**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 технологии (в проектировании и производстве)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине «Оптимизация проектных решений»

на тему: **«АНАЛИЗ РАСПРЕДЕЛЕНИЯ ТЕМПЕРАТУР В СТЕРЖНЕ С БОКОВЫМ ТЕПЛООБМЕНОМ И ИСТОЧНИКОМ ТЕПЛОТЫ»**

Исполнитель: студент гр. ИТП-41

Ястребов А.А.

Руководитель: профессор Мурашко И.А.

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии

по защите курсового проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2020

**ОГЛАВЛЕНИЕ**

Введение4

1 Обзор численных методов моделирования в тепломассообмене5

1.1 Существующие численные методы 5

1.2 Метод конечных разностей 6

1.3 Метод конечных элементов 7

2 Алгоритмический анализ задачи 9

2.1 Постановка задачи 9

2.2 Метод теплового баланса для решения задачи вычисления распределения температуры в стержне 11

2.3 Применение метода теплового баланса для решения задачи распределения температуры в стержне 12

2.4 Метод Гаусса для решения системы линейных алгебраических уравнений 15

3 Программная реализация задачи 18

3.1 Структура разрабатываемого приложения 18

4 Верификация полученных результатов 21

4.1 Основы модульного тестирования 21

4.2 Тестирования пользовательского интерфейса 21

4.3 Сравнение полученных результатов 24

Заключение29

Список использованных источников30

Приложение А Листинг программы31

Приложение Б Чертёж стержня36

**ВВЕДЕНИЕ**

На протяжении многих лет люди находили и скапливали познания о находящемся вокруг мире, действиях и явлениях, протекающие непрерывно вокруг нас. При этом, люди упорядочивали всю информацию в согласовании с их схожестью, делая упор на приобретенные познания и эксперименты. И очень часто такая сортировка имела довольно большой практический смысл.

Такой процесс сортировки информации людьми на основе некоторых признаков получил название «классификация». С увеличением познаний о мире, роль классификации резко возросла. Р.Сокэл подчеркнул, что классификация или систематизация является интеллектуальной деятельностью высшего уровня, характеризующееся большим объемом научных достижений.

С ростом объёма и сложности получаемой информации чёткий и действенный её анализ стал практически неосуществим для человека. И для решения этой проблемы необходимы были новые методы обработки и сортировки информации.

Создание и развитие вычислительных машин положительно повлияло на решение проблемы классификации данных. В базу данного процесса легла идея о применении математических способов для сбора, сортировки и классификации информации.

В настоящее время, группу методов и алгоритмов, которые применяются для автоматической классификации полученных данных, называют кластерным анализом. Кластерный анализ дает возможность рассматривать довольно большой объем информации и классифицировать его, согласно предлагаемым условиям и целям исследования.

Кластерный анализ получил довольно обширное применение в различных сферах и отраслях науки таких как: биология, химия, математика, информатика, статистика, медицина и других. В любой научной деятельности классификация является одной из фундаментальных составляющих, без которой невозможны построение и проверка научных гипотез и теорий.

Целью данного курсового проекта является разработка программного комплекса для кластеризации числовой информации. Такая информация состоит из набора чисел, которые описывают конкретные характеристики объекта. Кластеризация информации такого типа может использоваться во многих сферах.

Для решения данной задачи необходимо произвести анализ существующих методов кластеризации данных с помощью нейронных сетей и разработать приложение, которое моделирует выбранную нейронную сеть, производит её обучение и выполняет кластеризацию данных.

Первый раздел содержит обзор литературы по существующим методам кластеризации данных с помощью нейронных сетей. Указываются подходы и методы решения рассматриваемой задачи.

Второй раздел посвящён алгоритмическому анализу задачи. Он содержит полную постановку задачи, перечень исходных данных, а также математическое описание решения рассматриваемой задачи.

Третий раздел содержит описание структуры программы, реализующей решение задачи.

В четвёртом разделе производится верификация полученных результатов. Описывается процесс тестирования программы. Производится кластеризация тестовых данных и выполняется оценка корректности работы разработанного алгоритма.

**1 ОБЗОР МЕТОДОВ КЛАСТЕРНОГО АНАЛИЗА**

* 1. **Понятие кластерного анализа**

Кластерный анализ – это совокупность математических методов, предназначенных для разбиения выборки объектов на непересекающиеся подмножества, называемые кластерами. Разбиение производится таким образом, чтобы каждый кластер состоял из множества схожих объектов, а объекты разных кластеров значительно различались. Задача кластеризации расценивается как распознавание образов без учителя.

Большое достоинство кластерного анализа в том, что он позволяет производить разбиение объектов не по одному критерию, а по целому множеству признаков. Кроме того, кластерный анализ в отличие от большинства математико-статистических методов не накладывает никаких ограничений на вид рассматриваемых объектов, и позволяет рассматривать множество исходных данных практически произвольной природы.

Однако, как и любой другой метод, кластерный анализ имеет определенные недостатки и ограничения. В частности, состав и количество кластеров зависит от выбираемых критериев разбиения. Также может возникнуть проблема неопределённости отнесения объекта к какому-либо кластеру.

При выполнении кластеризации важно подготовить исходные данные к анализу. Первоначальным этапов является выделение в исходной выборке наиболее важных характеристик для повышения скорости расчётов. Далее необходимо выбрать масштаб данных. Выбор масштаба играет большую роль. Как правило, данные нормализуют вычитанием среднего и делением на стандартное отклонение так, что дисперсия оказывается равной единице. Чаще всего значение признаков варьируются от нуля до единицы, либо от минус единицы до единицы.

**1.2 Задача кластерного анализа**

Задача кластерного анализа заключается в том, чтобы на основании данных, содержащихся во множестве *Х*, разбить множество объектов *G* на *m* кластеров (где *m –* целое число) *Q1, Q2, …, Qm,* так, чтобы каждый объект *Gj* принадлежал одному и только одному подмножеству разбиения и чтобы объекты, принадлежащие одному и тому же кластеру, были сходными, в то время, как объекты, принадлежащие разным кластерам были разнородными.

Решением задачи кластерного анализа являются разбиения, удовлетворяющие некоторому критерию оптимальности. Этот критерий может представлять собой некоторый функционал, выражающий уровни желательности различных разбиений и группировок, который называют целевой функцией.

Кластеризация данных выполняется на основе расстояния между кластерами и объектами. Данное понятие является интегральной мерой сходства объектов между собой. Два объекта относятся к одному и тому же кластеру, если расстояние между ними достаточно малое, и к разным кластерам, если расстояние между ними достаточно большое. Таким образом, отнесение объектов к определённым кластерам зависит от расстояния между объектом и кластером в евклидовом пространстве. Расстоянием между объектами в пространстве называется такая неотрицательная функция, которая является функцией расстояния, удовлетворяющая следующим условиям:

– d(Xi, Xj) >= 0 для всех X и Xj из Ep;

– d(Xi, Xj) = 0 тогда и только тогда, когда X = Xj;

– d(Xi, Xj) = d(Xj, X);

– d(Xi, Xj) <= d(Xi, Xk) + d(Xk, Xj), где Xj, Xi и Xk – любые вектора из Ep;

**1.3 Метод конечных элементов**

***1.3.1*** Метод конечных элементов – это численный метод решения физических задач. Суть метода заключается в дискретизации исследуемого объекта на конечное множество элементов и кусочно-элементной аппроксимации исследуемых функций. Исходная расчётная область дискретизируется конечными элементами, которые могут быть одномерными, двумерными и трёхмерными. Пример конечных элементов изображён на рисунке 1.2.

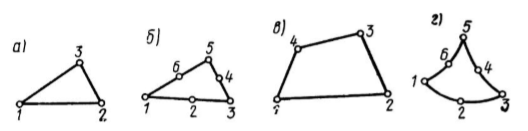


Рисунок 1.2 – Пример двумерных конечных элементов

Поведение функции на каждом конечном элементе выражается через базисные функции (функции формы). Данные функции берутся в виде кусочно-непрерывных функций, которые обращаются в нуль всюду, кроме ограниченных подобластей, являющихся конечными элементами. Это ведёт к ленточной разреженной структуре матрицы коэффициентов разрешающей системы уравнений.

Метод конечных элементов обладает следующими достоинствами:

– исследуемые объекты могут иметь любую форму и различную физическую природу – твёрдые деформируемые тела, жидкости, газы, электромагнитные среды;

– конечные элементы могут иметь различную форму и различные размеры;

– можно исследовать однородные и неоднородные, изотропные и анизотропные объекты с линейными и нелинейными свойствами;

– можно решать как стационарные, так и нестационарные задачи;

– можно решать контактные задачи;

– можно моделировать любые граничные условия;

– вычислительный алгоритм, представленный в матричной форме, формально единообразен для различных физических задач и для задач различной размерности, что удобно для компьютерного программирования;

– на одной и той же сетке конечных элементов можно решать различные физические задачи, что облегчает анализ связанных задач;

– разрешающая система уравнений имеет экономичную разреженную симметричную ленточную матрицу «жёсткости», что ускоряет вычислительный процесс на ЭВМ;

– удобно осуществляется иерархическая дискретизация исследуемой области на подобласти с образованием суперэлементов, что позволяет эффективно использовать параллельное решение задачи.

***1.3.2*** Решение задачи теплообмена методом конечных элементов делится на два основных этапа.

Первый этап численного решения задачи включает выбор элементов и способа расположения в них узловых точек, разбиение области на элементы и размещение узлов, а также определение функции формы. Следует учитывать, что эти функции существенным образом зависят от вида используемых элементов и способа расположения узлов.

Вторым этапом является составление системы разностных уравнений. Система уравнений для определения температур в каждом узле составляется на основании условий минимума функционала.

Третьим этапом является решение системы уравнений прямыми или итерационными методами [3, с. 4].

**2 АЛГОРИТМИЧЕСКИЙ АНАЛИЗ ЗАДАЧИ**

**2.1 Постановка задачи**

***2.1.1*** Для решения задачи теплопроводности для стержня, как правило, требуется произвести расчёт температурного поля. При этом необходимо учитывать начальные и граничные условия.

Существует два вида температурного поля – стационарное и нестационарное. Стационарным температурным полем называется такое поле, в котором температура в любой его точке не изменяется во времени, т.е. является функцией, параметрами которой являются координаты точки. Математическая запись такого поля приведена в формуле (2.1).

|  |  |  |
| --- | --- | --- |
|  |  | (2.1) |

где  –пространственные координаты в декартовой системе;

 *­*– время.

Нестационарным температурным полем называется такое поле, температура которого изменяется не только в пространстве, но и с течением времени. Такое поле задаётся уравнением (2.2).

|  |  |  |
| --- | --- | --- |
|  |  | (2.2) |

В исследуемой задаче необходимо произвести расчёт стационарного температурного поля, соответственно, в данной задаче отсутствуют начальные условия, а также такой параметр, как время [4, с. 7].

Для описания процесса стационарного переноса тепла внутри стержня используется дифференциальное уравнение теплопроводности. Данное уравнение имеет следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2.3) |

В исследуемой задаче расчёт температурного поля проводится с учётом наличия внутренних источников теплоты . Соответственно, дифференциальное стационарное уравнение теплопроводности (для одномерного случая) примет следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2.4) |

где  –функция переменной теплопроводности.

Также в поставленной задаче заданы граничные условия третьего рода. Математическая запись данного граничного условия имеет следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2.5) |

где  –коэффициенты теплоотдачи на левой и правой границе;

 – мощность теплового потока на левой и правой границе [5, с.27].

***2.1.2*** Для решения поставленной задачи необходимо разработать приложение для вычисления распределения температуры в стержне с прямоугольным сечением заданной ширины, высоты и длины при наличии внутренних источников теплоты, а также теплообмена с окружающей средой на торцах. Для проверки правильности вычислений необходимо произвести аналогичный расчет в программном комплексе *ANSYS.*

Необходимыми требованиями к разрабатываемому приложению являются:

– наличие графического пользовательского интерфейса, содержащий поля ввода для параметров задачи и позволяющий выводить результаты расчета в виде текста и графиков;

– наличие алгоритма построения разностной схемы;

– наличие алгоритма, составляющего систему линейных алгебраических уравнений;

– наличие алгоритма для решения составленной системы линейных алгебраических уравнений.

В качестве исходных данных в программе выступают:

– длина стержня и размеры прямоугольного сечения ;

– коэффициент теплопроводности ;

– коэффициенты теплообмена на левом и правом торцах стержня ;

– мощность внутренних источников теплоты .

– отсутствует теплообмен на боковых сторонах стержня, т.е. боковые стенки стержня изолированы.

Чертёж стержня представлен в приложении Б.

**2.2 Метод теплового баланса для решения задачи распределения температуры в стержне**

В методе конечных разностей основной проблемой является выбор вида аппроксимации дифференциального оператора. В простых случаях этот оператор аппроксимируется простейшим способом – производные в дифференциальном уравнении и граничных условиях заменялись конечными разностями. Однако, для более сложных задач, описываемых нелинейными уравнениями и уравнениями с переменными коэффициентами, замена производных конечными разностями может привести к схемам, имеющим большую погрешность, либо вообще окажутся непригодными для расчётов.

Для получения разностного решения, которое хорошо описывает реальный процесс изменения температурного поля, необходимо учитывать выполнение закона сохранения энергии.

Для непрерывного точного решения закон сохранения энергии выполняется для произвольной области тела. Для разностного решения требование закона сохранения энергии имеет важную особенность, обусловленную дискретным разбиением тела. Соответственно, поскольку поиск разностного решения производится в отдельных точках тела, то необходимо разбить тело на некоторое число элементарных объёмов, каждый из которых будет включать одну точку, а затем потребовать выполнение закона сохранения энергии как для произвольного элементарного объёма. Данное требование будет выполнено, если обеспечить условие согласования тепловых потоков для любых соседних элементарных объёмов, которое заключается в равенстве значений протекающих через общую границу тепловых потоков.

Разностные схемы, при которых получаются численные решения, удовлетворяющие закону сохранения энергии, называются консервативными.

Для построения консервативной схемы необходимо первоначально нанести сетку на исследуемое тело. Далее назначаются элементарные объёмы. Они могут быть назначены любым способом, однако наиболее естественным и широко распространённым является выбор границ элементарных объёмов в серединах отрезков, образованных соседними узлами сетки (рисунок 2.1)

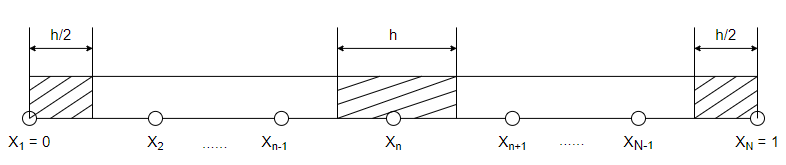


Рисунок 2.1 – Элементарные объёмы

Таким образом, в данном случае элементарные объёмы имеют следующий вид: 

Для точного решения закон сохранения энергии для элементарного объёма  записывается в виде:

|  |  |  |
| --- | --- | --- |
|  |  | (2.6) |

и условие согласования выполняется автоматически, поскольку поток, протекающий через общую границу двух объёмов, равен для любого из них значению  в их общей граничной точке.

Разностным аналогом для уравнения (2.6) является следующее уравнение:

|  |  |  |
| --- | --- | --- |
|  |  | (2.7) |

Уравнение (2.7) записано для элементарного объёма с центром в точке . Для соседнего левого элементарного объёма с центром в точке  получится уравнение, аналогичное уравнению (2.7). Для правой границы с учётом выполнения условия согласования уравнение имеет следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2.8) |

Вышеописанный способ построения консервативных разностных схем называется методом теплового баланса. В данном методе разностные уравнения получаются путём аппроксимации отношений теплового баланса, записанных для элементарных объёмов.

**2.3 Применение метода теплового баланса для решения задачи распределения температуры в стержне**

Применение метода теплового баланса состоит из нескольких этапов:

– исследуемая область разбивается на элементарные объёмы (ячейки), построенные вокруг каждого узла сетки;

– для всех внутренних и граничных ячеек записываются уравнения теплового баланса. В качестве уравнений баланса для граничных ячеек используются граничные условия;

– аппроксимируются члены, входящие в уравнения теплового баланса.

Поскольку число элементарных ячеек равно числу узлов пространственного разбиения, то в результате выполнения вышеописанных этапов получается система алгебраических уравнений – разностная схема. Решение данной системы является разностным решением.

Для поставленной задачи стационарное уравнение теплопроводности выглядит следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | (2.9) |

Выбирается равномерная сетка, т.е. её узлы находятся на одинаковом расстоянии друг от друга.

Уравнение теплового баланса для внутренней элементарной ячейки имеет следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2.10) |

где  – тепловые потоки на границах элементарных ячеек.

Уравнение (2.10) представляет собой закон сохранения энергии для внутренней элементарной ячейки и имеет следующий смысл: сумма потоков от левой и правой границы, а также от мощности внутренних источников равно нулю.

Все составляющие уравнения теплового баланса относятся к единице площади поперечного сечения и выражаются в .

Так как коэффициент теплопроводности постоянен на всей длине стержня, то приближение для потока строится исходя из предположения о его малом изменении на каждом интервале. Из закона Фурье тепловой поток можно выразить следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | (2.11) |

После интегрирования равенства (2.11) на интервале  получается следующее выражение:

|  |  |  |
| --- | --- | --- |
|  |  | (2.12) |

Предполагая, что тепловой поток мало изменяется на отрезке , полагаем, что . В результате получается следующее приближение:

|  |  |  |
| --- | --- | --- |
|  |  | (2.13) |

Таким образом, тепловые потоки, проходящие через границы элементарной ячейки можно выразить через разности температур в узлах:

|  |  |  |
| --- | --- | --- |
|  |  | (2.14) |

Подставляя полученные приближения тепловых потоков в уравнение теплового баланса для внутренней элементарной ячейки (2.10), получается следующее разностное уравнение:

|  |  |  |
| --- | --- | --- |
|  |  | (2.15) |

Для граничных элементарных ячеек заданы граничные условия третьего рода. Для аппроксимации уравнения (2.5) используются полученные выше выражения для тепловых потоков. Уравнения для граничной ячейки при *x = 0* составляются исходя из закона сохранения энергии:

|  |  |  |
| --- | --- | --- |
|  |  | (2.16) |

где  – тепловой поток, рассеиваемый в среду на границе;

 – мощность, выделяемая внутренними источниками.

Аналогичным образом строится аппроксимация граничного условия при *x = l.*

После вывода уравнений для внутренних и граничных элементарных ячеек составляется система уравнений вида . Причём, количество неизвестных переменных равно количеству уравнений. Примером системы уравнений при четырёх неизвестных является формула (2.17).

|  |  |  |
| --- | --- | --- |
|  |  | (2.17) |

Решением данной системы уравнения является вектор , что является разностным решением [6, c. 69].

**2.4 Метод Гаусса для решения системы линейных алгебраических уравнений**

Метод Гаусса – классический метод решения системы линейных алгебраических уравнений (далее СЛАУ). Суть метода заключается в последовательном исключении переменных, когда с помощью элементарных преобразований система уравнений приводится к равносильной системе треугольного вида. Далее в такой системе находятся все переменные этой системы, начиная с последнего уравнения.

Он обладает рядом преимуществ по сравнению с другими методами:

– нет необходимости предварительно исследовать систему уравнений на совместность;

– 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;

– приводит к результату при сравнительно небольшом количестве вычислительных операций.

Решение СЛАУ методом Гаусса делится на два этапа: прямой и обратный ход.

Есть система из *n* линейных уравнений с *n* неизвестными следующего вида:

|  |  |  |
| --- | --- | --- |
|  |  | (2.18) |

Тогда матричный вид записи СЛАУ (2.18) имеет вид , где *A –* матрица коэффициентов перед неизвестными переменными, *X –* матрица-столбец неизвестных переменных, *B –* матрица свободных членов. Для того чтобы существовало решение, матрица *A* должна быть невырожденной, т.е. определитель матрицы не должен быть равен нулю.

Для решения СЛАУ методом Гаусса первым этапом необходимо составить расширенную матрицу. Данная матрица получается при добавлении в качестве *(n + 1)* столбца матрицу-столбец свободных членов. Расширенная матрица имеет следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2.18) |

Следующим этапом решения является прямой ход. На данном этапе строки системы приводят к ступенчатой или треугольной форме. Для этого среди элементов первого столбца необходимо найти ненулевой элемент, затем с помощью перестановки строк перенести его на крайнее левое положение. Далее необходимо исключить переменную  из всех уравнений, начиная со второго, путём прибавления к ним первого уравнения, умноженного на . После проведения вышеописанных действий СЛАУ примет следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2.19) |

где ;

.

Далее производятся аналогичные действия с остальными уравнениями. При этом, после исключения переменной  также и исключаются соответствующие этой переменной строка и столбец.

Прямой ход заканчивается после того, как система принимает следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2.20) |

Завершающим этапом решения является обратный ход. На данном этапе производится вычисление  из последнего уравнения как , затем с помощью полученного значения  вычисляется значение  из предпоследнего уравнения и так до  из первого уравнения системы [7].

**3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ЗАДАЧИ**

**3.1 Структура разрабатываемого приложения**

***3.1.1*** Для реализации анализа распределения температур в стержне с внутренним источником теплоты и теплообменом с окружающей средой в приложении разработаны классы, реализующие определённые объекты задачи. Листинг разработанной программы приведён в приложении А.

Разработанное приложение содержит следующие библиотеки классов:

– «*Models» –* хранит классы моделей задачи;

– «*SolverElements» –* хранит классы решателя;

– *«TempSolver» –* хранит классы и формы, реализующие графический интерфейс.

***3.1.2*** Библиотека классов *«Models»* хранит следующие классы:

– *«Node» –* описывает узел разностной схемы;

– *«Figure» –* описывает исследуемое геометрическое тело;

– *«Model» –* содержит исходные данные задачи, массив узлов разностной сетки и объект геометрического тела;

– *«EquationSystem» –* содержит матричное представление системы алгебраических уравнений.

В классе *«Node»* содержатся свойства, описывающие определённый узел разностной схемы. Описание свойств этого класса приведено в таблице 3.1.

Таблица 3.1 – Свойства класса «*Node*»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Описание** | **Тип данных** |
| *Index* | Номер узла | *int* |
| *X* | Координаты узла по оси *X* | *float* |
| *Temperature* | Значение температуры в узле | *float* |

Класс «*Figure*» содержит свойства, описывающие размеры прямоугольного стержня, а также содержит метод *«GetSquare»* для определения площади сечения. Описание свойств этого класса приведено в таблице 3.2.

Таблица 3.2 – Свойства класса «*Figure*»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Описание** | **Тип данных** |
| 1 | 2 | 3 |
| *SectionWidth* | Ширина сечения | *float* |

Продолжение таблицы 3.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| *SectionHeight* | Высота сечения | *float* |
| *Length* | Длина стержня | *float* |

Класс *«Model»* содержит свойства, содержащие исследуемый геометрический объект, массив узлов разностной схемы, а также параметры задачи. Также данный класс содержит следующие методы:

– *«CreateGrid» –* закрытый метод, который генерирует узлы разностной сетки. В качестве принимаемого параметра является количество узлов сетки;

– *«GetMaxTemperature» –* определяет узел, в котором зафиксировано максимальное значение температуры;

– *«GetMinTemperature» –* определяет узел, в котором зафиксировано минимальное значение температуры;

– *«GetTemperatureGradient» –* возвращает массив температур в каждом узле.

Свойства этого класса описаны в таблице 3.3.

Таблица 3.3 – Свойства класса «*Model*»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Описание** | **Тип данных** |
| *Nodes* | Список с объектами класса *«Node»* | *List<Node>* |
| *GridStep* | Расстояние между узлами сетки | *float* |
| *Figure* | Ссылка на объект класса *«Figure»* | *Figure* |
| *ThermalConductivityCoeff* | Коэффициент теплопроводности | *float* |
| *HeatFlow* | Мощность внутреннего источника тепла | *float* |
| *LeftConvectionCoeff* | Коэффициент теплоотдачи на левой границе | *float* |
| *RightConvectionCoeff* | Коэффициент теплоотдачи на правой границе | *float* |

Класс *«EquationSystem»* содержит свойства, содержащие массивы с коэффициентами СЛАУ. Также данный класс содержит закрытый метод *«SetZeros»* для заполнения массивов начальными значениями. Описание свойств данного класса приведено в таблице 3.4

Таблица 3.4 – Свойства класса «*EquationSystem*»

|  |  |  |
| --- | --- | --- |
| **Поле** | **Описание** | **Тип данных** |
| *Coefficients* | Матрица коэффициентов СЛАУ перед неизвестными переменными | *float[,]* |
| *FreeOdds* | Матрица свободных членов СЛАУ | *float[]* |

***3.1.3*** Библиотека классов *«SolverElements»* хранит класс *«Solver»*. Данный класс составляет СЛАУ для разностной схемы и производит поиск решения. Метод *«CreateEquationsSystem»* генерирует СЛАУ исходя из информации об узлах сетки. Принимаемым параметром является объект класса *«Model»*. Метод *«GaussMethod»* производит расчёт СЛАУ и записывает значения температур соответствующим узлам сетки. В качестве принимаемого параметра служит объект класса *«EquationSystem»* и *«Model*».

***3.1.4*** Библиотека классов *«TempSolver»* содержит класс *«Programm»,* который служит точкой запуска приложения, а также класс *«Form»*, реализующий графический интерфейс пользователя. В классе *«Form»* реализованы обработчики событий для получения данных из полей ввода, обработки этих данных и отправка классам-решателям для получения решения задачи.

**4 ВЕРИФИКАЦИЯ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ**

**4.1 Основы модульного тестирования**

Модульное тестирование – это тестирование программы на уровне отдельно взятых модулей, функций или классов. Цель модульного тестирования состоит в выявлении локализованных в модуле ошибок в реализации алгоритмов, а также в определении степени готовности системы к переходу на следующий уровень разработки и тестирования. Модульное тестирование проводится по принципу "белого ящика", то есть основывается на знании внутренней структуры программы, и часто включает те или иные методы анализа покрытия кода.

Модульное тестирование обычно подразумевает создание вокруг каждого модуля определенной среды, включающей заглушки для всех интерфейсов тестируемого модуля. Некоторые из них могут использоваться для подачи входных значений, другие для анализа результатов, присутствие третьих может быть продиктовано требованиями, накладываемыми компилятором и сборщиком.

На уровне модульного тестирования проще всего обнаружить дефекты, связанные с алгоритмическими ошибками и ошибками кодирования алгоритмов, типа работы с условиями и счетчиками циклов, а также с использованием локальных переменных и ресурсов [8].

Так как доступ к закрытым методам класса извне не возможен, не используя механизмы рефлексии, то и тестировать такие методы модульными тестами нельзя. Остаются только публичные методы. Не нужно знать о приватных методах, которые используются публичными. Это означает, что при правильном выполнении публичных методов все использующиеся скрытые внутри класса методы работают корректно и отдельно их тестировать не нужно и нельзя из-за модификаторов доступа.

**4.2 Тестирование пользовательского интерфейса**

***4.2.1*** Для работы программы требуется полностью скомпилированная её версия. Основным исполняющим файлом программы является *TempSolver.exe*.

После запуска исполняющего файла *TempSolver.exe* открывается главное окно программы, содержащее поля для ввода необходимых данных для анализа поставленной задачи. Внешний вид главного окна программы представлен на рисунке 4.1.

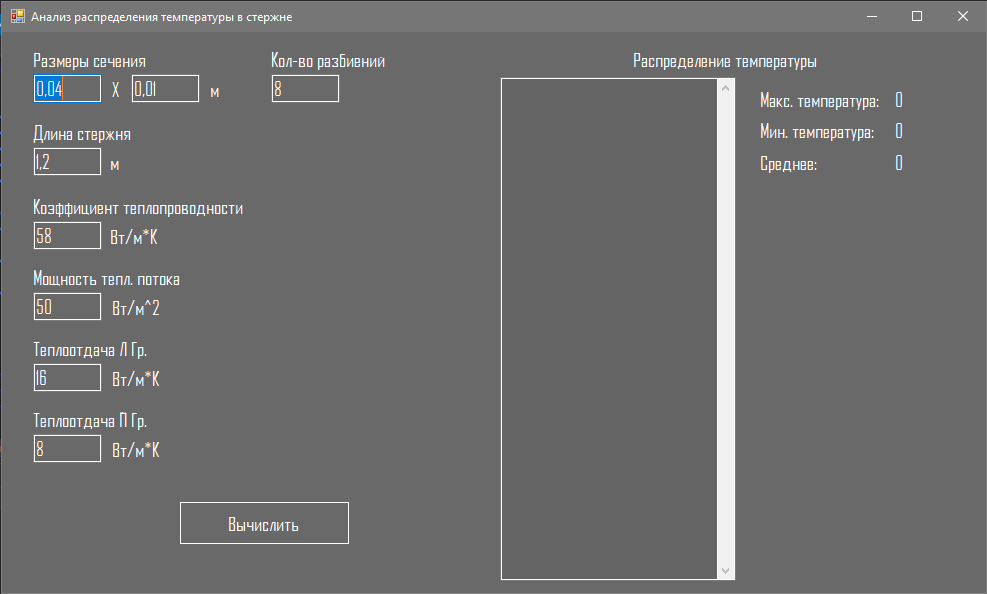


Рисунок 4.1 – Внешний вид главного окна программы

***4.2.2*** Для выполнения расчётов необходимо заполнить поля ввода данных. В данные поля вводятся исходные данные решаемой задачи. Причём, для решения обязательно должны быть заполнены все поля.

Пользователь должен заполнить следующие поля ввода данных:

– «Размеры сечения» – в данное поле вводятся ширина и высота сечения стержня в метрах. Вводимое значение должно быть положительным и отличным от нуля числом;

– «Длина стержня» – в данное поле вводится длина стержня в метрах. Вводимое значение должно быть положительным и отличным от нуля числом;

– «Коэффициент теплопроводности» – в данное поле вводится коэффициент теплопроводности материала, из которого изготовлен стержень. Единица измерения – . Вводимое значение должно быть положительным и отличным от нуля числом;

– «Мощность тепл. потока» – в данное поле вводится мощность внутреннего источника тепла в . Вводимое значение должно быть положительным и отличным от нуля числом;

– «Теплоотдача Л Гр.» – в данное поле вводится коэффициент теплоотдачи на левой границе стержня. Единица измерения – . Вводимое значение должно быть положительным и отличным от нуля числом;

– «Теплоотдача П Гр.» – в данное поле вводится коэффициент теплоотдачи на правой границе стержня. Единица измерения – . Вводимое значение должно быть положительным и отличным от нуля числом;

– «Кол-во разбиений» – в данное поле вводится размер сетки, а именно число узлов. Вводимой значение должно быть положительным, целым, отличным от нуля числом.

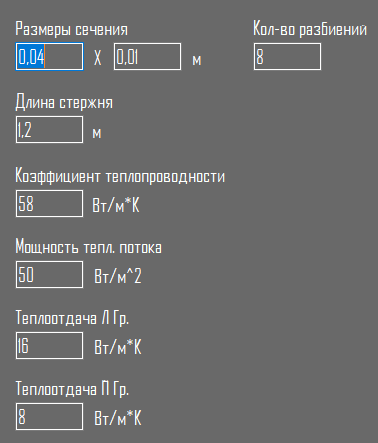


Рисунок 4.2 – Внешний вид полей ввода начальных данных

***4.2.3*** После ввода начальных данных в соответствующие поля, для произведения расчётов необходимо нажать на кнопку «Вычислить». После нажатия на кнопку программа произведёт вычисления. В качестве результатов выводятся температуры по всей длине стержня, максимальное, минимальное и среднее значения температуры. Вид главного окна программы после вычислений представлен на рисунке 4.4.

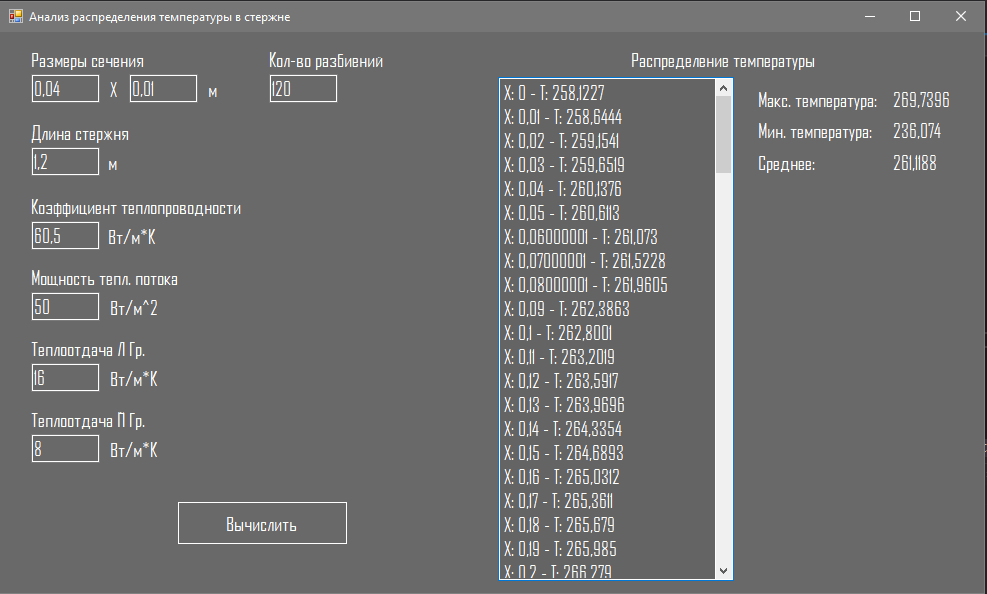


Рисунок 4.4 – Вид программы после выполнения расчётов

**4.3 Сравнение полученных результатов**

Для сравнения полученных результатов необходимо провести аналогичный анализ в программном комплексе *ANSYS*. Все вводимые параметры совпадают с параметрами при тестировании разработанной программы.

Анализ выполняется в пакете *ANSYS Workbench R2 2019*. Первым этапом необходимо выбрать тип исследования. Все возможные варианты исследований расположены в списке *«Analysis Systems».* Для решения поставленной задачи необходимо выбрать тип *«Steady-State-Thermal»* (рисунок 4.6).

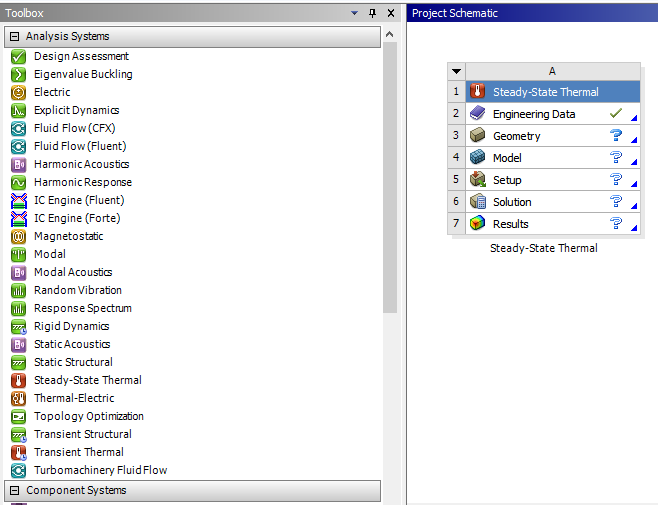


Рисунок 4.6 – Выбор типа исследования

Далее необходимо построить геометрическую модель исследуемого объекта. Для этого необходимо открыть программу *«ANSYS Design Modeler»* и построить в ней эскиз детали (рисунок 4.7).

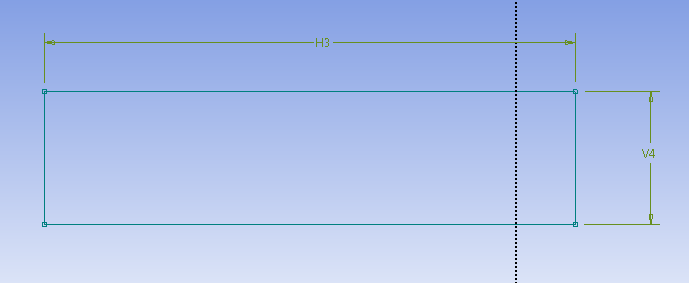


Рисунок 4.7 – Эскиз детали

После создания эскиза необходимо вытянуть бобышку для создания детали. Для этого необходимо выбрать эскиз и нажать на кнопку *«Extrude».* Далее нужно указать длину бобышки и нажать на кнопку *«Generate»*. Результат выполнения операции изображён на рисунке 4.8.

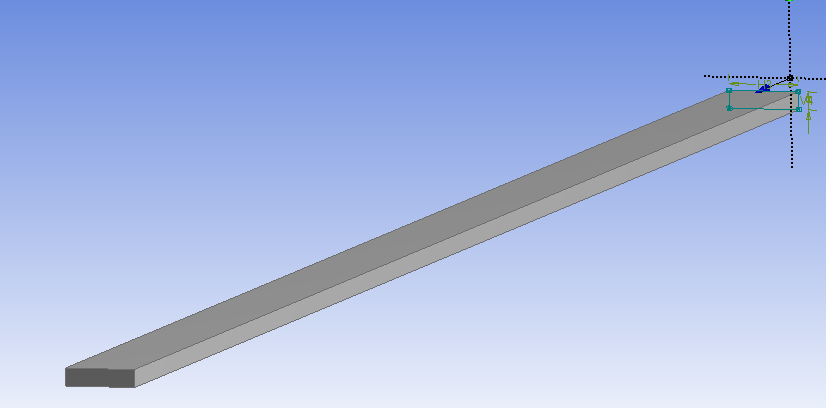


Рисунок 4.8 – Полученная деталь

Следующим этапом является настройка материала детали, построение сетки, задание граничных условий и получение решения. В качестве материала детали выступает сталь с коэффициентов теплопроводности в 60,5 . Для создания сетки первым шагом необходимо указать размер сетки. Для указания размера сетки используется инструмент *«Face Sizing»*. Размер сетки устанавливается равным 0,1 м. Для построения сетки необходимо нажать на кнопку *«Generate»*. Результат изображён на рисунке 4.9.

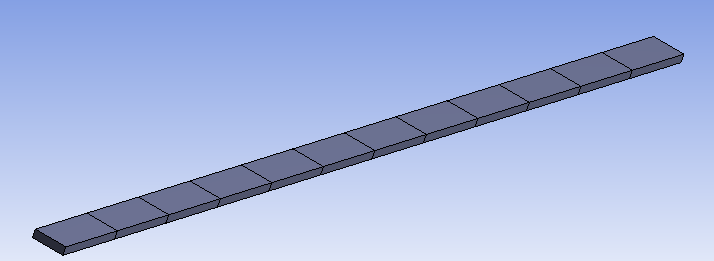


Рисунок 4.9 – Результат построения сетки

Далее необходимо указать условия и ограничения задачи. По условию, на обоих торцах стрежня происходит отдача теплоты в окружающую среду, т.е. задано граничное условие третьего рода. Для того чтобы задать условие, необходимо добавить свойство *«Convection»*, указать грань и задать коэффициент теплоотдачи. Также необходимо изолировать боковые грани стержня от контакта с окружающей средой. Для этого используется свойство *«Heat Flow».* Для того чтобы боковые грани не отдавали тепловую энергию в окружающую среду, необходимо выбрать нужные грани и в поле *«Magnitude»* задать значение 0. Для задания внутреннего теплового потока используется свойство *«Heat Flux»*. В поле *«Magnitude»* указывается мощность теплового потока. Для решаемой задачи мощность составляет. Заданные условия и ограничения изображены на рисунке 4.10.

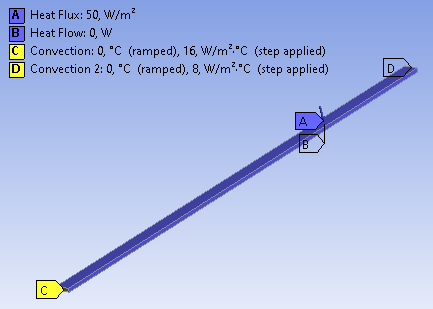


Рисунок 4.10 – Условия и ограничения решаемой задачи

Последним этапом является расчёт и анализ полученных результатов. Для запуска решателя необходимо нажать на кнопку *«Solve».* В результате графически отобразится распределение температуры в стержне, а также будет выведен градиент температур с максимальным и минимальным значением. Результаты расчётов приведены на рисунке 4.11.

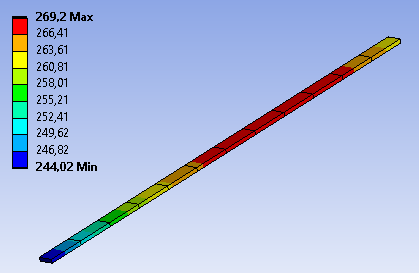


Рисунок 4.11 – Результаты расчётов

После нахождения значений температурного поля стержня при помощи программного пакета *ANSYS* необходимо найти погрешность вычислений. Для этого требуется сравнить максимальное, минимальное и среднее значение температуры из программного пакета *ANSYS* и разработанной программы. Данные значения из расчётов разработанной программы и *ANSYS* изображены на рисунках 4.12 и 4.13 соответственно.

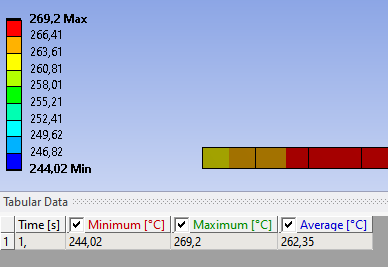


Рисунок 4.12 – Результаты вычислений программного пакета *ANSYS*

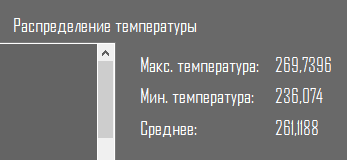


Рисунок 4.13 – Результаты вычислений разработанного приложения

Исходя из полученных результатов значения максимальной, минимальной и средней температуры практически совпадают. Для более точной проверки для рассчитанных значений температур в точках строятся графики.

Функционал разработанной программы позволяет выводить результаты вычисления температурного поля в виде текста, в котором указана координата узла и соответствующее ей значение температуры. В программном комплексе *ANSYS* можно экспортировать результаты вычислений в текстовый файл.

Сравнение вычислений производится с помощью программы *Exel.* Для этого в столбы записываются результаты вычислений и по этим данным строятся графики.

На основе табличного и графического сравнения значений температур можно с уверенностью сделать вывод о том, что погрешность вычислений минимальна. Результаты сравнения приведены на рисунке 4.14.

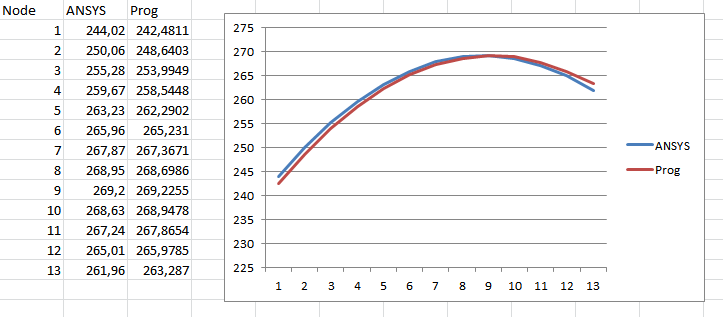


Рисунок 4.14 – Результат сравнения решений

**ЗАКЛЮЧЕНИЕ**

Разработан программный комплекс для реализации анализа распределения температур в стержне с боковым теплообменом и источником теплоты при помощи метода теплового баланса. Для разработки программы использовался объектно-ориентированный язык программирования *C#* на платформе *.NET Framework*. Графический интерфейс разработан с помощью технологии *Windows Forms.*

Разработанный программный комплекс полностью удовлетворяет поставленным требованиям и выполняет поставленные задачи. Произведены тесты программного комплекса на корректную работу. Верификация разработанного программного комплекса была выполнена с помощью программного пакета *ANSYS.* В результате проведения верификации процент погрешности не превышает допустимые нормы. Результаты вычислений разработанного приложения практически совпадают с результатами в системе *ANSYS*.

Так как приложение написано на объектно-ориентированном языке, оно может быть улучшено и дополнено различным функционалом. В частности, программа может быть дополнена методом построения неравномерной сетки и может быть улучшен алгоритм составления уравнений для узлов сетки.

**Список использованных источников**

1. Строительная механика: Электронный учебный курс. – Электрон. данные. ­– Режим доступа: http://www.stroitmeh.ru/lect83.htm. – Дата доступа: 13.05.2020.

2. Дегтярёв, А. А. Метод конечных разностей: учебное пособие / А. А. Дегтярёв. – Самара : СГАУ им. С. П. Королёва, 2011. – 79 с.

3. Фокин, В. Г. Метод конечных элементов: учебное пособие / В. Г. Фокин*.* – Самара : Самар. гос. техн. ун-т, 2010. – 131 с.

4. Лыков, А. В. Теория теплопроводности: учебное пособие / А. В. Лыков. – М. : Высшая школа, 1967. – 592 с.

5. Овсянник, А. В. Тепломассообмен: курс лекций для студентов специальностей 1-43 01 05 «Промышленная теплоэнергетика» дневн. и заоч. форм обучения и 1-43 01 07 «Техническая эксплуатация энергооборудования организаций» днев. формы обучения / А. В. Овсянник, М. Н. Новиков, А. В. Шаповалов. – Гомель : ГГТУ им. П. О. Сухого, 2010. – 212 с.

6. Дульнев, Г. Н. Применение ЭВМ для решения задач теплообмена: учеб. Пособие для теплофизич. и теплоэнергетич. спец. вузов / Г. Н. Дульнев, В. Г. Парфенов, А. В. Сигалов. – М.: Высш. шк., 1990. – 207 с.

7. Комраков, В. В. Численные методы математической физики : практикум по одноим. курсу для студентов специальности 1-40 01 02 «Информационные системы и технологии (по направлениям)» днев. и заоч. форм обучения / В. В. Комраков. – Гомель : ГГТУ им. П. О. Сухого, 2013. – 83 с.

8. Основы тестирования программного обеспечения: Файловый архив для студентов. Studfiles. – Электрон. данные. – Режим доступа: https://studfile.net/preview/4494386/. – Дата доступа: 20.05.2020.

**ПРИЛОЖЕНИЕ A**

(обязательное)

**Листинг программы**

Код *Node.cs:*

namespace Models

{

public class Node

{

public int Index { get; }

public float X { get; }

public float Temperature { get; set; }

public Node(int index, float x, float temperature = 0)

{

Index = index;

X = x;

Temperature = temperature;

}

}

}

Код *Figure.cs:*

namespace Models

{

public class Figure

{

// Ширина сечения

public float SectionWidth { get; private set; }

// Высота сечения

public float SectionHeight { get; private set; }

// Длина

public float Length { get; private set; }

public Figure(float sectionWidth, float sectionHeight, float length)

{

SectionWidth = sectionWidth;

SectionHeight = sectionHeight;

Length = length;

}

public float GetSquare()

{

return SectionHeight \* SectionWidth;

}

}

}

Код *Model.cs:*

using System;

using System.Collections.Generic;

namespace Models

{

public class Model

{

public List<Node> Nodes { get; private set; }

public float GridStep { get; private set; }

public Figure Figure { get; private set; }

public float ThermalConductivityCoeff { get; set; }

public float HeatFlow { get; set; }

public float LeftConvectionCoeff { get; set; }

public float RightConvectionCoeff { get; set; }

public Model(Figure figure, int gridSize)

{

Figure = figure;

GridStep = figure.Length / gridSize;

Nodes = new List<Node>(gridSize);

CreateGrid(gridSize);

}

private void CreateGrid(int gridSize)

{

float x = 0;

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

{

Nodes.Add(new Node(i, x));

x += GridStep;

}

}

public Node GetMaxTemperature()

{

Node node = Nodes[0];

for (int i = 1; i < Nodes.Count; i++)

{

if (Nodes[i].Temperature > node.Temperature)

{

node = Nodes[i];

}

}

return node;

}

public Node GetMinTemperature()

{

Node node = Nodes[0];

for (int i = 1; i < Nodes.Count; i++)

{

if (Nodes[i].Temperature < node.Temperature)

{

node = Nodes[i];

}

}

return node;

}

public float[] GetTemperatureGradient()

{

float[] gradient = new float[Nodes.Count];

for (int i = 0; i < Nodes.Count; i++)

{

gradient[i] = Nodes[i].Temperature;

}

Array.Sort(gradient);

return gradient;

}

}

}

Код *EquationSystem.cs:*

namespace Models

{

public class EquationsSystem

{

public float[,] Coefficients { get; private set; }

public float[] FreeOdds { get; private set; }

public EquationsSystem(int size)

{

Coefficients = new float[size, size];

FreeOdds = new float[size];

SetZeros();

}

private void SetZeros()

{

for (int i = 0; i < FreeOdds.Length; i++)

{

FreeOdds[i] = 0;

for (int j = 0; j < FreeOdds.Length; j++)

{

Coefficients[i, j] = 0;

}

}

}

}

}

Код *Sovler.cs:*

using Models;

namespace SolverElements

{

public class Solver

{

private static Solver instance;

public static Solver GetSolver()

{

if (instance == null)

{

instance = new Solver();

}

return instance;

}

/// <summary>

/// Создаёт СЛАУ всех узлов

/// </summary>

/// <param name="model">Модель</param>

/// <returns>СЛАУ</returns>

public EquationsSystem CreateEquationsSystem(Model model)

{

int eqSize = model.Nodes.Count;

float localConduct = model.ThermalConductivityCoeff / model.GridStep;

EquationsSystem system = new EquationsSystem(eqSize);

// Граничное условие 3 рода

// Левое

system.Coefficients[0, 0] = localConduct - model.LeftConvectionCoeff;

system.Coefficients[0, 1] = (-1) \* localConduct;

system.FreeOdds[0] = (-1) \* model.HeatFlow - model.HeatFlow \* model.GridStep / 2;

// Правое

system.Coefficients[eqSize - 1, eqSize - 1] = localConduct - model.RightConvectionCoeff;

system.Coefficients[eqSize - 1, eqSize - 2] = (-1) \* localConduct;

system.FreeOdds[eqSize - 1] = (-1) \* model.HeatFlow - model.HeatFlow \* model.GridStep / 2;

// Внутренние узлы

for (int i = 1; i < eqSize - 1; i++)

{

system.Coefficients[i, i - 1] = localConduct;

system.Coefficients[i, i] = (-2) \* localConduct;

system.Coefficients[i, i + 1] = localConduct;

system.FreeOdds[i] = (-1) \* model.HeatFlow \* model.GridStep;

}

return system;

}

public void GaussMethod(EquationsSystem system, Model model)

{

int size = system.FreeOdds.Length;

// Расширенная матрица

float[,] extMatrix = new float[size, size + 1];

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

{

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

{

extMatrix[i, j] = system.Coefficients[i, j];

}

}

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

{

extMatrix[i, size] = system.FreeOdds[i];

}

// Прямой ход

for (int k = 1; k < size; k++)

{

for (int j = k; j < size; j++)

{

float m = extMatrix[j, k - 1] / extMatrix[k - 1, k - 1];

for (int i = 0; i < size + 1; i++)

{

extMatrix[j, i] -= m \* extMatrix[k - 1, i];

}

}

}

// Обратный ход

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

{

model.Nodes[i].Temperature = extMatrix[i, size] / extMatrix[i, i];

for (int c = size - 1; c > i; c--)

{

model.Nodes[i].Temperature -= extMatrix[i, c] \* model.Nodes[c].Temperature / extMatrix[i, i];

}

}

}

}

}

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Чертеж стержня**