким образом, аппроксимируется поведением дискретной модели, полученной путем сборки или ансамблирования всех элементов [3].

В зависимости от типа конструкции и характера ее деформации КЭ могут иметь различную форму. Так, при расчете стержневых систем (фермы, балки, рамы) КЭ представляют собой участки стержней; для двумерных континуальных конструкций (пластины, плиты, оболочки) чаще всего применяют треугольные и прямоугольные (плоские или изогнутые) КЭ; а для трехмерных областей (толстые плиты, массивы) – КЭ в форме тетраэдра или параллелепипеда, рисунок 1.3. В отличие от реального сооружения в дискретной модели конечные элементы связываются между собой только в отдельных точках (узлах) определенным конечным числом узловых параметров.

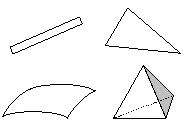


Рисунок 1.3 – Возможные виды конечных элементов

Кроме геометрии элемента существуют и другие свойства конечных элементов, атрибуты:

* Собственная размерность. Конечные элементы могут описываться одной, двумя или тремя пространственными координатами в зависимости от размерности задачи, для решения которой они предназначены. Соответствующее число внутренних или локальных координат называется собственной размерностью элемента. В динамическом анализе время рассматривается как дополнительная размерность. Отметим, что в расчетах используются также специальные элементы с нулевой размерностью, такие как, точечные массы или сосредоточенные упругие элементы (пружины).
* Узловые точки. Каждый элемент описывается множеством характерных точек, называемых узловыми точками или узлами для краткости. Узлы предназначены для описания геометрии элемента и для задания физических степеней свободы (числа неизвестных функций). Узлы обычно находятся в угловых или крайних точках элемента, но могут быть также расположены между угловыми узлами и внутри элемента. Данное различие связано с порядком аппроксимации, который обеспечивает данный конечный элемент. Элементы, имеющие только угловые узлы, называются линейными и обеспечивают линейную интерполяцию геометрии и функций. Элементы, имеющие дополнительные узлы на своих границах между угловыми точками, могут обеспечивать квадратичную или даже кубичную интерполяцию. Отметим также, что существуют элементы, имеющие внутренние узлы.
* Степени свободы. Степени свободы определяют физическое состояние элемента, т.е. физическое поле, которое описывает данный элемент. Благодаря общим степеням свободы в соседних элементах осуществляется сборка модели и формирование глобальной системы конечно-элементных уравнений. В качестве степеней свободы могут фигурировать как узловые значения неизвестной функции, так и ее производные по пространственным координатам в узлах. В первом случае элементы относятся к типу лагранжевых элементов, а во втором случае – типу эрмитовых элементов.
* Узловые силы. Система узловых сил полностью соответствует степеням свободы элемента и выражается с помощью глобального вектора узловых сил.
* Определяющие соотношения. Для конечных элементов, используемых в механических расчетах, определяющее соотношение задает поведение материала, из которого изготовлена конструкция.
* Свойства сечения. К свойствам сечения относятся площади и моменты инерции одномерных и двумерных конечных элементов, таких как балки, стержни, пластины. В эту группу также входит толщина пластин и оболочек. При построении конечного элемента свойства сечений считаются заданными и входят в результирующую матрицу жесткости элемента [4].

*1.1.3* Сравнение МКР и МКЭ представляет собой сложную задачу, так как получаемые результаты существенно зависят от конкретного случая и от используемых методов. При использовании прямоугольных областей и квадратных расчетных сеток или конечных элементов решение задач для некоторых частных случаев показало, что неявные схемы МКР и МКЭ (при использовании линейных функций формы) дают одинаковые результаты, а потому являются практически идентичными.

При расчете криволинейных областей неправильной формы МКЭ предоставляют некоторые преимущества за счет возможности построения конечных элементов с границами, которые могут быть криволинейными и не обязательно должны быть перпендикулярны друг другу. Кроме более точной аппроксимации геометрии и более точного описания изменения расчетных величин, МКЭ предоставляет следующие преимущества по сравнению с МКР:

* рассматриваемая геометрия может быть любой, поскольку она определяется независимо от компьютерной программы, это означает, что программы, реализующие МКЭ, работают независимо от геометрии;
* возможность определения расчетных параметров в любой точке рассматриваемой области;
* поскольку уравнения МКЭ решаются одновременно, существует возможность учесть все взаимодействия, имеющие место в системе, с высокой степенью гибкости и точности.

Тем не менее МКЭ тоже не свободен от недостатков:

* время, необходимое для расчетов, а также требования к аппаратным средствам компьютера и объему носителей информации в несколько раз превышают аналогичные требования для МКР;
* поскольку геометрия канала, а также начальные и граничные условия задаются пользователем самостоятельно, время, необходимое для расчета, существенно больше, чем для МКР, где эти параметры более или менее фиксированы;

**1.2 Программные средства конечно-элементного анализа**

Развитие метода конечных элементов обусловлено взаимосвязью трех факторов: наличием высокопроизводительной вычислительной техники; разработкой математических моделей исследуемых явлений, адекватных реальным процессам с достаточной степенью точности; особенностями самого метода.

Отметим, что разработка программных комплексов является дорогостоящим делом. Поэтому, как правило, организации и фирмы – собственники разработанных программ, рассматривают их как коммерческий научно-технический продукт. Регулярно печатаемые обзоры существующих комплексов программ и их характеристик, сведения о программах в отраслевых фондах алгоритмов и программ позволяют пользователям программной продукции целенаправленно выбирать необходимые для их деятельности программы расчета.

У каждой программы есть свои сильные и слабые стороны при расчете конкретной конструкции. Выбор программы расчета зависит от подготовленности пользователя в своей научной области, типа решаемой задачи, типа доступной ЭВМ, размерности задачи и других факторов.

К критериям, помогающим сделать выбор, следует отнести следующие факторы: программа широко используется; в программе используются новейшие научные достижения; программа коммерчески вполне доступна; имеется подробная и понятная документация.

Для МКЭ характерны особенности, которые следует учитывать при выборе и разработке программы расчета. Такими особенностями являются большие объемы исходных данных, промежуточных и окончательных результатов расчета. Поэтому расчет по МКЭ состоит из трех основных этапов: разработка расчетной конечно-элементной схемы и подготовка исходных данных; проверка самого расчета; обработка результатов расчета.

В настоящее время существует достаточно систем для решения задач методом конечных элементов, среди них: *ANSYS, ABAQUS, MSC.Marc, FEMLAB, COSMOS/M, CAPA, Piezo Products, PzFLEX, ATILA, FlexPDE* и т.д. Рассмотрим кратко возможности наиболее популярных из них и определим их преимущества.

*ABAQUS* изначально проектировался как пакет, ориентированный на решение физически и геометрически нелинейных задач механики. появление *ABAQUS* на рынке коммерческих конечно-элементных пакетов было знаменательным вкладом в развитие конечно-элементного анализа нелинейных задач механики. *ABAQUS* является динамично развивающейся системой и существует в виде двух независимых пакетов *ABAQUS/Standard* (неявные методы решения задач) и *ABAQUS/Explicit* (явные методы решения задач). Система имеет собственный мощный пре- и постпроцессор *ABAQUS/CAE*. Пакет позволяет использовать созданные пользователем собственные процедуры, описывающие нестандартное поведение материала, определяющие пользовательские конечные элементы и т. д. Программный конечно-элементный комплекс *ABAQUS* – универсальная система общего назначения, предназначенная как для проведения многоцелевого инженерного многодисциплинарного анализа, так и для научно-исследовательских и учебных целей в самых разных сферах деятельности, в числе которых: автомобилестроение; авиастроение и оборонная промышленность; электроника; металлургия; производство электроэнергии; нефтедобыча и переработка.

*ANSYS* является одним из самых распространенных комплексов сегодня, использующим метод конечных элементов. Многоцелевая направленность программы, независимость от аппаратных средств (от персональных компьютеров до рабочих станций и суперкомпьютеров), средства геометрического моделирования на базе B-сплайнов, полная совместимость с *CAD/CAM/CAE* системами ведущих производителей и «дружеский» интерфейс привели к тому, что именно *ANSYS* в настоящее время используется во многих университетах для обучения студентов и выполнения научно-исследовательских работ.

Она первой реализовала такие нововведения, как выполнение анализа на персональном компьютере (*РС*), интегрированное средство решения задач гидроаэродинамики (*CFD*) и многоцелевой пакет для решения сложных проблем физики и механики. Фирма *ANSYS* инвестирует исследовательские работы и дальнейшее развитие компании, гарантируя своим клиентам непрерывное пополнение программных средств с маркой *ANSYS*, неизменно отвечающих их инженерным запросам.

Многоцелевая направленность программы (т.е. реализация в ней средств для описания отклика системы на воздействия различной физической природы) позволяет использовать одну и ту же модель для решения таких связанных задач, как прочность при тепловом нагружении, влияние магнитных полей на прочность конструкции, тепломассоперенос в электромагнитном поле. Это обеспечивает всем пользователям программы удобные возможности для решения широкого круга инженерных задач [5].

Среди недостатков данных программных комплексов сильно выделяется их стоимость. Из-за высокой стоимости не каждое предприятие может позволить себе такой программный комплекс, особенно, если на предприятии решается узкий круг задач, то покупка подобных программ не является рациональной. Поэтому разработка приложений, решающий какой-то конкретный вид задач и, следовательно, имеющих гораздо меньшую стоимость, является актуальной задачей.

**2 АЛГОРИТМИЧЕСКИЙ АНАЛИЗ ЗАДАЧИ**

**2.1 Постановка задачи**

Реализацию алгоритма для анализа напряженно-деформированного состояния пространственного объекта необходимо реализовать с помощью языка высокого уровня. Результаты решения необходимо верифицировать в системе ANSYS.

Разработанное Windows-приложение,должновыполнять следующие действия:

* разбивать пространственный объект на конечные элементы;
* графически изображать пространственный объект до и после приложения нагрузки;
* проводить анализ напряженно-деформированного состояния.

Исходными данными для проекта являются: пространственный объект – параллелепипед, материал хром, размеры 600x800x1000мм, давление сверху 100Т, закрепление снизу сплошное.

2.2 Описание математической модели

Моделируемая конструкция представляет собой параллелепипед. Размеры параллелепипеда 600х800х1000мм. Длина параллелепипеда равна 600 мм, ширина равна 800 мм, а высота 1000 мм.

Распределённая нагрузка сверху равна 100Т. Материал, из которого сделан параллелепипед – хром. Модуль упругости для данного материала равен 23∙1011 Па, а коэффициент Пуассона равен 0,21.

Данные с размерами детали представлены в таблице 2.1.

Таблица 2.1 – Геометрические характеристики детали

|  |  |  |
| --- | --- | --- |
| Ширина, мм | Длина, мм | Высота, мм |
| 800 | 600 | 1000 |

Выберем в качестве исходного конечного элемента тетраэдр, к его вершинам приложим узловые усилия, формула (2.1).

 (2.1)

которым будут соответствовать узловые перемещения

 (2.2)

Для тетраэдра можно взять линейные функции для перемещений:

 (2.3)

или

 (2.4)

где  (2.5)

 (2.6)

Так как формула (2.4) имеет место для любой точки тетраэдра, то для его узлов будем иметь

 (2.7)

где

 (2.8)

 – координаты узлов тетраэдра.

Из формулы (2.7) следует

 (2.9)

где 

где  – объём элементарного тетраэдра,



где  – номера вершин элементарного тетраэдра.

Остальные значения  получаются круговой перестановкой индексов. Используя уравнения Коши и физические уравнения теории упругости, получим:

 (2.10)

 (2.11)

где , (2.12)

 (2.13)

где  - модуль сдвига и коэффициент Ламе соответственно.

Подставим (2.9) в (2.10) и (2.11), тогда получим:

 (2.14)

 (2.15)

На основании принципа возможных перемещений:

 (2.16)

Подставив в (2.16) выражения (2.14) и (2.15), учитывая, что в рассматриваемом случае все матрицы и  не зависят от координат, получим

 (2.17)

следовательно, для тетраэдра:

 (2.18)

где 

Выполнив матричные операции в (2.18), получим

 (2.19)

гдe ;



Добавляем найденную локальную матрицу жёсткости в глобальную. Формирование глобальной матрицы жесткости осуществляется в следующем порядке:

,

где *n* – количество конечных элементов, которыми дискретизирована рассматриваемая система; – элемент глобальной матрицы жесткости , характеризующий вклад *j*-го единичного перемещения в *i*-й компонент узловых сил всей системы в целом;  – элемент локальной матрицы жесткости  *r*-го конечного элемента, характеризующий вклад *j*-го единичного перемещения в *i*-й компонент узловых сил.

Разумеется, под знаком суммы ненулевой вклад дадут лишь элементы, примыкающие к узлу, в котором приложен *i*-й компонент сил. Таким образом, для всей рассматриваемой системы будет получено:

, (2.20)

где  – глобальная матрица жесткости;

 – вектор узловых перемещений всей системы;

 – вектор узловых усилий.

Для решения полученной системы линейных алгебраических уравнения выбран метод Гаусса. Он обладает рядом преимуществ по сравнению с другими методами:

* во-первых, нет необходимости предварительно исследовать систему уравнений на совместность;
* во-вторых, методом Гаусса можно решать не только СЛАУ, в которых число уравнений совпадает с количеством неизвестных переменных и основная матрица системы невырожденная, но и системы уравнений, в которых число уравнений не совпадает с количеством неизвестных переменных или определитель основной матрицы равен нулю;
* в-третьих, метод Гаусса приводит к результату при сравнительно небольшом количестве вычислительных операций;
* в-четвертых, метод Гаусса легко может быть легко распараллелен, благодаря чему можно добиться высокой производительности.

Алгоритм решения СЛАУ методом Гаусса:

1. Прямой ход: путём элементарных преобразований над строками систему приводят к ступенчатой или треугольной форме, либо устанавливают, что система несовместна. Выбирается разрешающая строка *k-*ая , где *k = 0…n* - 1, и для каждой следующей строки выполняется преобразование элементов, формула (2.21):

, (2.21)



для *i = k+1, k+2 … n–1; j = k+1,k+2 … n*.

1. Обратный ход: осуществляется определение значений неизвестных. Из последнего уравнения преобразованной системы вычисляется значение переменной *хn*, после этого из предпоследнего уравнения становится возможным определение переменной *xn-1* и так далее [7].

2.3 Графическая схема алгоритма и ее описание

Графическая схема алгоритма приведена на рисунке 2.2.

Этапы алгоритма:

* блок 1 – ввод исходных данных (размеров деталей и материала детали);
* блок 2 – разбиение детали на конечно-элементную сетку;
* блок 3 – создание для каждого конечного элемента локальной матрицы жесткости;
* блок 4 – создание глобальной матрицы жесткости на основе локальных матриц;
* блок 5 – решение системы линейных алгебраических уравнений с помощью метода Гаусса;
* блок 6 – вывод результатов решения.



Рисунок 2.2 - Графическая схема алгоритма

**3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ**

## **3.1 Моделирование задачи в пакете ANSYS**

*(Пояснительная пункт 31.docx)*

**3.2 Разработка программного комплекса**

Реализация математической модели производится на языке программирования C#.

Приложение сконструировано на основе трехслойной структуры. Что способствует расширяемости программы. Слой *DAL* позволяет получить данные об узлах и элементах из *csv*-файла. Слой *BLL*, он же *Business Logic Layer*, представляет собой бизнес-логику всего приложения. Слой GUI представляет собой интерфейс приложения. В таблицах 3.1, 3.2, 3.3, 3.4 представлены структуры основных классов. Диаграмма классов BLL слоя представлена на рисунке 3.9.

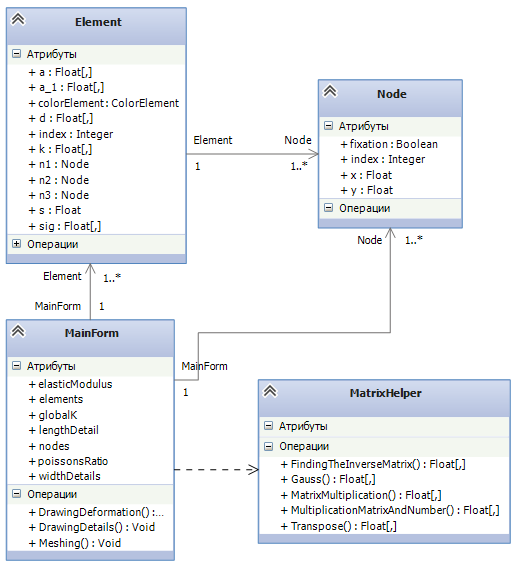


Рисунок 3.9 ­­– UML-диаграмма классов

Таблица 3.1 – Структура класса *Node*

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Комментарий** |
| x | float | Координата узла по оси OX |
| y | float | Координата узла по оси OY |
| index | int | Глобальный номер узла |
| fixation | bool | Переменная, определяющая закреплен ли узел |

Таблица 3.2 – Структура класса *Element*

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Комментарий** |
| n1 | Node | Первый узел конечного элемента |
| n2 | Node | Второй узел конечного элемента |
| n3 | Node | Третий узел конечного элемента |
| Продолжение таблицы 3.2 | | |
| **Имя переменной** | **Тип** | **Комментарий** |
| index | int | Глобальный номер элемента |
| colorElement | ColorElement | Цвет элемента для отрисовки |
| a | float[,] | Координатная матрица |
| a\_1 | float[,] | Обратная координатная матрица |
| k | float[,] | Локальная матрица жесткости |
| d | float[,] | Матица упругости |
| sig | float[,] | Смещение узлов элемента |
| s | float | Напряжение элемента |

Таблица 3.3 – Структура класса *MatrixHelper*

|  |  |  |
| --- | --- | --- |
| **Имя метода** | **Тип возвращаемого значения** | **Комментарий** |
| MatrixMultiplication | float[,] | Произведение двух матриц |
| Transpose | float[,] | Транспонирование матрицы |
| MultiplicationMatrixAndNumber | float[,] | Произведение матрицы на число |
| FindingTheInverseMatrix | float[,] | Нахождение инверсированной матрицы |
| Gaus | float[,] | Решение СЛАУ методом Гаусса |

Таблица 3.4 – Структура класса *MainForm*

|  |  |  |
| --- | --- | --- |
| **Имя переменной/**  **метода** | **Тип возвращаемого значения** | **Комментарий** |
| widthDetails | float | Ширина детали |
| lengthDetail | float | Длина детали |
| elasticModulus | float | Модуль упругости материала |
| poissonsRatio | float | Коэффициент Пуассона материала |
| nodes | List<Node> | Список узлом конечно-элементной сетки |
| elements | List<Element> | Список элементов конечно-элементной сетки |
| globalK | float[,] | Глобальная матрица жесткости |
| DrawingDeformation | void | Отрисовка деформации детали |
| Продолжение таблицы 3.2 | | |
| **Имя переменной/**  **метода** | **Тип возвращаемого значения** | **Комментарий** |
| DrawingDetails | void | Отрисовка детали |
| Meshing | void | Отрисовка конечно-элементной сетки |

Результатом работы приложения является:

* графическое отображение смещения узлом детали под действием нагрузки;
* графическое отображение распределения нагрузки по элементам детали;
* численное представления максимального смещения узлов и максимальной деформации.

Листинги разработанного программного комплекса приведены в приложениях А-В.

## **3.3 Верификация полученных результатов**

Результатом разработанного приложения стало моделирование и расчет напряжённо-деформированного состояние плоской конструкции. Для проверки результатов расчетов, поставленная задача была реализована в пакете ANSYS. Сравнение производится на самых критических участках (элементах). Данные для сравнения предоставлены в таблице 3.5.

Таблица 3.5 ­– Результаты расчетов

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ элемента** | **ANSYS** | | | **Разработанное приложение** | | | **% расхождения результатов** | | |
| **σx, МПа** | **σy, МПа** | **σxy, МПа** | **σx, МПа** | **σy, МПа** | **σxy, МПа** | **σx** | **σy** | **σxy** |
| 406 | 75,89 | 2,27 | 6,18 | 75,86 | 2,26 | 6,18 | 0,03 | 0,05 | 0,04 |
| 407 | 93,44 | 5,94 | 11,74 | 93,41 | 5,94 | 11,73 | 0,03 | 0,03 | 0,04 |
| 408 | 130,61 | 19,14 | 28,51 | 130,56 | 19,14 | 28,50 | 0,04 | 0,03 | 0,04 |
| 357 | 64,75 | 1,15 | 6,92 | 64,73 | 1,15 | 6,92 | 0,03 | 0,06 | 0,04 |
| 358 | 70,67 | 3,77 | 12,56 | 70,64 | 3,77 | 12,55 | 0,03 | 0,03 | 0,03 |
| 359 | 64,98 | 21,33 | 20,12 | 64,96 | 21,32 | 20,11 | 0,03 | 0,03 | 0,03 |
| 360 | 52,75 | 10,13 | 25,20 | 52,73 | 10,13 | 25,19 | 0,03 | 0,04 | 0,04 |
| 409 | 104,82 | 29,66 | 42,33 | 104,78 | 29,65 | 42,31 | 0,03 | 0,03 | 0,03 |
| 410 | 56,52 | 14,84 | 28,85 | 56,50 | 14,84 | 28,84 | 0,03 | 0,03 | 0,03 |
| 412 | 30,39 | 4,48 | -11,70 | 30,38 | 4,48 | -11,69 | 0,05 | 0,05 | 0,05 |
| 413 | 51,86 | 15,53 | -28,77 | 51,83 | 15,52 | -28,76 | 0,05 | 0,05 | 0,05 |
| 414 | 104,51 | 34,44 | -48,53 | 104,46 | 34,43 | -48,50 | 0,05 | 0,05 | 0,05 |
| 415 | 126,89 | 20,79 | -30,76 | 126,83 | 20,78 | -30,75 | 0,05 | 0,05 | 0,05 |
| 363 | 37,65 | 8,10 | -23,03 | 37,63 | 8,09 | -23,02 | 0,05 | 0,05 | 0,05 |
| 364 | 51,38 | 11,62 | -29,30 | 51,36 | 11,62 | -29,29 | 0,05 | 0,05 | 0,05 |
| 465 | 10,74 | 1,12 | 3,49 | 10,73 | 1,12 | 3,49 | 0,04 | 0,08 | 0,04 |
| 371 | 55,05 | -2,02 | -20,47 | 55,02 | -2,02 | -20,46 | 0,05 | 0,06 | 0,05 |
| 372 | 45,02 | -3,44 | -11,39 | 45,00 | -3,44 | -11,38 | 0,05 | 0,05 | 0,05 |
| 373 | 54,53 | -2,98 | -20,02 | 54,50 | -2,98 | -20,00 | 0,05 | 0,04 | 0,05 |
| 450 | 56,20 | -5,21 | -16,57 | 56,17 | -5,20 | -16,56 | 0,05 | 0,06 | 0,05 |

Анализ результатов показывает, что рассчитанные величины практически одинаковы (процент расхождения варьируется от 0.01% до 0.09%), т.е. результаты разработанного приложения сходятся с профессиональным пакетом *ANSYS* более чем на 99%. Из всего этого можно сделать вывод о том, что с помощью разработано приложения можно получить достаточно верное решение. Log-файл моделирования задачи в *ANSYS* приведен в приложении Д.

## **3.4 Решение поставленной задачи**

Максимально допустимое напряжения для стали равно 150МПа. Необходимо провести некоторое количество опытов для определения оптимальных размеров детали, при которых не произойдет разрушения конструкции под заданной нагрузкой (10000Н). Результаты опытов приведены в таблице 3.6.

Таблица 3.6 ­– Результаты опытов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№ опыта** | **Размеры детали** | | | | **Максимальное напряжение, МПа** | **Результат** |
| **Ширина, мм** | **Длина, мм** | **Базовое значение, мм** | **Толщина, мм** |
| 1 | 42 | 168 | 7 | 10 | 188.19 | Деталь разрушится |
| 2 | 48 | 192 | 8 | 10 | 164.67 | Деталь разрушится |
| 3 | 54 | 216 | 9 | 10 | 146.35 | Деталь выдержит |
| 4 | 60 | 240 | 10 | 10 | 131.73 | Деталь выдержит |
| 5 | 66 | 264 | 11 | 10 | 119.75 | Деталь выдержит |

Исходя из результатов, полученных в опытах можно сделать вывод о том, что при действии нагрузки в 10000Н и при уменьшении базового значения от 9 мм. (ширина детали 54 мм, длина детали 216 мм.) материал больше не сможет выдерживать такую нагрузку и деталь начнет разрушаться. Следовательно, базовое значение равное 9 мм. (ширина детали 54 мм, длина детали 216 мм) является граничным значением, при котором деталь сможет выдержать заданную нагрузку, рисунок 3.10 и 3.11.

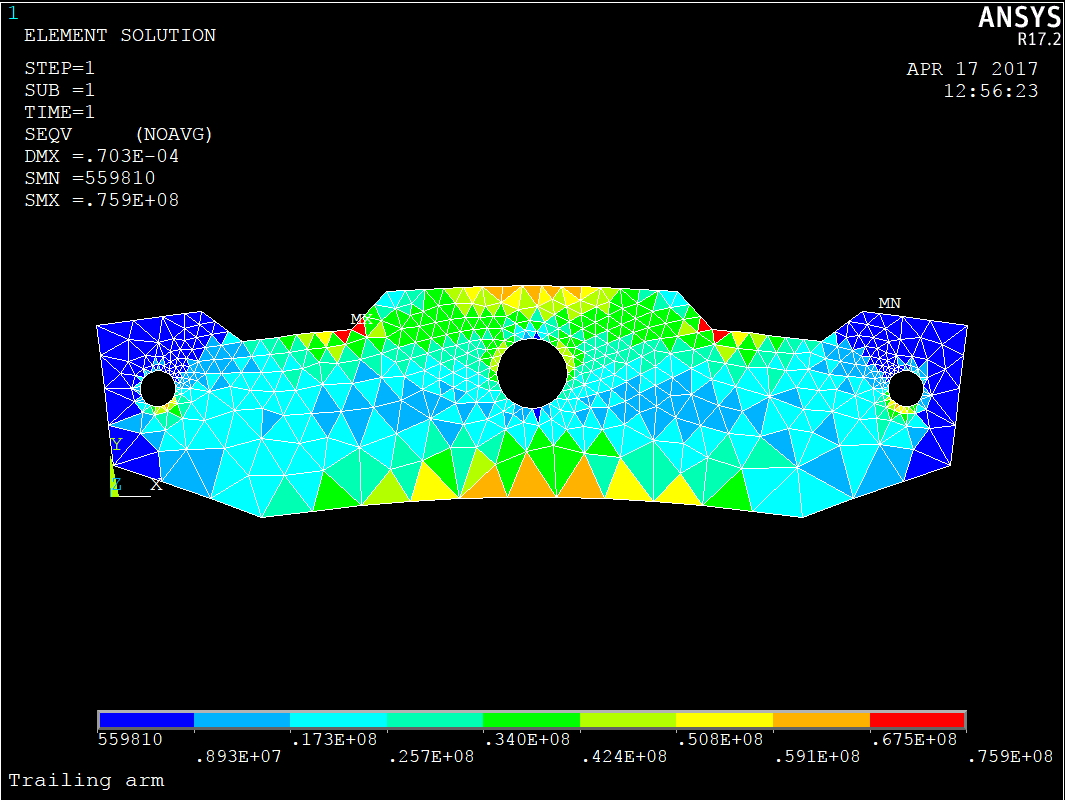


Рисунок 3.10 ­­– Решенная задача в пакете ANSYS

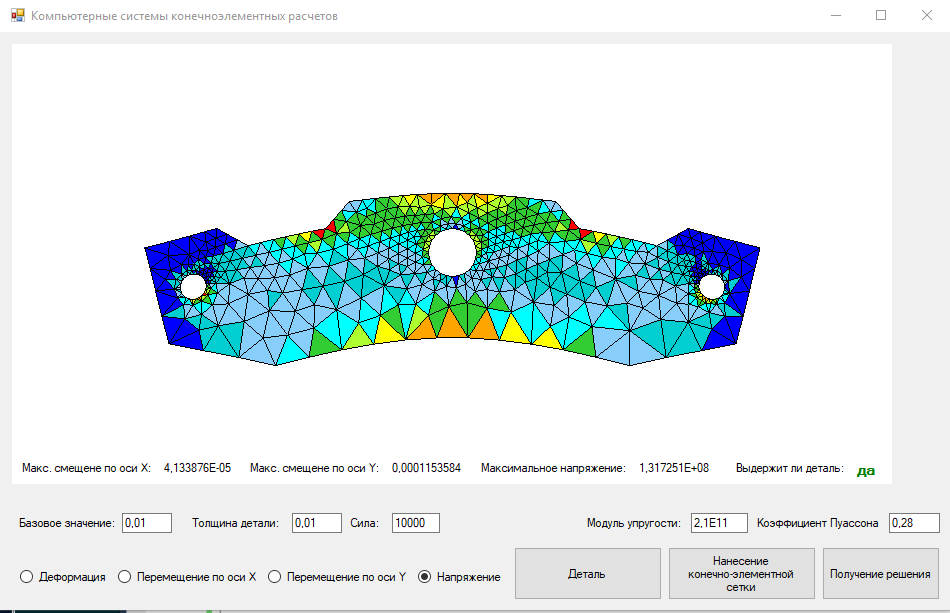


Рисунок 3.11­– Решенная задача в разработанном приложении

Руководство по использованию разработанного программного комплекса приведено в приложении Г.

# ЗАКЛЮЧЕНИЕ

Для решения поставленной задачи было разработано приложение моделирования напряжённо-деформированного состояния конструкции при наличии нагрузки. В приложении реализован графический интерфейс пользователя. Приложение позволяет быстро и эффективно решить поставленную задачу, используя метод конечных элементов. Полученные результаты перемещений и напряжений представлены как в графическом, так и численном виде. Разработанное приложение верифицировано с пакетом *ANSYS*. Результаты сравнения совпадают более чем на 99%. Т.е. различия между результатами расчётов минимальны, что говорит о правильности реализации задачи в *ANSYS* и на языке высокого уровня.

При сравнении разработанного программного комплекса с такими профессиональными пакетами как *ANSYS*, очень заметен недостаток в виде ограниченности в функционале. Однако разработанное приложение имеет очень серьезное преимущество, а именно стоимость. Ведь некоторым предприятиям, специализирующимся в некоторой узкой области, не рационально покупать такие большие пакеты как *ANSYS*. Им достаточно простого и удобного в использовании, небольшого приложения, которое в полной мере будет решать поставленную задачу.

**Список использованных источников**

1. Бахвалов, Н.С. Численные методы: Учеб. пособие / Н.П. Жидков, Г.М. Кобельков. – М.: Наука, 1987. – 600с.
2. Метод конечных разностей [Электронный ресурс]. – Режим доступа: http://www.simumath.net/library/book.html?code=Ur\_Mat\_Ph\_method\_net. – Дата доступа: 10.04.2015.
3. Метод конечных элементов [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Метод\_конечных\_элементов. – Дата доступа: 10.04.2015.
4. Варвак, П.М. Справочник по теории упругости (для инженеров - строителей) / А.Ф. Рябов. – К.: Будивельник, 1971. – 418 с.
5. Каплун, А.Б. Ansys в руках инженера: Практическое руководство / Е.М. Морозов, М.А. Олферьева. – М.: Едиториал УРСС, 2003. – 272 с.
6. Зенкевич, О. Метод конечных элементов в технике / О. Зенкевич. – М.: Мир, 1975. – 541 с.
7. Метод гаусса [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Метод\_Гаусса. – Дата доступа: 10.04.2015.

**ПРИЛОЖЕНИЕ А**

(Обязательное)

**Листинг класса “Node”**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Coursework\_kskr

{

class Node

{

float x;

float y;

int index;

bool fixation;

public Node() { }

public Node(int \_index, float \_x, float \_y, bool \_fixation)

{

index = \_index - 1;

x = \_x;

y = \_y;

fixation = \_fixation;

}

public float X

{

get { return (x); }

set { x = value; }

}

public float Y

{

get { return (y); }

set { y = value; }

}

public int Index

{

get { return (index); }

set { index = value; }

}

public bool Fixation

{

get { return (fixation); }

set { fixation = value; }

}

}

}

**ПРИЛОЖЕНИЕ Б**

(Обязательное)

**Листинг класса “Element”**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

namespace Coursework\_kskr

{

class Element

{

Node n1;

Node n2;

Node n3;

SolidBrush colorElement;

int index;

float[,] a;

float[,] a\_1;

float[,] q;

float[,] k;

float[,] b;

float[,] d;

float[,] sig;

float[,] e;

float[,] stress;

float s;

public Element(int \_index, Node \_n1, Node \_n2, Node \_n3)

{

index = \_index;

n1 = \_n1;

n2 = \_n2;

n3 = \_n3;

sig = new float[6,1];

}

public void CreateMatrix(float t, float elasticModulus, float poissonsRatio)

{

CreateA();

CreateQ();

CreateD(elasticModulus, poissonsRatio);

CreateB();

CreateK(t);

}

void CreateA()

{

a = new float[6, 6];

a\_1 = new float[6, 6];

a[0, 0] = 1;

a[0, 1] = n1.X;

a[0, 2] = n1.Y;

a[1, 3] = 1;

a[1, 4] = n1.X;

a[1, 5] = n1.Y;

a[2, 0] = 1;

a[2, 1] = n2.X;

a[2, 2] = n2.Y;

a[3, 3] = 1;

a[3, 4] = n2.X;

a[3, 5] = n2.Y;

a[4, 0] = 1;

a[4, 1] = n3.X;

a[4, 2] = n3.Y;

a[5, 3] = 1;

a[5, 4] = n3.X;

a[5, 5] = n3.Y;

a\_1 = MatrixHelper.FindingTheInverseMatrix(a);

}

void CreateQ()

{

q = new float[3, 6];

q[0, 1] = 1;

q[1, 5] = 1;

q[2, 2] = 1;

q[2, 4] = 1;

}

void CreateD(float elasticModulus,float poissonsRatio)

{

d = new float[3, 3];

float koeff = elasticModulus / (1 - poissonsRatio \* poissonsRatio);

d[0, 0] = 1;

d[0, 1] = poissonsRatio;

d[0, 2] = 0;

d[1, 0] = poissonsRatio;

d[1, 1] = 1;

d[1, 2] = 0;

d[2, 0] = 0;

d[2, 1] = 0;

d[2, 2] = (1 - poissonsRatio) / 2;

d = MatrixHelper.MultiplicationMatrixAndNumber(d, koeff);

}

void CreateB()

{

b = new float[3, 6];

b = MatrixHelper.MatrixMultiplication(q, a\_1);

}

void CreateK(float t)

{

k = new float[6, 6];

float[,] transpB = MatrixHelper.Transpose(b);

k = MatrixHelper.MatrixMultiplication(transpB, d);

k = MatrixHelper.MatrixMultiplication(k, b);

k = MatrixHelper.MultiplicationMatrixAndNumber(k, t);

float ds = triangleArea(n1.X, n1.Y, n2.X, n2.Y, n3.X, n3.Y);

k = MatrixHelper.MultiplicationMatrixAndNumber(k, ds);

}

public void SolveStress()

{

float[,] bb = MatrixHelper.MatrixMultiplication(q, a\_1);

e = MatrixHelper.MatrixMultiplication(bb, sig);

stress = MatrixHelper.MatrixMultiplication(d, e);

s = (float)Math.Sqrt(Math.Pow(stress[0, 0], 2) + Math.Pow(stress[1, 0], 2) - stress[0, 0] \* stress[1, 0] + 3 \* (Math.Pow(stress[2, 0], 2)));

}

float triangleArea(float x1, float y1, float x2, float y2, float x3, float y3)

{

float a, b, c, p;

c = (float)Math.Sqrt(Math.Pow(y1 - y2, 2) + Math.Pow(x1 - x2, 2));

a = (float)Math.Sqrt(Math.Pow(y2 - y3, 2) + Math.Pow(x2 - x3, 2));

b = (float)Math.Sqrt(Math.Pow(y1 - y3, 2) + Math.Pow(x1 - x3, 2));

p = (a + b + c) / 2;

return ((float)Math.Sqrt(p \* (p - a) \* (p - b) \* (p - c)));

}

public SolidBrush ColorElement

{

get { return (colorElement); }

set { colorElement = value; }

}

public Node N1

{

get { return (n1); }

set { n1 = value; }

}

public Node N2

{

get { return (n2); }

set { n2 = value; }

}

public Node N3

{

get { return (n3); }

set { n3 = value; }

}

public int Index

{

get { return (index); }

set { index = value; }

}

public float S

{

get { return (s); }

set { s = value; }

}

public float[,] A

{

get { return (a); }

set { a = value; }

}

public float[,] Q

{

get { return (q); }

set { q = value; }

}

public float[,] K

{

get { return (k); }

set { k = value; }

}

public float[,] B

{

get { return (b); }

set { b = value; }

}

public float[,] D

{

get { return (d); }

set { d = value; }

}

public float[,] Sig

{

get { return (sig); }

set { sig = value; }

}

public float[,] Stress

{

get { return (stress); }

set { stress = value; }

}

}

}

**ПРИЛОЖЕНИЕ В**

(Обязательное)

**Листинг класса “MatrixHelper”**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Coursework\_kskr

{

static class MatrixHelper

{

//Произведение матриц

static public float[,] MatrixMultiplication(float[,] a, float[,] b)

{

float[,] c = new float[a.GetLength(0), b.GetLength(1)];

for (int i = 0; i < a.GetLength(0); i++)

{

for (int j = 0; j < b.GetLength(1); j++)

{

for (int k = 0; k < b.GetLength(0); k++)

{

c[i, j] = c[i, j] + a[i, k] \* b[k, j];

}

}

}

return (c);

}

//Транспонирование матрицы

static public float[,] Transpose(float[,] a)

{

float[,] b = new float[a.GetLength(1), a.GetLength(0)];

for (int i = 0; i < a.GetLength(0); i++)

for (int j = 0; j < a.GetLength(1); j++)

b[j, i] = a[i, j];

return (b);

}

//Произвадение матрицы на число

static public float[,] MultiplicationMatrixAndNumber(float[,] a, float b)

{

for (int i = 0; i < a.GetLength(0); i++)

for (int j = 0; j < a.GetLength(1); j++)

a[i, j] = a[i, j] \* b;

return (a);

}

//Нахождение обратной матрицы с помощью метода Гаусса

static public float[,] FindingTheInverseMatrix(float[,] a)

{

float[,] a\_1=new float[a.GetLength(0), a.GetLength(1)];

for (int i = 0; i < 6; i++)

a\_1[i, i] = 1;

float variable;

for (int i = 0; i < 6; i++)

{

if (a[i, i] == 0)

{

int index = i + 1;

while (index < 6)

{

if (a[index, i] != 0)

break;

index++;

}

if (index != 6)

{

for (int j = 0; j < 6; j++)

{

a[i, j] = a[i, j] + a[index, j];

a[index, j] = a[i, j] - a[index, j];

a[i, j] = a[i, j] - a[index, j];

a\_1[i, j] = a\_1[i, j] + a\_1[index, j];

a\_1[index, j] = a\_1[i, j] - a\_1[index, j];

a\_1[i, j] = a\_1[i, j] - a\_1[index, j];

}

}

}

variable = a[i, i];

for (int j = 0; j < 6; j++)

{

a[i, j] = a[i, j] / variable;

a\_1[i, j] = a\_1[i, j] / variable;

}

for (int j = 0; j < 6; j++)

{

if (j != i)

{

variable = a[j, i];

if (variable != 0)

for (int k = 0; k < 6; k++)

{

a[j, k] = a[j, k] - variable \* a[i, k];

a\_1[j, k] = a\_1[j, k] - variable \* a\_1[i, k];

}

}

}

}

return (a\_1);

}

//Решение СЛАУ методом Гаусса

static public float[,] Gaus(List<Node> nodes, float[,] globalK)

{

float variable;

for (int i = 0; i < nodes.Count \* 2; i++)

{

if (globalK[i, i] == 0)

{

int index = i + 1;

while (index < nodes.Count \* 2)

{

if (globalK[index, i] != 0)

break;

index++;

}

if (index != nodes.Count \* 2)

{

for (int j = i; j < nodes.Count \* 2 + 1; j++)

{

globalK[i, j] = globalK[i, j] + globalK[index, j];

globalK[index, j] = globalK[i, j] - globalK[index, j];

globalK[i, j] = globalK[i, j] - globalK[index, j];

}

}

}

variable = globalK[i, i];

for (int j = nodes.Count \* 2; j >= i; j--)

{

globalK[i, j] = globalK[i, j] / variable;

}

for (int j = i + 1; j < nodes.Count \* 2; j++)

{

variable = globalK[j, i];

if (variable != 0)

for (int k = nodes.Count \* 2; k >= i; k--)

globalK[j, k] = globalK[j, k] - variable \* globalK[i, k];

}

}

for (int i = nodes.Count \* 2 - 1; i > 0; i--)

{

for (int j = i - 1; j >= 0; j--)

{

globalK[j, nodes.Count \* 2] -= globalK[j, i] \* globalK[i, nodes.Count \* 2];

globalK[j, i] = 0;

}

}

return (globalK);

}

}

}

**ПРИЛОЖЕНИЕ Г**

(Справочное)

**Руководство пользователя**

Разработанное приложение позволяет производить расчет напряжённо-деформированное состояние пластины под действием нагрузки.

Для запуска приложения необходимо запустить исполняемый файл “*Coursework\_kskr.exe*”. При запуске приложения перед пользователем появляется начальное окно с информацией о моделируемой детали (рисунок Г.1).

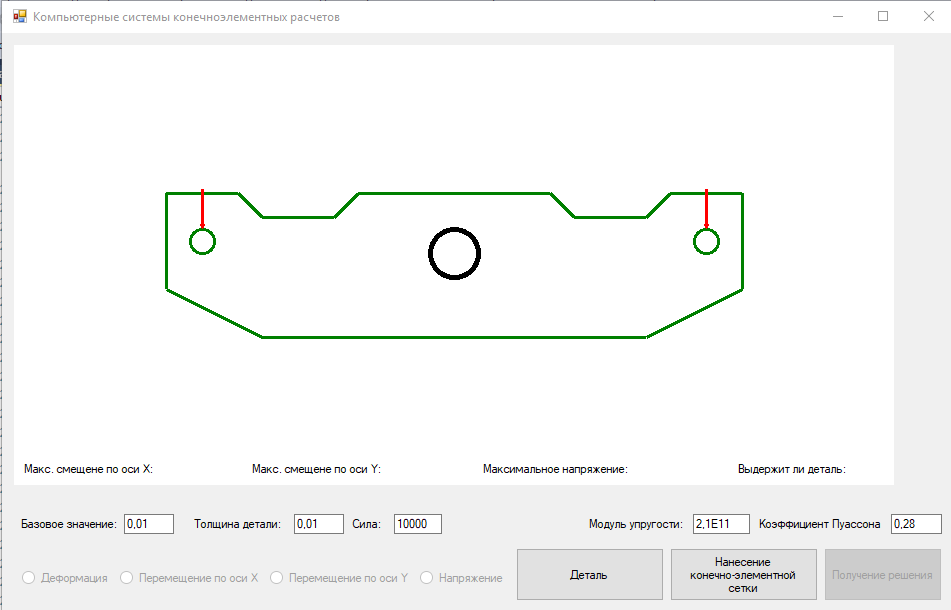


Рисунок Г.1 ­­– Начальное окно с информацией о детали

Далее необходимо определить размеры детали, которые выражаются через базовую величину (радиус отверстия) и толщину непосредственно самой детали, а также силу, рисунок Г.2.



Рисунок Г.2 ­­– Начальные условия

Затем необходимо определить материал. Ввод модуля упругости и коэффициента Пуассона вручную.

После ввода всех исходных данных можно переходить непосредственно к построению детали и вычислениям. Для построения детали необходимо нажать кнопки “Деталь”. После построения пользователю представляется моделируемая деталь (рисунок Г.5).

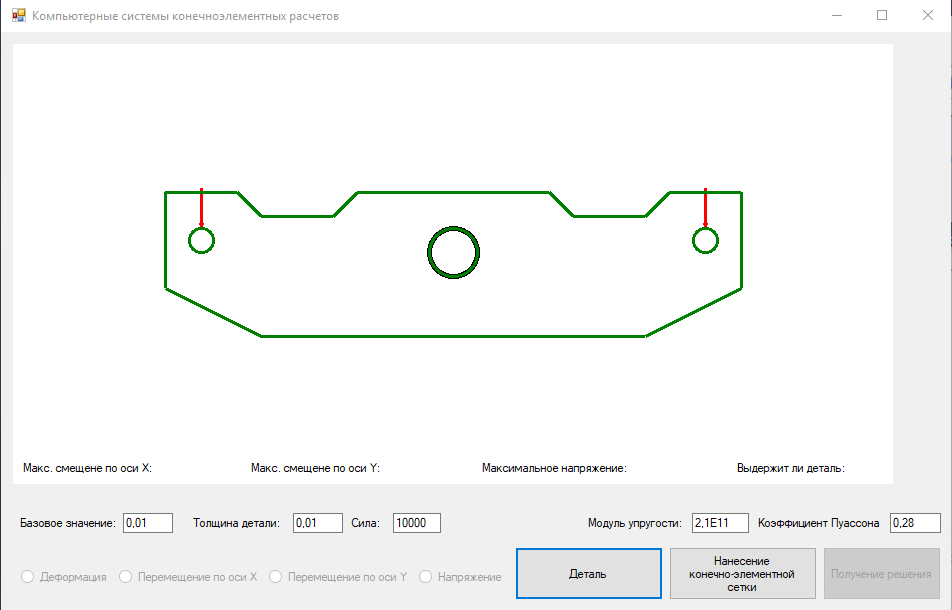


Рисунок Г.5 ­­– Смоделированная деталь

После этого необходимо построить конечно-элементную сетку, назначить силы и закрепления, для этого необходимо нажать кнопку “Нанесение конечно-элементной сетки”. Тем самым пользователь демонстрируется разбиение детали конечно-элементной сеткой, указываются узлы, на которые действует сила (выделены красным), направление действия силы (указано красной стрелкой), а также указано закрепление детали (выделено жирной линией), рисунок Г.6.

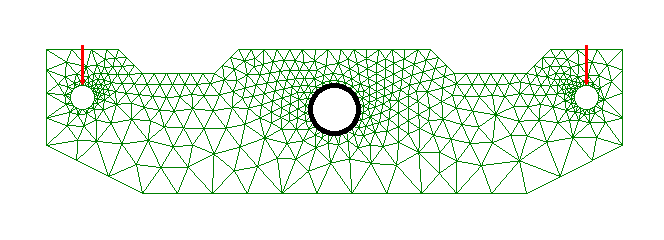


Рисунок Г.6 ­­– Конечно-элементная сетка

Далее следует начать вычисления нажав кнопку “Получение решения”. После непродолжительного ожидания, пользователю выводятся ответы на поставленную задачу, а именно: максимальные смещения узлов по осям X, Y и XY; максимальное напряжение в детали; а также вывод о том, выдержит ли заданная конструкция такую нагрузку (рисунок Г.7).



Рисунок Г.7 ­­– Результаты вычислений

Пользователю предоставляется также выбор для просмотра результатов решения в графическом виде (рисунок Г.8).



Рисунок Г.8 ­­– Выбор просмотра результатов в графическом виде

При выборе пункта “Деформация” будет представлено смещение всех узлов детали под действием заданной нагрузкой (рисунок Г.9).

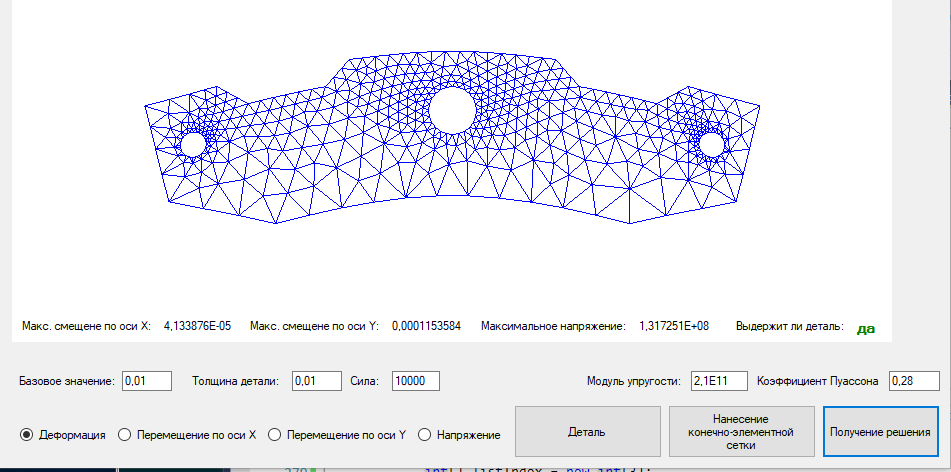


Рисунок Г.9 ­­– Графическое отображение деформации

При нажатии кнопки “Деталь” также можно посмотреть первоначальную форму детали (рисунок Г.9).

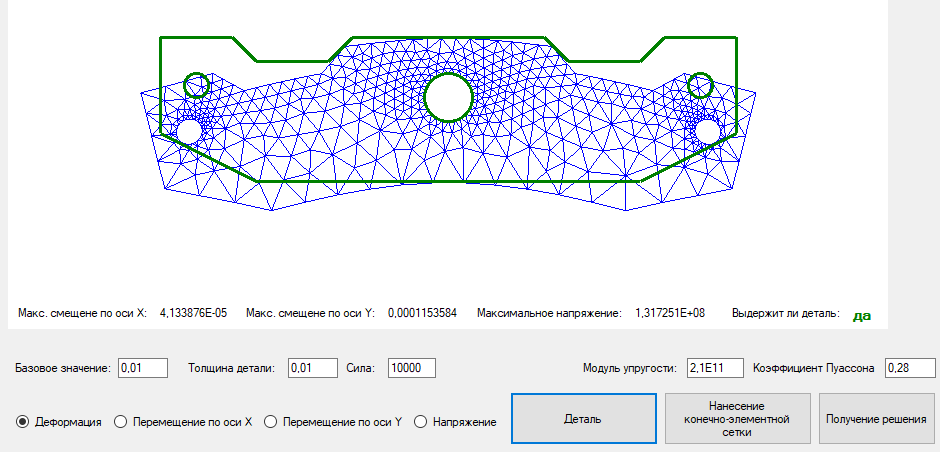


Рисунок Г.9 ­­– Графическое отображение деформации

При выборе пункта “Перемещение по оси X” или “Перемещение по оси Y будет представлена графическая градация смещения узлов детали под действием заданной нагрузкой (рисунок Г.10).

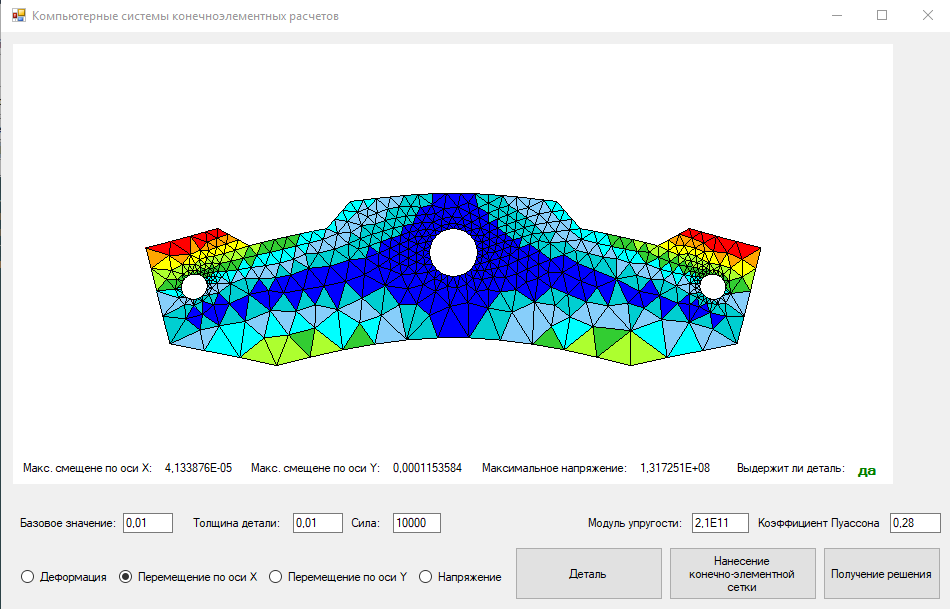


Рисунок Г.10 ­­– Градация распределения смещения узлов

При выборе пункта “Напряжение”, будет графически представлена градация распределения нагрузки для каждого конечного элемента детали под действием заданной нагрузкой (рисунок Г.11).

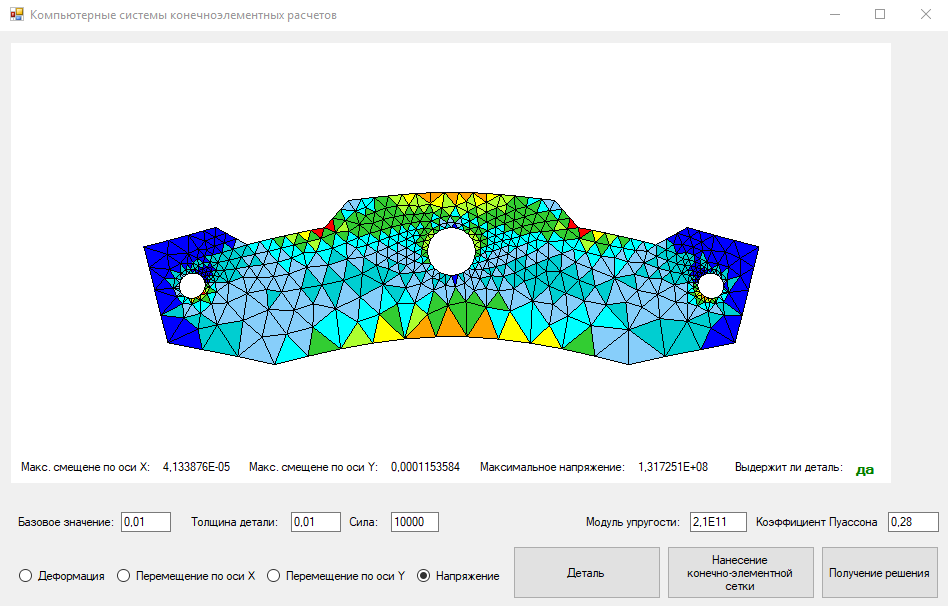


Рисунок Г.11 ­­– Градация распределения нагрузки

Для выхода из программы достаточно нажать кнопку выхода в верхнем правом углу приложения.

**ПРИЛОЖЕНИЕ Д**

(Обязательное)

**Log-файл, созданной модели из ANSYS**

/FILNAME,Trailing\_arm,0

/TITLE,Trailing arm

/UNITS, si

! Установка фильтров -------------------

/NOPR

KEYW,PR\_SET,1

KEYW,PR\_STRUC,1

KEYW,PR\_THERM,0

KEYW,PR\_FLUID,0

KEYW,PR\_ELMAG,0

KEYW,MAGNOD,0

KEYW,MAGEDG,0

KEYW,MAGHFE,0

KEYW,MAGELC,0

KEYW,PR\_MULTI,0

/GO

!\*

/COM,

/COM,Preferences for GUI filtering have been set to display:

/COM, Structural

! Выбор типа элементов -----------------

!\*

/PREP7

!\*

! Рисуем фигуру ---------------------------------------

!\*

K,1,0,0.02,,

K,2,0,0.06,,

K,3,0.03,0.06,,

K,4,0.04,0.05,,

K,5,0.07,0.05,,

K,6,0.08,0.06,,

K,7,0.16,0.06,,

K,8,0.17,0.05,,

K,9,0.20,0.05,,

K,10,0.21,0.06,,

K,11,0.24,0.06,,

K,12,0.24,0.02,,

K,13,0.2,0,,

K,14,0.04,0,,

LSTR, 1, 2

LSTR, 2, 3

LSTR, 3, 4

LSTR, 4, 5

LSTR, 5, 6

LSTR, 6, 7

LSTR, 7, 8

LSTR, 8, 9

LSTR, 9, 10

LSTR, 10, 11

LSTR, 11, 12

LSTR, 12, 13

LSTR, 13, 14

LSTR, 14, 1

FLST,2,14,4

FITEM,2,1

FITEM,2,2

FITEM,2,3

FITEM,2,4

FITEM,2,5

FITEM,2,6

FITEM,2,7

FITEM,2,8

FITEM,2,9

FITEM,2,10

FITEM,2,11

FITEM,2,12

FITEM,2,13

FITEM,2,14

AL,P51X

CYL4,0.015,0.04,0.005

CYL4,0.120,0.035,0.01

CYL4,0.225,0.04,0.005

!\*

/PNUM,ELEM,0

/REPLOT

!\*

APLOT

ASBA, 1, 2

APLOT

ASBA, 5, 3

APLOT

ASBA, 1, 4

APLOT

!\*

ET,1,PLANE182

!\*

KEYOPT,1,1,0

KEYOPT,1,3,3

KEYOPT,1,6,0

!\*

!\*

R,1,0.01,

!\*

!\*

MPTEMP,,,,,,,,

MPTEMP,1,0

MPDATA,EX,1,,2.1E+11

MPDATA,PRXY,1,,0.28

MPTEMP,,,,,,,,

MPTEMP,1,0

MPDE,EX,1

MPDE,PRXY,1

MPDATA,EX,1,,2.1E+011

MPDATA,PRXY,1,,0.28

!\*

MSHAPE,1,2D

MSHKEY,0

!\*

CM,\_Y,AREA

ASEL, , , , 2

CM,\_Y1,AREA

CHKMSH,'AREA'

CMSEL,S,\_Y

!\*

AMESH,\_Y1

!\*

CMDELE,\_Y

CMDELE,\_Y1

CMDELE,\_Y2

!\*

FLST,2,4,4,ORDE,2

FITEM,2,19

FITEM,2,-22

!\*

/GO

DL,P51X, ,ALL,0

FLST,2,4,4,ORDE,4

FITEM,2,17

FITEM,2,-18

FITEM,2,25

FITEM,2,-26

/GO

!\*

SFL,P51X,PRES,60E+6,

!\*

/STATUS,SOLU

SOLVE

FINISH

/POST1

!\*

PLESOL, S,EQV, 0,1.0

**ПРИЛОЖЕНИЕ Е**

(Обязательное)

**Презентация**

