МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра: «Информационные технологии»

направление специальности 1-40 05 01-01 «Информационные системы и технологи (в проектировании и производстве)»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту

по дисциплине: «Компьютерные системы конечно-элементных расчетов»

на тему: «Определение размеров деталей, соединяемых с помощью заклепочного соединения»

Исполнитель: студент группы ИТ-32
Гуменников Е. Д.
Руководитель: доцент
Комраков В.В.

Дата проверки: ______________________________

Дата допуска к защите: ______________________________

Дата защиты: ______________________________

Оценка работы: ____________________________

Подписи членов комиссии
по защите курсового проекта: _____________________

Гомель 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

При выполнении инженерных расчетов, связанных с анализом прочности конструкций, на практике используют численные методы. Применение аналитических методов требует высокого уровня математического аппарата. Кроме того, как правило, аналитические расчеты позволяют получить решение задач для простых тел и для простой схемы нагруженности. В то же время применение численных методов, к которым относятся методы конечных разностей, конечных элементов, граничных элементов и др., не ограничено ни сложностью геометрии тела, ни способами приложения нагрузок.

Программа ANSYS – это гибкое, надежное средство проектирования и анализа. Она работает в среде операционных систем самых распространенных компьютеров – от PC до рабочих станций и суперкомпьютеров, однако она обладает серьезным недостатком. Это дорогостоящий и многогранный, сложный продукт. Для внедрения его на узкоспециализированном малом производстве необходимы немалые средства.

Целью курсовой работы является моделирование напряжённо-деформированного состояния резьбового соединения с метрической цилиндрической резьбой воспринимающую осевую сжимающую нагрузку в пакете ANSYS и написание программного приложения на языке высокого уровня, выполняющего аналогичный расчёт конструкции методом конечных элементов и определяющую минимально необходимое количество витков резьбы. На сегодняшний день данная тема является важной, так как резьбовые соединения все еще актуальны они используются в авиа- и судостроении, металлоконструкциях и других сферах. Различным инженерам, проектирующим машины, станки, мосты, для соединения которых требуются болты, необходимо знать оптимальные размеры резьбы, какую нагрузку прилагать или какой материал использовать, чтобы конструкции не разрушилась.

Актуальность темы заключается в том, что расчёты подобного рода востребованы при реализации разнообразных конструкций.

1 ОБЗОР ЧИСЛЕННЫХ МЕТОДОВ МОДЕЛИРОВАНИЯ В МЕХАНИКЕ

1.1 Численные методы

Задачи, на которые ответ нужно дать в виде числа, как известно, решаются с помощью математических методов. На сегодняшний день существует три основных группы таких методов: аналитические, графические и численные.

При использовании аналитических методов решение задачи удается выразить с помощью формул. Например, если задача состоит в решении простейших алгебраических, тригонометрических, дифференциальных и т.д. уравнений, то использование известных из курса математики приемов сразу приводит к цели.

Преимущество аналитических методов: в результате применения аналитических методов за небольшой отрезок сразу получается точный ответ.

Недостаток аналитических методов: аналитические методы применимы лишь к небольшому числу, как правило, не очень сложных по своей структуре задач. Так, например, до сих пор не удалось решить в общем виде уравнение пятой степени.

Основная идея графических методов состоит в том, что решение находится путем геометрических построений. Например, если уравнение не удается решить аналитически, то строят график функции и абсциссу точки пересечения его с осью берут за приближенное значение корня.

Недостаток графических методов: в результате применения графических методов ответ получается с погрешностью, недопустимой в силу своей большой величины.

Основным инструментом для решения сложных математических моделей и задач в настоящее время являются численные методы. Они сводят решение задачи к выполнению конечного числа арифметических действий над числами и дают результат в виде числового значения с погрешностью, приемлемой для данной задачи.

Численные методы разработаны давно. Однако при вычислениях вручную они могли использоваться лишь для решения не слишком трудоемких задач. С появлением компьютеров, которые за короткое время могут выполнить миллиарды операций, начался период бурного развития численных методов и внедрения их в практику.[1].

1.2 Понятие метода конечных элементов

В последнее время широкую известность приобрело одно из направлений диакоптики– метод конечных элементов. Этот метод является одним из вариационных методов и часто трактуется как метод Ритца. Область, занимаемая телом, разбивается на конечные элементы. Чаще всего это треугольники в плоском случае и тетраэдры в пространственном. Внутри каждого элемента задаются некоторые функции формы, позволяющие определить перемещения внутри элемента по перемещениям в узлах, т. е. в местах стыков конечных элементов. За координатные функции принимаются функции, тождественно равные нулю всюду, кроме одного конечного элемента, внутри которого они совпадают с функциями формы. В качестве неизвестных коэффициентов метода Ритца берутся узловые перемещения. После минимизации функционала энергии, получается алгебраическая система уравнений (так называемая основная система). Таким образом, ситуация здесь такая же, как и в вариационных разностных методах, в которых для получения разностной системы уравнений применяется один из вариационных принципов.

Метод конечных элементов (МКЭ) – основной метод современной вычислительной механики, лежащий в основе подавляющего большинства современных программных комплексов, предназначенных для выполнения расчетов инженерных конструкций на ЭВМ. МКЭ используется для решения разнообразных задач, как в области прочностных расчетов, так и во многих других сферах: гидродинамике, электромагнетизме, теплопроводности и др.

Метод конечных элементов позволяет практически полностью автоматизировать расчет механических систем, хотя, как правило, требует выполнения значительно большего числа вычислительных операций по сравнению с классическими методами механики деформируемого твердого тела. Современный уровень развития вычислительной техники открывает широкие возможности для внедрения МКЭ в инженерную практику[2].

1.3 Метод конечных разностей

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 A, в которой ищется решение, строится сеточная область As 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 As соответствующим конечно-разностным уравнением;

3) с учетом граничных условий устанавливаются значения искомого решения в граничных узлах области As[3].

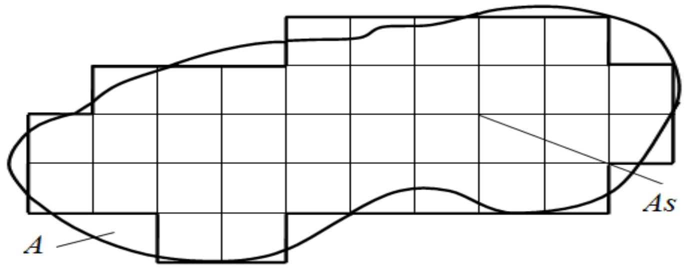

Рисунок 1.1 - Построение сеточной области

1.4 Сравнение МКР и МКЭ

МКР обладает следующими преимуществами:

– МКР позволяет рассчитывать геометрические конфигурации, к которым не-применимы простые расчетные методы на основе аналитических решений;

– МКР позволяет получить расчетные значения во всех точках сетки, а не только интегральные значения по поперечному сечению канала;

– точность получаемых результатов повышается за счет того, что в систему дифференциальных уравнений можно включить практически любой закон течения материала;

– возможность учета взаимосвязи дифференциальных уравнений (например, уравнений движения и энергии, зависимостей температуры и скорости сдвига от вязкости);

– так как система уравнений решается одновременно во всей расчетной области, результаты отражают все эффекты взаимодействия (это утверждение не является справедливым для явных разностных схем).

Однако наряду с перечисленными преимуществами МКР обладает рядом внутренне присущих недостатков:

– определение геометрии или рабочей точки возможно только методом итераций, поскольку уравнения необратимы;

– расчеты невозможно выполнить вручную или на карманном калькуляторе; для их осуществления необходим, как минимум, персональный компьютер;

– для выполнения расчетов требуется существенно большее время по сравнению с простыми аналитическими методами.

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 гиб-кости 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. Для решения задач этим методом требуется как минимум высокопроизводительный 16- или 32-разрядный ПК. За редким исключением, применение программ, реализующих МКЭ, ограничивается плоскими задачами;

– 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;

– большая гибкость МКЭ, касающаяся выбора геометрии, плотности сетки, вы-бора типов элементов и граничных условий требует от пользователя более глубокого понимания сущности данного метода, иначе получение надежных результатов становится проблематичным.

В связи с изложенными преимуществами и недостатками для решения поставленной задачи лучше использовать МКЭ.

2 АЛГОРИТМИЧЕСКИЙ АНАЛИЗ ЗАДАЧИ

2.1 Постановка задачи

Для решения поставленной задачи требуется разработать приложение на языке высокого уровня, моделирующее напряжённо-деформированное состояние плоской конструкции, провести аналогичный расчет в системе ANSYS (рисунок 2.1).

Разработанное приложение, должно выполнять следующие действия:

– считывать данные из текстовых файлов и выполнять их разбор;

– графически изображать конструкцию до и после приложения нагрузки.

– выполнять вычисления необходимого количества витков резьбы для восприятия указанной нагрузки без разрушений.

Исходными данными для проекта являются схема плоской конструкции с приложенной сжимающей нагрузкой, значения характеристик материала (модуль упругости, коэффициент Пуассон), размеры пластинки (толщина), нагрузка.

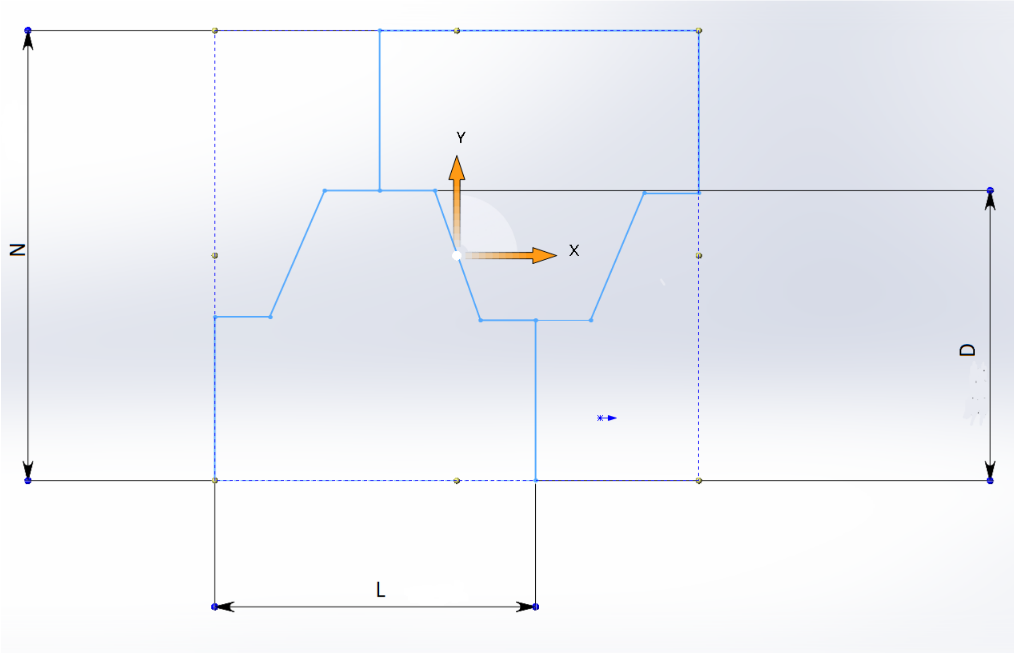

Рисунок 2.1 – Вид моделируемой конструкции.

Данные с размерами детали представлены в таблице 2.1.

2.2 Этапы решения задачи

Основные этапы решения задач с применением МКЭ могут быть представлены в виде схемы (рисунок 2.2).

Первая стадия – геометрическое моделирование включает создание геометрии модели конструкции, пригодной для МКЭ, с учетом всех параметров, которые могут оказать существенное влияние на результаты расчетов. На этой стадии помимо ввода геометрических параметров конструкции задаются физические свойства материалов, из которых она изготовлена. На этапе создания сетки конечных элементов выясняется целесообразность использования различных видов конечных элементов (оболочечных, балочных, пластин, объемных и т. д.) в рассматриваемой модели. На этой стадии выполняются мероприятия по созданию максимально возможного количества областей с регулярной сеткой конечных элементов. В местах, где предполагаются большие градиенты напряжений, необходима более мелкая сетка. На стадии моделирования граничных условий учитывают, как действие активных сил, так и наложенных на систему связей. Приложение силовых факторов должно учитывать особенности реальной работы конструкции при рассматриваемых режимах эксплуатации. Количество связей должно быть достаточным, чтобы обеспечить построение кинематически неизменяемой модели. Численное решение системы уравнений равновесия выполняется, как правило, автоматически с использованием ЭВМ. На пятом этапе проводят анализ полученных результатов путем получения полей законов распределения напряжений и деформаций, а также построения необходимых графических зависимостей либо табличных форм вывода результатов.

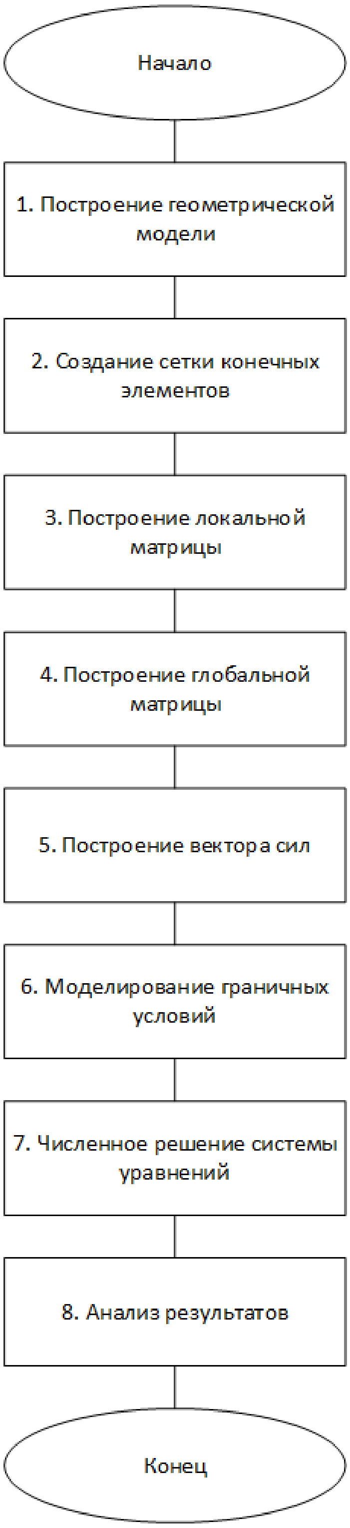

Рисунок 2.2 – Основные этапы решения задачи с применением МКЭ

2.3 Создание сетки

Ключевой шаг в методе конечных элементов создание сетки. Это процесс разделения модели на небольшие части (конечные элементы). Сеть узлов и элементов называется сеткой. Для сетки с большим количеством узлов количество элементов больше, чем узлов. Отношение между элементами и узлами приблизительно 2:1 для 2D неструктурной сетки и 6:1 для 3D неструктурной сетки с шестигранными элементами.

Меньший размер ячейки сетки h соответствует большему количеству конечных элементов в модели. Время расчета увеличивается по экспоненте с уменьшением размера ячейки. Количество ошибок уменьшается для более плотной сетки, но все равно полностью не устраняются (Рисунок 2.3).

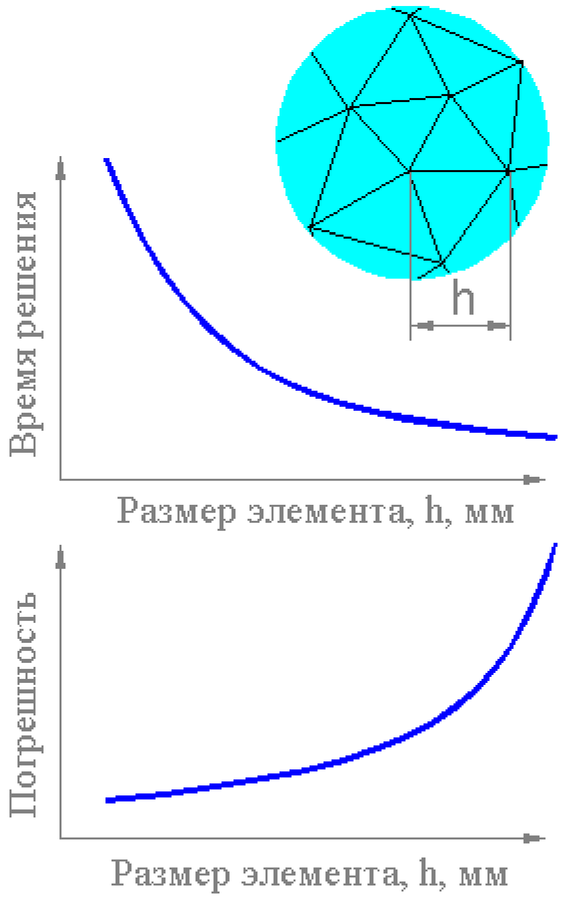

Рисунок 2.3 – Графики зависимости времени решения и погрешности от размера элемента

Точность решений зависит от допущений, сделанных в пределах типов элемента и сетки. Плотная сетка требуется, где есть изменения напряжений и деформаций (изменяются на порядок). Редкая сетка используется в областях достаточно постоянного напряжения или зон, которые не интересуют пользователя. Пользователь должен быть способен идентифицировать области концентрации напряжений. Интересующие точки могут быть в точках разрушения предварительно испытанной структуры, отверстиях, галтелях, углах, зонах контакта и в областях с высоким напряжением.

При создании сетки необходимо руководствоваться следующими правилами:

– Точность уменьшается, если размеры соседних элементов около концентраторов напряжений сильно различаются;

– форма конечных элементов влияет на точность. В конечных элементах предпочтительно не иметь тупых углов. Элементы с одинаковыми сторонами дают меньше ошибок (Рисунок 2.4 а).

– Сетку конечных элементов строят без промежутков между элементами (Рисунок 2.4 б). И треугольные и прямоугольные элементы могут быть использованы в одной и той же модели.

– Узлы нумеруются последовательно при ручном создании сетки. Запрещается строить 4-узловые элементы с тупым (> 90°) внутренним углом (Рисунок 2.4в).

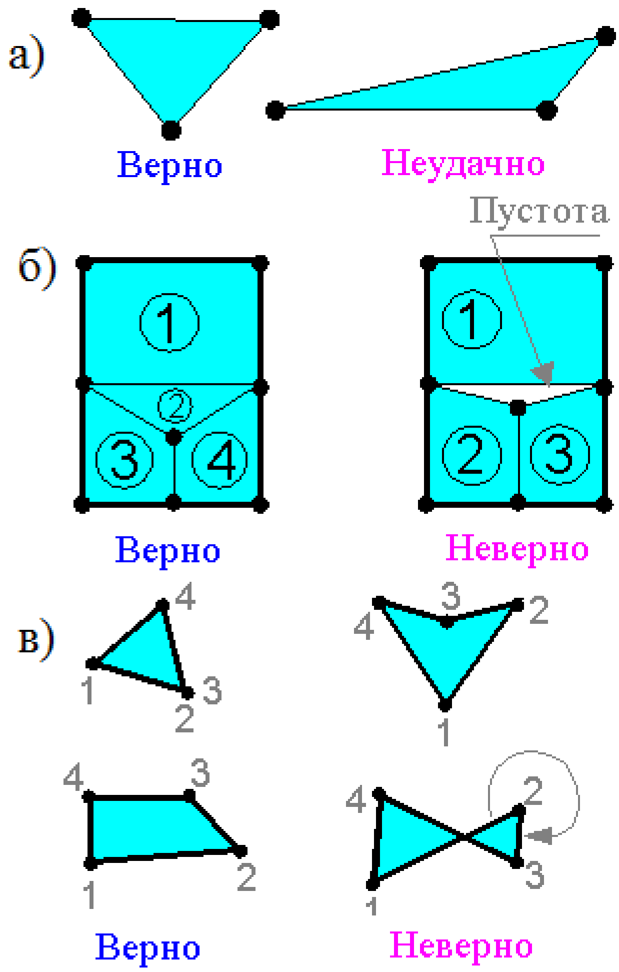

Рисунок 2.4 – Примеры построения сетки

2.4 Описание математической модели

Для расчета вся деталь разбивалась на треугольные элементы как показано на рисунке 2.5. Каждый элемент состоит из трех узлов.

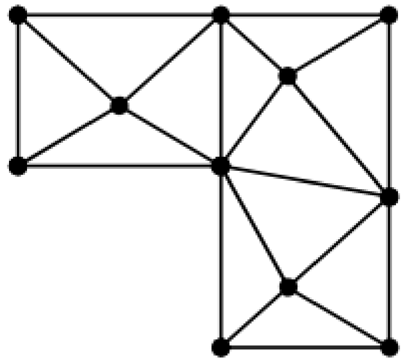

Рисунок 2.5 – Разбиение области на треугольные элементы

Перемещения каждого узла имеют два компонента, формула (2.1):

$$\{\delta_i\} = \begin{Bmatrix} u_i \\ v_i \end{Bmatrix} \qquad (2.1)$$

где u_i, v_i –перемещение узла по оси OX и OY. Шесть компонентов перемещений узлов элемента образуют вектор перемещений {δ}, формула (2.2):

$$\{\delta\} = \{u_i \quad v_i \quad u_j \quad v_j \quad u_k \quad v_k\}^T \qquad (2.2)$$

Матрица коэффициентов имеет вид, формула (2.3):

$$[A] = \begin{Bmatrix} u_0 \\ v_0 \\ u_1 \\ v_1 \\ u_2 \\ v_2 \end{Bmatrix} = \begin{bmatrix} 1 & x_0 & y_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_0 & y_0 \\ 1 & x_1 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & y_1 \\ 1 & x_2 & y_2 & o & o & o \\ o & o & o & 1 & x_2 & y_2 \end{bmatrix} \qquad (2.3)$$

где x_0, x_1, x_2, y_0, y_1, y_2– координаты первого, второго, третьего узла.

Матрица дифференциальных операторов имеет вид, формула (2.4):

$$[Q] = \begin{Bmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial y} \\ \dfrac{\partial v}{\partial x} + \dfrac{\partial u}{\partial y} \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{2.4}$$

Деформации и перемещения связаны между собой следующим образом:

$$\{\varepsilon\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = \begin{pmatrix} \partial/\partial x & 0 \\ 0 & \partial/\partial y \\ \partial/\partial y & \partial/\partial x \end{pmatrix} \begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{Bmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial v}{\partial y} \\ \dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x} \end{Bmatrix} \tag{2.5}$$

Формулу (2.6) можно представить в виде:

$$\{\varepsilon\} = [D][B]\{\delta\} \tag{2.6}$$

Напряжение элемента $-\sigma$ считается по формуле (2.7)

$$\sigma = \sqrt{\varepsilon_x^2 + \varepsilon_y^2 - \varepsilon_x \varepsilon_y + 3\gamma_{xy}} \tag{2.7}$$

При плоском деформированном состоянии в изотропном материале матрица упругих постоянных $[D]$ определяется по формуле (2.8):

$$[D] = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \dfrac{\nu}{1-\nu} & 0 \\ \dfrac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \dfrac{1-2\nu}{2(1-\nu)} \end{bmatrix} \tag{2.8}$$

где E – модуль упругости, ν – коэффициент Пуассона.,
B – градиентная матрица вида, формула (2.9):

$$[B] = [Q][A] \tag{2.9}$$

Матрица жесткости конечного элемента имеет вид:

$$[K] = \int [B]^T [D][B] h^e dxdy \tag{2.10}$$

где h^e – толщина элемента, а интегрирование производится по площади треугольника. Если предположить, что толщина элемента постоянна, что тем ближе к истине, чем меньше размеры элемента, то поскольку ни одна из матриц не содержит *x* или *y,* имеем простое выражение:

$$[K^e] = [B^e]^{\mathrm{T}}[D^e][B^e]h^e A^e \tag{2.11}$$

где A^e – площадь элемента.

Для учета условий закрепления существует следующий метод. Пусть имеется некоторая система *N* уравнений (2.12):

$$\begin{vmatrix} K_{11} & K_{12} & K_{13} & \cdot & K_{1n} & \cdot & K_{1N} \\ K_{21} & K_{22} & K_{23} & \cdot & K_{2n} & \cdot & K_{2N} \\ K_{31} & K_{32} & K_{33} & \cdot & K_{3n} & \cdot & K_{3N} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ K_{n1} & K_{n2} & K_{3n} & \cdot & K_{nn} & \cdot & K_{nN} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ K_{N1} & K_{N2} & K_{N3} & \cdot & K_{Nn} & \cdot & K_{NN} \end{vmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \\ \cdot \\ U_n \\ \cdot \\ U_N \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ \cdot \\ F_n \\ \cdot \\ F_N \end{Bmatrix} \tag{2.12}$$

В случае, когда одна из опор неподвижна, т.е. $U_i=0$, используют следующую процедуру. Пусть $U_2=0$, тогда:

$$\begin{vmatrix} K_{11} & 0 & K_{13} & \cdot & K_{1n} & \cdot & K_{1N} \\ 0 & 1 & 0 & \cdot & 0 & \cdot & 0 \\ K_{31} & 0 & K_{33} & \cdot & K_{3n} & \cdot & K_{3N} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ K_{n1} & 0 & K_{3n} & \cdot & K_{nn} & \cdot & K_{nN} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ K_{N1} & 0 & K_{N3} & \cdot & K_{Nn} & \cdot & K_{NN} \end{vmatrix} \begin{Bmatrix} U_1 \\ 0 \\ U_3 \\ \cdot \\ U_n \\ \cdot \\ U_N \end{Bmatrix} = \begin{Bmatrix} F_1 \\ 0 \\ F_3 \\ \cdot \\ F_n \\ \cdot \\ F_N \end{Bmatrix} \tag{2.13}$$

то есть соответствующие строка и столбец задаются нулевыми, а диагональный элемент – единичным. Соответственно, приравнивается нулю и F_2 [6].

Для нахождения перемещения узлов необходимо воспользоваться формулой (2.14):

$$[K] * \Delta = F \tag{2.14}$$

где [K] – глобальная матица жесткости, а Δ – перемещение всех узлов.

2.5 Решение СЛАУ методом Гаусса

Для решения системы линейных алгебраических уравнений будет использоваться метод Гаусса.

Процесс решения по методу Гаусса состоит из двух этапов: прямой и обратный ходы.

1. *Прямой ход*. На первом этапе осуществляется так называемый прямой ход, когда путём элементарных преобразований над строками систему приводят к ступенчатой или треугольной форме. А именно, среди элементов первого столбца матрицы выбирают ненулевой, перемещают его на крайнее верхнее положение перестановкой строк и вычитают получившуюся после перестановки первую строку из остальных строк, умножив её на величину, равную отношению первого элемента каждой из этих строк к первому элементу первой строки, обнуляя тем самым столбец под ним.

2. *Обратный ход*. Обратный ход: осуществляется определение значений неизвестных. Из последнего уравнения преобразованной системы вычисляется значение переменной x_n, после этого из предпоследнего уравнения становится возможным определение переменной x_{n-1} и так далее.

Преимущества метода:

– менее трудоёмкий по сравнению с другими методами;

– позволяет однозначно установить, совместна система или нет, и если совместна, найти её решение;

– позволяет найти максимальное число линейно независимых уравнений – ранг матрицы системы.

Существенным недостатком этого метода является невозможность сформулировать условия совместности и определенности системы в зависимости от значений коэффициентов и свободных членов. С другой стороны, даже в случае определенной системы этот метод не позволяет найти общие формулы, выражающие решение системы через ее коэффициенты и свободные члены, которые необходимо иметь при теоретических исследованиях.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

3.1 Реализация задачи на языке программирования *C#*

Для реализации поставленной задачи был использован язык программирования *C#*. Задача была разделена на следующие модули:

– *MainWindow;*
– *Gauss;*
– *Solver;*
– *MatrixAction;*
– *Element*
– *LocalRigityMatrix*
– *Nod*

Для работы с вышеперечисленными модулями разработан понятный и простой в использовании интерфейс пользователя. Структура классов приведена ниже в таблицах 3.1 – 3.16. Код данных классов представлен в приложении А.

В таблице 3.1представлена структура класса *MainWindow*. В данном классе содержатся вся логика отображения детали, парсинг файлов исходных данных, логика интерфейса пользователя.

Таблица 3.1 – Структура класса MainWindow

Имя переменной	Вид элемента	Тип	Специфи катор	Комментарий
1	2	3	4	5
load	Поле	Double	Private	Распределенна я нагрузка
puasson	Поле	Double	Private	Коэффициент Пуассона
thikness	Поле	Double	Private	Толщина
allowebleTension	Поле	Double	Private	!!!
nodesFormats	Поле	String[]	Private	Содержимое файла узлов
elementsFormats	Поле	String[]	Private	Содержимое файла элементов
constraintsFormat	Поле	String[]	Private	Содержимое файла закреплений

Продолжение таблицы 3.1

1	2	3	4	5
loadsFormat	Поле	String[]	Private	Содержимое файла нагрузок
elements	Поле	List<Element>	Private	Конечные элементы модели
nods	Поле	List<Element>	Private	Узлы конечно элементной сетки модели
solver	Поле	Solver	Private	Решатель
OpenNodeFile	Метод	-	Private	Открытие и загрузка файла узлов
OpenElementsFile	Метод	-	Private	Открытие и загрузка файла элементов
OpenLoadsFile	Метод	-	Private	Открытие и загрузка файла нагрузок
OpenConstraintsFile	Метод	-	Private	Открытие и загрузка файла закреплений
Calculate	Метод	-	Private	Выполнение расчёта напряженного состояния, определение количества витков резьбы
DrawFree	Метод	-	Private	Отрисовка ненагруженной детали
DrawLoaded	Метод	-	Private	Отрисовка нагруженной детали
parse	Метод	-	Private	Разбор файлов

В таблице 3.2 представлена структура класса Gauss. В данном классе выполняется решение СЛАУ.

Таблица 3.2 – Структура класса Gauss

Имя переменной	Вид элемента	Тип	Спецификатор	Комментарий
1	2	3	4	5
initial_a_matrix	Поле	Double[,]	Private	Инициализированная матрица
a_matrix	Поле	Double[,]	Private	Матрица А
x_vector	Поле	Double[]	Private	Вектор неизвестных х
initial_b_vector	Поле	Double[]	Private	Инициализированная матрица В
b_vector	Поле	Double[]	Private	Векор В
u_vector	Поле	Double[]	Private	Вектор невязки
eps	Поле	Double	Private	Порядок точности для сравнения вещественных чисел
size	Поле	Double	Private	Размерность задачи
Gauss	Конструктор	-	Public	Конструктор класса
InitIndex	Метод	-	Private	Инициализация массива индексов столбцов
FindR	Метод	-	Private	Поиск главного элемента в матрице
GaussSolve	Метод	-	Private	Нахождение решения СЛУ методом
GaussForwardStroke	Метод	-	Private	Прямой ход метода Гаусса
GaussBackwardStrok	Метод	-	Private	Обратный ход метода Гаусса

Продолжение таблицы 3.2

1	2	3	4	5
GaussDiscrepancy	Метод	-	Private	Вычисление невязки решения

В таблице 3.3представлена структура класса Solver. В данном классе выполняется решение поставленной задачи.

Таблица 3.3 – Структура класса Solver

Имя переменной	Вид элемента	Тип	Специфи катор	Комментарий
1	2	3	4	5
coliNodsCount	Поле	Integer	Private	Количество узлов в одном витке резьбы
coilElementsCount	Поле	Integer	Private	Количество элементов в одном витке
coilCount	Поле	Integer	Private	Количество витков
globalMatrix	Поле	Double[,]	Private	Глобальная матрица жесткости
matrixE	Поле	Double[,]	Private	Матрица E
forceVector	Поле	Double[,]	Private	Вектор сил
allowebleTension	Поле	Double[,]	Private	!!!
coilLength	Поле	Double	Private	Длина витка
Elements	Свойство	List<Element>	Public	Конечные элементы
Nods	Свойство	List<Nod>	Public	Узлы элементов
NodesDisplacement	Свойство	Double[]	Public	Вектор перемещений

Продолжение таблицы 3.3

1	2	3	4	5
RemoveOveredStringsAndColumInGlobal	Метод	-	Private	Удалить строки и столбцы
InsertLocalToGlobal	Метод	-	Private	Вставить локальную матрицу жесткости в глобальную
CalculateTensions	Метод	-	Private	Рассчитать перемещения и напряжения
GrowCoil	Метод	-	Private	Нарастить виток резьбы
Solve	Метод	-	Public	Рассчитать необходимое количество витков резьбы
Solver	Конструктор классса	-	Public	Конструктор класса

В таблице 3.4 представлена структура класса *MatrixAction*. Данный класс производит действия с матрицами.

Таблица 3.4– Структура класса *MatrixAction*

Имя переменной	Вид элемента	Тип	Спецификатор	Комментарий
1	2	3	4	5
MultipleMatVec	Метод	-	Public static	Умножение матрицы на вектор
MultipleMatMat	Метод	-	Public static	Умножение матрицы на матрицу
MultipleMatConst	Метод	-	Public static	Умножение матрицы на константу
TransponMat	Метод	-	Public static	Транспонирование матрицы

В таблице 3.5 представлена структура класса *Element*. В данном классе содержатся данные об конечном элементе модели.

Таблица 3.5– Структура класса *Element*

Имя переменной	Вид элемента	Тип	Специфи катор	Комментарий
1	2	3	4	5
Nod1	Свойство	Nod	Public	Первый узел элемента
Nod2	Свойство	Nod	Public	Второй узел элемента
Nod3	Свойство	Nod	Public	Третий узел элемента
Tension	Свойство	Double	Public	Напряжение в элементе
Element	Конструк тор	-	Public	Конструктор класса
Squre	Свойство	Double	Public	Площадь элемента

В таблице 3.6 представлена структура класса *Nod*. В данном классе содержатся данные об узлах конечных элементов.

Таблица 3.6– Структура класса *Nod*

Имя переменной	Вид элемента	Тип	Специфи катор	Комментарий
1	2	3	4	5
Number	Свойство	Integer	Public	Номер узла
X	Свойство	Double	Public	Положение по X
Y	Свойство	Double	Public	Положение по Y
XLoad	Свойство	Double	Public	Нагрузка по X
YLoad	Свойство	Double	Public	Нагрузка по Y
XDisplacement	Свойство	Double	Public	Перемещение по X
YDisplacement	Свойство	Double	Public	Перемещение по Y
IsConstraedByX	Свойство	Bool	Public	Закреплен ли узел по X
IsConstraedByY	Свойство	Bool	Public	Закреплен ли узел по Y

Продолжение таблицы 3.6

1	2	3	4	5
Nod	Конструк тор	-	Public	Конструктор
GetDistanceTo	Метод	-	Public	Определить расстояние до другого узла

В таблице 3.7 представлена структура класса *LocalRigidityMatrix*. В данном классе содержатся данные об локальные матрицы жесткости.

Таблица 3.7 – Структура класса *LocalRigidityMatrix*

Имя переменной	Вид элемента	Тип	Специфи катор	Комментарий
1	2	3	4	5
matrix	Поле	Double[,]	Public	Непосредствен но матрица
location	Поле	Integer[]	Public	Вектор размещения локальной матрицы в глобальной

Для реализации математической модели был выбран язык программирования C#. Листинг программы приведен в приложении Б.

Для реализации конечно-элементной сетки в программе были разработаны классы *Node* и *Element,* которые хранят характеристики узлов и элементов соответственно, а также классы *MainWindow* – создает список узлов и список элементов и *Solver* – строит глобальную матрицу и решает СЛАУ [6].

Для решения СЛАУ был реализован метод Гаусса [7].

В программе реализованы следующие варианты вывода результатов расчета:

1. Графический вывод напряжения детали рисунок Д.3.
2. Графический вывод суммы перемещений в узлах по осям *OX* и *OY*, рисунок Д.6.

3.2 Реализация модели конструкции в пакете ANSYS

На первом шаге моделирования строиться исследуемая деталь:

1. Задаются ключевые точки:

Main Menu → Preprocessor → Modeling → Keypoints → In Active CS...

1 (0;0), **2** (0;0.5), **3** (0.9;0.5), **4** (0.9;0)

2. Соединяются полученные точки линиями, рисунок 3.1:

Preprocessor → Modeling → Create → Lines → Lines → Straight Line...

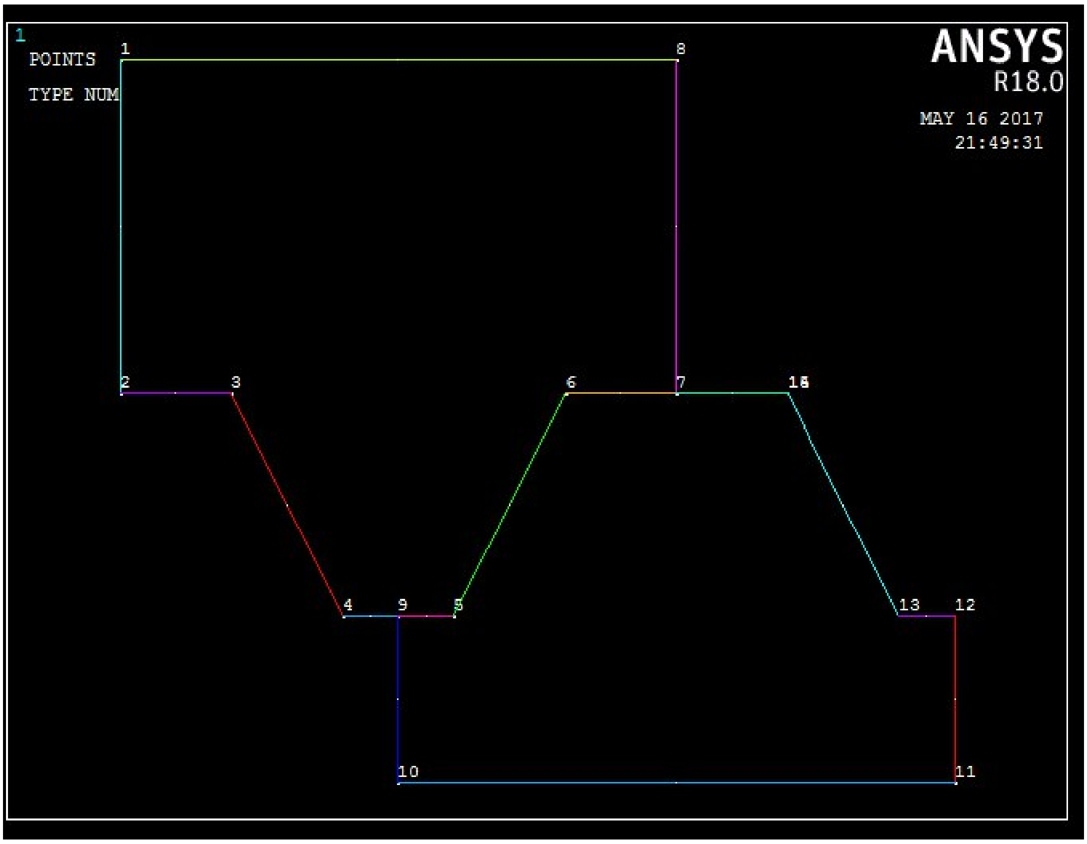

Рисунок 3.1 – Соединение линиями

3. Создаетсяплоскость:

Preprocessor → Modeling → Create → Areas → Arbitrary → By Lines

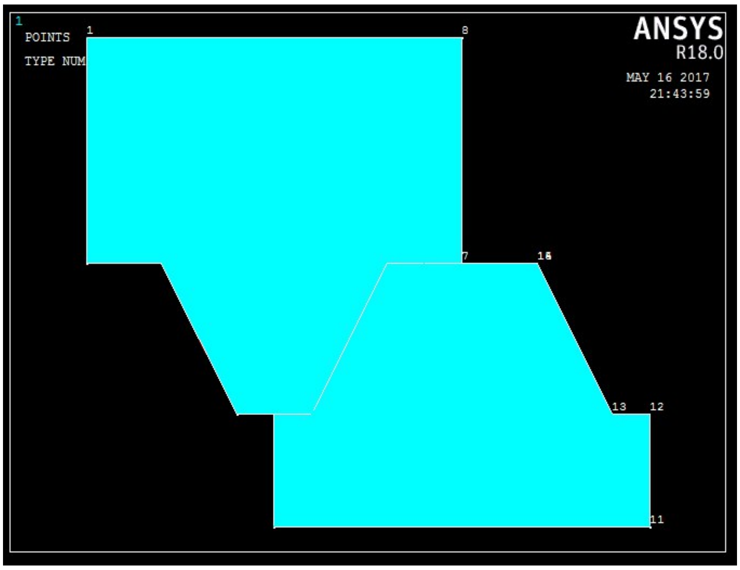

Рисунок 3.2 – Поверхность детали

4. Выбирается тип конечного элемента, рисунок 3.3:
Preprocessor → Element Type → Add/Edit/Delete... → Add...

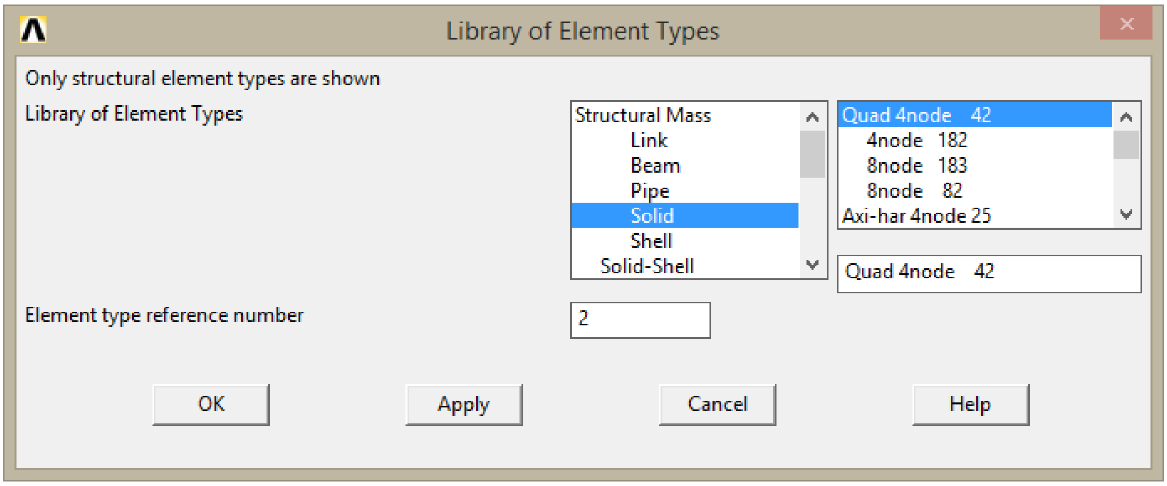

Рисунок 3.3 – Выбор типа конечного элемента

5. Задаются характеристики материала,рисунок 3.4:
Preprocessor → Material props → Materials Modes… → Structural → Linear → Elastic → Isotropic.

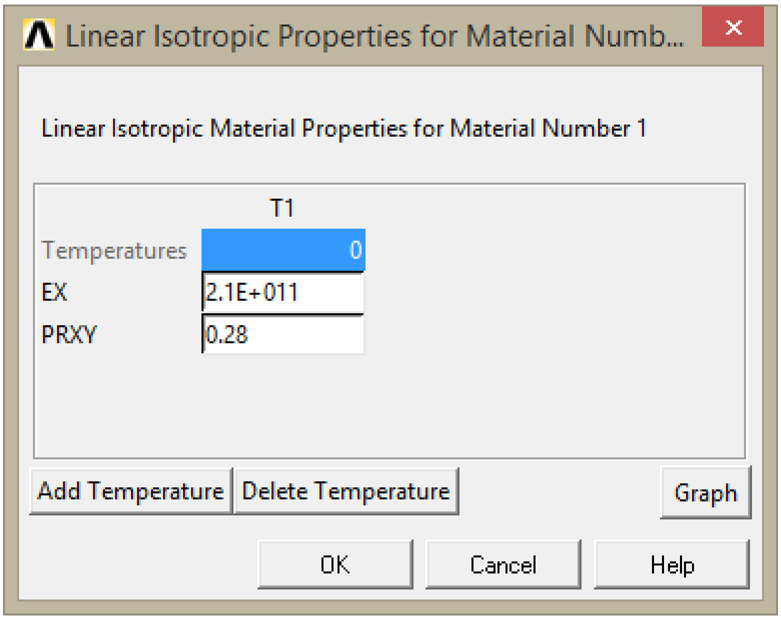

Рисунок 3.4 – Ввод модуля упругости и коэффициента Пуассона

Построение конечно-элементной сетки с треугольными элементами показано на рисунке 3.5:
Preprocessor → Meshing → MeshTool

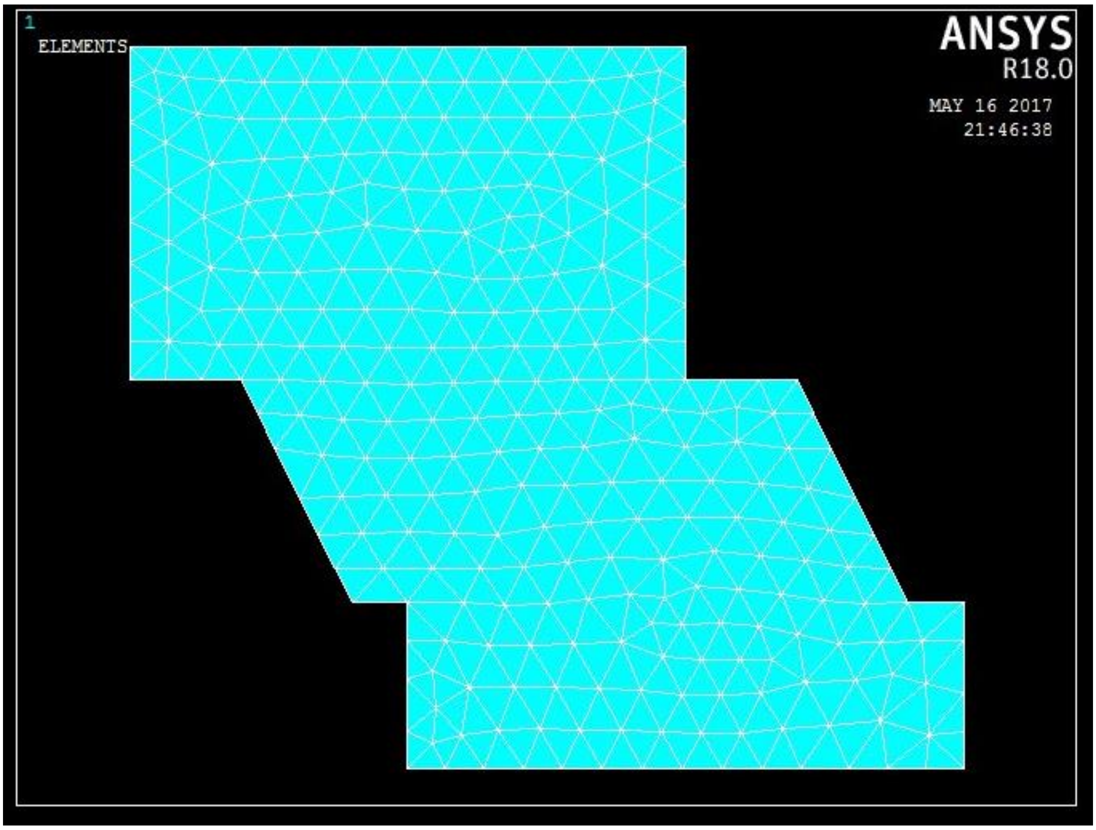

Рисунок 3.5 – Конечно-элементная сетка

6. Закрепляется и нагружается деталь в соответствии с заданием, рисунок 3.6:

Sulution → DefineLoads → Apply → Structural → Displacement → On Lines

Sulution → Define Loads → Apply → Structural → Force/Moment → On Nodes

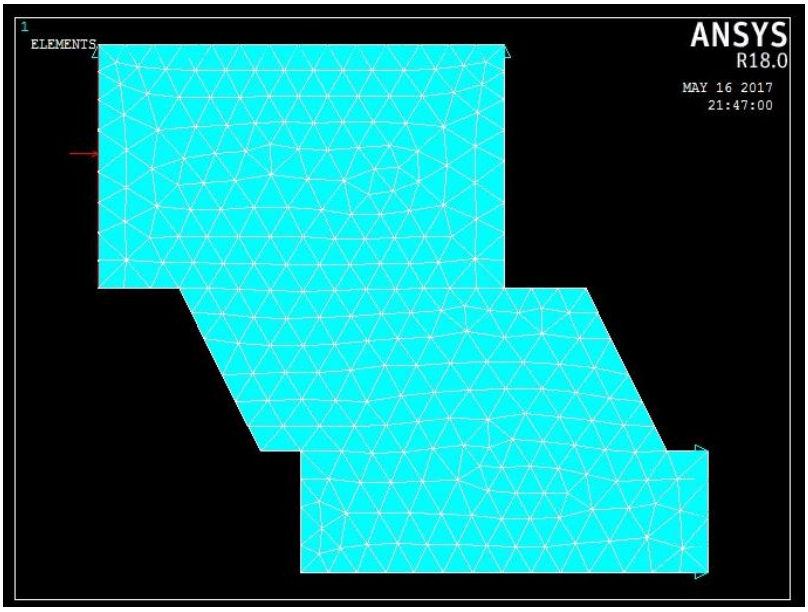

Рисунок 3.6–Нагрузкаизакреплениедетали

7. Расчитываетсядеталь:

Sulution → Solve → CurrentLS

8. Вывод результатов:

General Postproc → Plot Results → Contour Plot → Element Solu → Stress → von Mises stres[5].

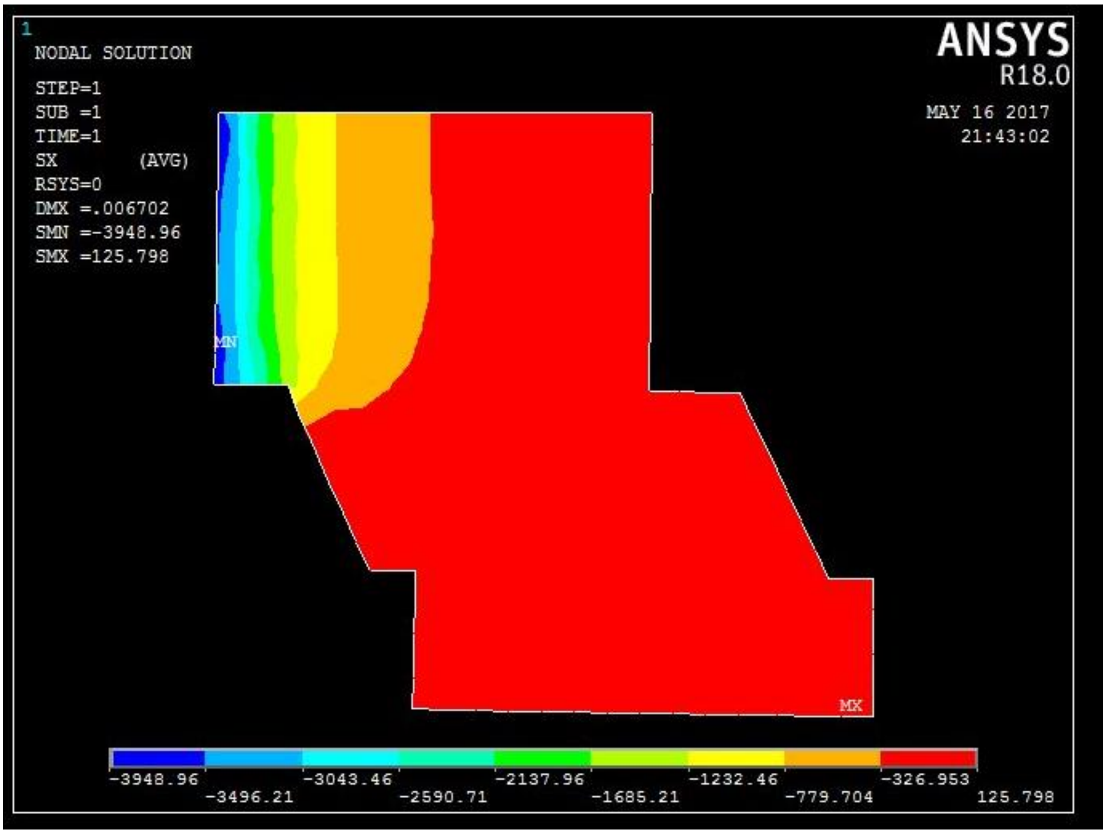

Рисунок 3.7 – Графическая интерпретация полученных результатов в программе ANSYS

3.3 Исследование полученных результатов

В результате проделанной работы было разработано приложение, которой исследует заданную деталь, разбивает деталь на конечные треугольные элементы и находит перемещения в узлах при заданных нагрузках и закреплениях.

Для проверки результатов, поставленная задача была реализована в пакете *ANSYS*. Так как в программе реализована ручная сетка разбиения, которая не совпадает с сеткой разбиения *ANSYS*, то для сравнения были выбраны элементы, расположение которых приблизительно одинаково. Для этого сравнить результаты перемещений узлов сетки в разработанном приложении и в пакете *ANSYS*. Результаты сравнения приведены в таблице 3.3.

Таблица 3.3 – Результаты расчетов

№ п/п	Перемещения по оси *X* в *ANSYS*, м	Перемещения в приложении по оси *X*, м	Погрешность, %
1	-0,37516E+09	-3,54703283E+8	5,45
2	-0,42202E+09	-3,68387021E+8	12,7
3	-0,38123E+09	-3,40316134E+8	10,7
4	-0,33880E+09	-3,31705421E+8	2,09
5	-0,38697E+09	-3,74556442E+8	3,2
6	-0,40135E+09	-3,84645E+9	4,16
7	0,28783E+07	25537862,0	11,27
8	0,89702E+08	83547118,0	6,86
9	-0,41345E+09	-395040672,0	4,45
10	-0,35694E+09	-3,40316134E+8	4,65
11	-0,66589E+08	-6,867663E+7	3,13
12	-0,79944E+08	-756239648,0	5,4
13	-0,13866E+09	-1,20041843E+8	13,4
14	0,17901E+09	1,53400115E+8	14,3
15	0,28800E+09	252178992,0	12,4
16	-0,11838E+09	-1,22202278E+8	3,22
17	-0,66478E+08	-752165248,0	13,14
18	-0,81506E+08	-9,22031E+7	13,12
19	-0,24556E+09	-235471184,0	4,1
20	0,11014E+09	1,13386086E+8	2,94

Проанализировав результаты видно, что результаты близки, но есть и небольшие расхождения. Эти погрешности находятся в пределах 7.5 процентов. Это связано с тем, что элементы разработанного приложения не соответствует элементам *ANSYS*. В программе можно изменить густоту сетки, как показано в приложении В.5. Для сравнения на рисунках 3.7 и 3.8 приведены результаты работы пакета *ANSYS* и разработанного приложения.

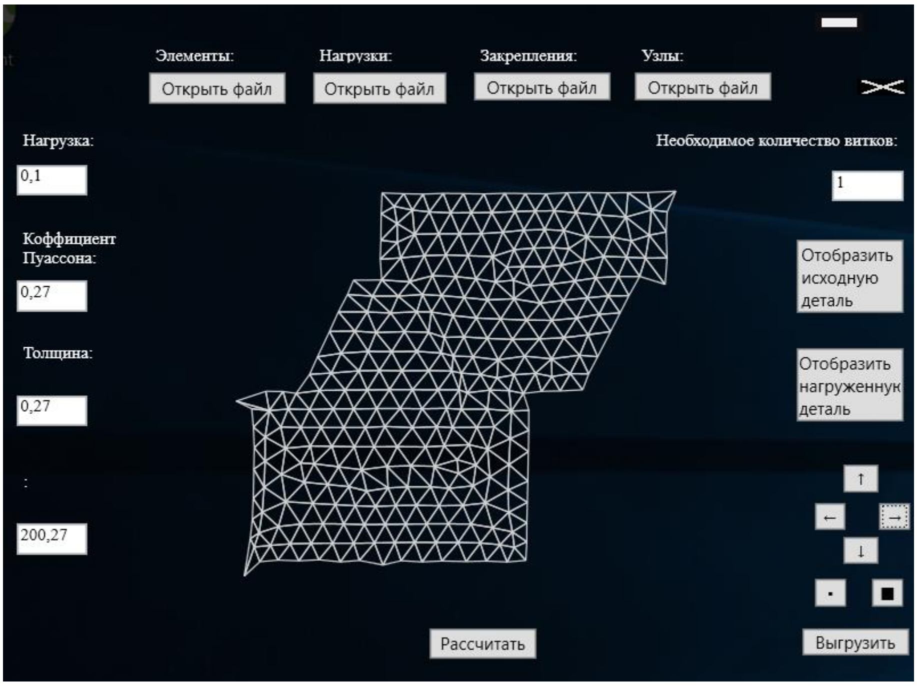

Рисунок 3.8 – Решенная задача в разработанном приложении

3.4 Решение поставленной задачи

Максимально допустимое напряжения для стали равно 50 МПа. Чтобы определить оптимальные размеры детали, при которых она не разрушиться под нагрузкой 1 КПа, необходимо произвести единичный расчёт и в левом верхнем углу в текстовом поле «Необходимое количество витков» будет отображено необходимое количество витков для данной толщины, материала и нагрузки. Пример результатов вычислений приведен на рисунке 3.10.

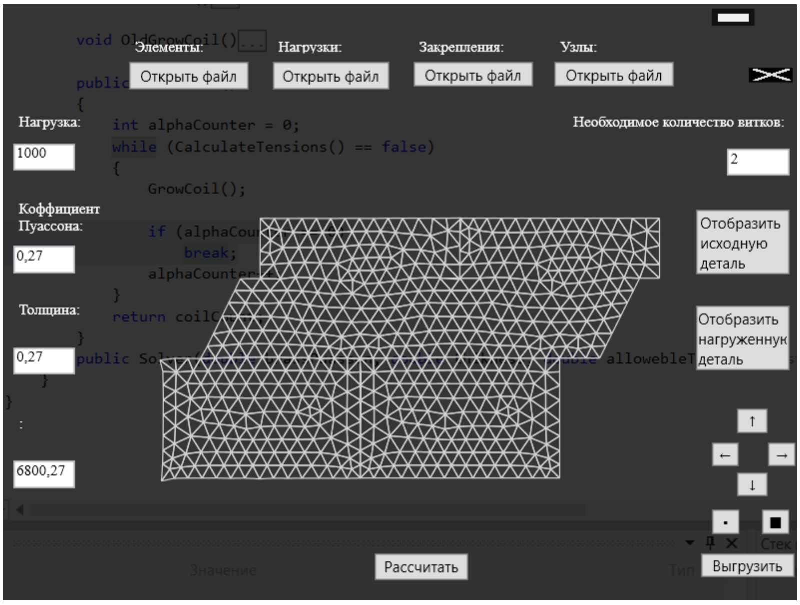

Рисунок 3.10 – Определенное количество витков

ЗАКЛЮЧЕНИЕ

Для определения оптимального количества витков резьбы был разработан программный комплекс, моделирующий напряжённо-деформированное состояние плоской конструкции с нагрузкой в заданных точках. Приложение может выполнять расчет максимального напряжения для конструкции с заданными параметрами и отображает результаты работы в графическом виде. Для проведения более точных расчетов густоту сетки разбиения можно изменить, что, конечно же, отрицательно повлияет на производительность приложения. Преимуществом данного приложение является:

– низкая стоимость;
– отсутствие ошибок при выполнении расчетов;
– простой и интуитивно понятный интерфейс;

Единственный недостаток приложения – это узкая специализация. Расчет можно проводить над резьбовым соединением только данного типа.

Приложение позволяет довольно быстро рассчитывать размеры деталей соединяемых с помощью заклепок.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бахвалов, Н.С. Численные методы: Учеб. пособие / Н.С.Бахвалов, 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: 10.04.2017.
3. Метод конечных элементов [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Метод_конечных_элементов. – Дата доступа: 10.04.2017.
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. Зенкевич, О. Метод конечных элементов в технике /О. Зенкевич. – М.: Мир, 1984. – 428 с.

ПРИЛОЖЕНИЕ А
(Обязательное)
Листинг класса “Node”

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ScrewBondCalculator.DataTypes
{
    class Nod
    {
        int number;
        double xLoad, yLoad, x, y, xDisplacement, yDisplacement;
        bool isConstraedByX, isConstraedByY;

        public int Number
        {
            get { return number; }
            set { number = value; }
        }
        public double X
        {
            get { return x; }
            set { x = value; }
        }
        public double Y
        {
            get { return y; }
            set { y = value; }
        }
        public double XLoad
        {
            get { return xLoad; }
            set { xLoad = value; }
        }
        public double YLoad
        {
            get { return yLoad; }
            set { yLoad = value; }
        }
        public double XDisplacement
        {
            get { return xDisplacement; }
            set { xDisplacement = value; }
        }
        public double YDisplacement
        {
            get { return yDisplacement; }
            set { yDisplacement = value; }
        }
        public bool IsConstraedByX
        {
            get { return isConstraedByX; }
            set { isConstraedByX = value; }
        }
        public bool IsConstraedByY
        {
            get { return isConstraedByY; }
            set { isConstraedByY = value; }
        }
```

```
        public Nod(int number, double x, double y, double xLoad, double yLoad)
        {
            this.number = number;
            this.x = x;
            this.y = y;
            this.xLoad = xLoad;
            this.yLoad = yLoad;
            isConstraedByX = false;
            isConstraedByY = false;
        }


        public double GetDistanceTo(Nod nod)
        {
            return Math.Sqrt(Math.Pow(this.x - nod.X, 2) + Math.Pow(this.y - nod.Y, 2));
        }

        public override string ToString()
        {
            return ("Номер узла: " + number + " X координата: " + x + " Y координата: " + y + " нагрузка по X: " + xLoad
+ " нагрузка по Y: " + yLoad + " перемещение по X: " + XDisplacement + " перемещение по Y: " + YDisplacement);
        }
    }
}
```

ПРИЛОЖЕНИЕБ
(Обязательное)
Листинг класса “Element”

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ScrewBondCalculator.DataTypes
{
    class Element
    {
        public Nod Nod1
        {
            get;
            set;
        }
        public Nod Nod2
        {
            get;
            set;
        }
        public Nod Nod3
        {
            get;
            set;
        }

        public double Tension
        {
            get;
            set;
        }


        public double Squre
        {
            get { return (1 * Nod2.X * Nod3.Y + Nod1.X * 1 * Nod2.Y + Nod1.Y * 1 * Nod3.X  - 1 * Nod2.Y * Nod3.X -
Nod1.X * 1 * Nod3.Y - Nod1.Y * Nod2.X * 1) / 2; }
        }

        public Element(Nod nod1, Nod nod2, Nod nod3)
        {
            this.Nod1 = nod1;
            this.Nod2 = nod2;
            this.Nod3 = nod3;
            Tension = 0;
        }
    }
}
```

ПРИЛОЖЕНИЕ В

(Обязательное)

Листинг класса “LocalRigidityMatrix”

```
namespace ScrewBondCalculator.DataTypes
{
    class LocalRigidityMatrix
    {
        public double[,] matrix;
        public int[] location;
        public LocalRigidityMatrix(double[,] matrix, int first, int second, int third)
        {
            first = first * 2;
            second = second * 2;
            third = third * 2;
            this.matrix = matrix;
            location = new int[6] { first - 1, first, second - 1, second, third - 1, third };
        }
    }
}
```

ПРИЛОЖЕНИЕ Г
(Обязательное)
Листинг класса “Solver”

```
namespace ScrewBondCalculator.Calculators
{
    class Solver
    {
        int coliNodsCount, coilElementsCount;
        int coilCount;
        double[,] globalMatrix;
        double[,] matrixE;
        double[] forceVector;
        double allowebleTension;
        double coilLength;

        public double Thikness
        {
            get;
            set;
        }
        public double Puasson
        {
            get;
            set;
        }

        public double[,] GlobalMatrix
        {
            get { return globalMatrix; }
            set { globalMatrix = value; }
        }
        public double[] ForceVector
        {
            get { return forceVector; }
            set { forceVector = value; }
        }
        public List<Element> Elements
        {
            get;
            set;
        }
        public List<Nod> Nods
        {
            get;
            set;
        }
        public double[] NodesDisplacement
        {
            get;
            private set;
        }

        void RemoveOveredStringsAndColumInGlobal(int la)
        {
            double[,] newGlobalMatrix = new double[globalMatrix.GetLength(0) - 1, globalMatrix.GetLength(1) - 1];
            double[] newForceVector = new double[forceVector.Length - 1];
            int inn = 0;
            int jnn = 0;

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

                if (i != la)
```

```
            {
                for (int j = 0; j < globalMatrix.GetLength(1); j++)
                {


                    if (j != la)
                    {
                        newGlobalMatrix[inn, jnn] = globalMatrix[i, j];
                        jnn++;
                    }
                }
                newForceVector[inn] = forceVector[i];

                inn++;
            }
            jnn = 0;
        }

        forceVector = new double[newForceVector.Length];
        forceVector = newForceVector;

        globalMatrix = new double[newGlobalMatrix.GetLength(0), newGlobalMatrix.GetLength(1)];
        globalMatrix = newGlobalMatrix;
    }
    void InsertLocalToGlobal(LocalRigidityMatrix local)
    {
        for (int i = 0; i < local.matrix.GetLength(0); i++)
            for (int j = 0; j < local.matrix.GetLength(1); j++)
                globalMatrix[local.location[i] - 1, local.location[j] - 1] += local.matrix[i, j];
    }
    bool CalculateTensions()
    {
        globalMatrix = new double[Nods.Count * 2, Nods.Count * 2];
        NodesDisplacement = new double[Nods.Count * 2];
        ForceVector = new double[Nods.Count * 2];
        for (int i = 0; i < Nods.Count; i++)
        {
            ForceVector[(Nods[i].Number * 2 - 2)] = Nods[i].XLoad;
            ForceVector[(Nods[i].Number * 2 - 1)] = Nods[i].YLoad;
        }
        LocalRigidityMatrix lrm;
        foreach (Element localElement in Elements)
        {

            double[,] matrixB = {
                                { localElement.Nod2.Y - localElement.Nod3.Y, 0,localElement.Nod3.Y - 
localElement.Nod1.Y,0 ,localElement.Nod1.Y - localElement.Nod2.Y ,0 },
                                { 0, localElement.Nod3.X - localElement.Nod2.X,0 ,localElement.Nod1.X - 
localElement.Nod3.X ,0 ,localElement.Nod2.X - localElement.Nod1.X },
                                { localElement.Nod3.X - localElement.Nod2.X,localElement.Nod2.Y - localElement.Nod3.Y 
,localElement.Nod1.X - localElement.Nod3.X ,localElement.Nod3.Y - localElement.Nod1.Y ,localElement.Nod2.X - 
localElement.Nod1.X ,localElement.Nod1.Y - localElement.Nod2.Y }
                              };
            matrixB = MatrixAction.MultipleMatConst(matrixB, 1/(2*localElement.Squre));
            lrm = new 
LocalRigidityMatrix(MatrixAction.MultipleMatConst((MatrixAction.MultipleMatMat(MatrixAction.MultipleMatMat(Ma
trixAction.TransponMat(matrixB), matrixE), matrixB)), (Thikness * localElement.Squre)), localElement.Nod1.Number, 
localElement.Nod2.Number, localElement.Nod3.Number);
            InsertLocalToGlobal(lrm);
        }
        {
            foreach(Nod localNod in Nods)
            {
                if (localNod.IsConstraedByX)
                {
```

```
                    RemoveOveredStringsAndColumInGlobal((localNod.Number * 2 - 2));
                }

                if (localNod.IsConstraedByY)
                {
                    RemoveOveredStringsAndColumInGlobal((localNod.Number * 2 - 1));
                }
            }
        }
        NodesDisplacement = new Gauss(globalMatrix, ForceVector).XVector;
        {
            Nod localNod;
            int curNodNumber;
            for (int i = 1; i < NodesDisplacement.Length; i += 2)
            {
                curNodNumber = i/2 + 1;
                localNod = Nods.Where(n => n.Number == curNodNumber).ToArray()[0];
                localNod.XDisplacement = NodesDisplacement[i - 1];
                localNod.YDisplacement = NodesDisplacement[i];
            }
        }
        {
            double[] u = new double[6];
            double[] tenComp;
            foreach (Element localElement in Elements)
            {
                double[,] matrixB = {
                                    { localElement.Nod2.Y - localElement.Nod3.Y, 0,localElement.Nod3.Y -
localElement.Nod1.Y,0 ,localElement.Nod1.Y - localElement.Nod2.Y ,0 },
                                    { 0, localElement.Nod3.X - localElement.Nod2.X,0 ,localElement.Nod1.X -
localElement.Nod3.X ,0 ,localElement.Nod2.X - localElement.Nod1.X },
                                    { localElement.Nod3.X - localElement.Nod2.X, localElement.Nod2.Y - localElement.Nod3.Y
,localElement.Nod1.X - localElement.Nod3.X ,localElement.Nod3.Y - localElement.Nod1.Y ,localElement.Nod2.X -
localElement.Nod1.X ,localElement.Nod1.Y - localElement.Nod2.Y }
                                  };
                matrixB = MatrixAction.MultipleMatConst(matrixB, 1 / (2 * localElement.Squre));
                u[0] = localElement.Nod1.XDisplacement;
                u[1] = localElement.Nod1.YDisplacement;
                u[2] = localElement.Nod2.XDisplacement;
                u[3] = localElement.Nod2.YDisplacement;
                u[4] = localElement.Nod3.XDisplacement;
                u[5] = localElement.Nod3.YDisplacement;
                tenComp = MatrixAction.MultipleMatVec(MatrixAction.MultipleMatMat(matrixE, matrixB), u);
                localElement.Tension = (1.0 / Math.Sqrt(2)) * Math.Sqrt(Math.Pow(tenComp[0] - tenComp[1], 2) +
Math.Pow(tenComp[0], 2) + Math.Pow(tenComp[1], 2) + 6 * tenComp[2]);
                if (localElement.Tension > allowebleTension)
                {
                    return true;
                }
            }
        }
        return true;
    }
    void GrowCoil()
    {
        Nod newCoilNod, newElementNod1, newElementNod2, newElementNod3;
        int oldNodsCount = Nods.Count, oldElementsCount = Elements.Count;
        for (int i = coliNodsCount * (coilCount - 1); i < oldNodsCount; i++)
        {
            newCoilNod = new Nod(oldNodsCount + i + 1, Nods[i].X + coilLength, Nods[i].Y, Nods[i].XLoad,
Nods[i].YLoad);
            newCoilNod.IsConstraedByY = Nods[i].IsConstraedByY;
            Nods[i].XLoad = 0;
            Nods[i].YLoad = 0;
            Nods.Add(newCoilNod);
```

```
            }

            for (int i = coilElementsCount * (coilCount - 1); i < oldElementsCount; i++)
            {
                newElementNod1 = Nods.Where(n => n.Number == coliNodsCount +
Elements[i].Nod1.Number).ToArray()[0];
                newElementNod2 = Nods.Where(n => n.Number == coliNodsCount +
Elements[i].Nod2.Number).ToArray()[0];
                newElementNod3 = Nods.Where(n => n.Number == coliNodsCount +
Elements[i].Nod3.Number).ToArray()[0];
                Elements.Add(new Element(newElementNod1, newElementNod2, newElementNod3));
            }
            coilCount++;
        }

        public int Solve()
        {
            while (CalculateTensions() == false)
            {
                GrowCoil();
            }
            return coilCount;
        }
        public Solver(double brassPuasson, double thikness, double allowebleTension, List<Element> elements, List<Nod>
nods)
        {
            double startCoilPoint = nods[0].X, finalCoilPoint = nods[0].X;

            Thikness = thikness;
            Puasson = brassPuasson;
            matrixE = new double[,] { { 1, Puasson, 0 }, { Puasson, 1, 0 }, { 0, 0, (1 - Puasson) / 2 } };
            Elements = elements;
            Nods = nods;
            this.allowebleTension = allowebleTension;

            foreach (Nod nod in nods)
            {
                if (nod.X > finalCoilPoint)
                {
                    finalCoilPoint = nod.X;
                }
                if (nod.X < startCoilPoint)
                {
                    startCoilPoint = nod.X;
                }
            }
            coilLength = startCoilPoint - finalCoilPoint;
            coliNodsCount = Nods.Count;
            coilElementsCount = Elements.Count;
            coilCount = 1;
        }
    }
}
```

ПРИЛОЖЕНИЕ Д
(обязательное)
Текст *APDL*-скрипта для пакета *ANSYS*

```
/PREP7
K, ,2,18,,
K, ,2,12,,
K, ,4,12,,
K, ,6,8,,
K, ,8,8,,
K, ,10,12,,
K, ,12,12,,
K, ,12,18,,
K, ,7,8,,
K, ,7,5,,
K, ,17,5,,
K, ,17,8,,
K, ,16,8,,
K, ,14,12,,
K, ,14,12,,
LSTR,    1,    2
LSTR,    2,    3
LSTR,    3,    4
LSTR,    4,    9
LSTR,    9,    5
LSTR,    5,    6
LSTR,    6,    7
LSTR,    7,    8
LSTR,    8,    1
LSTR,   10,    9
LSTR,    9,    5
LSTR,    5,    6
LSTR,    6,    7
LSTR,    7,   14
LSTR,   14,   13
LSTR,   13,   12
LSTR,   12,   11
LSTR,   11,   10
FLST,2,9,4
FITEM,2,1
FITEM,2,2
FITEM,2,4
FITEM,2,3
FITEM,2,5
FITEM,2,6
FITEM,2,7
FITEM,2,8
FITEM,2,9
AL,P51X
FLST,2,8,4
FITEM,2,10
FITEM,2,5
FITEM,2,6
FITEM,2,11
FITEM,2,12
FITEM,2,14
FITEM,2,13
FITEM,2,15
AL,P51X
FLST,2,9,4
FITEM,2,10
```

```
FITEM,2,5
FITEM,2,6
FITEM,2,7
FITEM,2,11
FITEM,2,12
FITEM,2,13
FITEM,2,14
FITEM,2,15
AL,P51X
!*
ET,1,PLANE182
!*
KEYOPT,1,1,0
KEYOPT,1,3,1
KEYOPT,1,6,0
KEYOPT,1,10,0
!*
!*
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,2e6
MPDATA,PRXY,1,,0.3
SMRT,6
TYPE,   1
MAT,       1
REAL,
ESYS,       0
SECNUM,
!*
TYPE,   1
MAT,       1
REAL,
ESYS,       0
SECNUM,
!*
MSHAPE,1,2D
MSHKEY,0
!*
FLST,5,2,5,ORDE,2
FITEM,5,1
FITEM,5,-2
CM,_Y,AREA
ASEL, , , ,P51X
CM,_Y1,AREA
CHKMSH,'AREA'
CMSEL,S,_Y
!*
AMESH,_Y1
!*
CMDELE,_Y
CMDELE,_Y1
CMDELE,_Y2
!*
SMRT,3
TYPE,   1
MAT,       1
REAL,
ESYS,       0
SECNUM,
!*
FLST,5,2,5,ORDE,2
FITEM,5,1
```

```
FITEM,5,-2
CM,_Y,AREA
ASEL, , , ,P51X
CM,_Y1,AREA
CHKMSH,'AREA'
CMSEL,S,_Y
!*
!*
ACLEAR,_Y1
AMESH,_Y1
!*
CMDELE,_Y
CMDELE,_Y1
CMDELE,_Y2
FLST,5,2,5,ORDE,2
FITEM,5,1
FITEM,5,-2
CM,_Y,AREA
ASEL, , , ,P51X
CM,_Y1,AREA
CHKMSH,'AREA'
CMSEL,S,_Y
!*
!*
ACLEAR,_Y1
AMESH,_Y1
!*
CMDELE,_Y
CMDELE,_Y1
CMDELE,_Y2
!*
FLST,2,1,4,ORDE,1
FITEM,2,15
/GO
FLST,2,11,1,ORDE,4
FITEM,2,10
FITEM,2,14
FITEM,2,53
FITEM,2,-61
/GO
FLST,2,1,4,ORDE,1
FITEM,2,14
!*
/GO
DL,P51X, ,UX,0
/UI,MESH,OFF
FLST,2,1,4,ORDE,1
FITEM,2,14
!*
/GO
DL,P51X, ,UX,0
FINISH
/POST1
FINISH
/SOL
FINISH
/PREP7
FLST,2,1,4,ORDE,1
FITEM,2,9
!*
/GO
DL,P51X, ,UY,0
```

```
FLST,2,1,4,ORDE,1
FITEM,2,14
!*
/GO
DL,P51X, ,UX,0
FLST,2,1,4,ORDE,1
FITEM,2,1
/GO
!*
SFL,P51X,PRES,5000,
FINISH
/SOL
/STATUS,SOLU
SOLVE
FINISH
/POST1
!*
/EFACET,1
PLNSOL, S,X, 0,1.0
```

ПРИЛОЖЕНИЕ Е

(обязательное)

Чертеж детали с простановкой основных размеров