Федеральное государственное автономное
образовательное учреждение высшего образования
«Волгоградский государственный университет»
институт Математики и информационных технологий
кафедра Информационных систем и компьютерного моделирования

Допустить работу к защите
Зав. каф. ИСКМ
_____А. В. Хоперсков
«___» _____ 2021 г.

Губенко Иван Борисович

Программный комплекс для расчета остаточных напряжений и деформаций металлоконструкций

Выпускная квалификационная работа
по направлению подготовки бакалавров
09.03.04 Программная инженерия

Студент	Губенко И.Б.	_____
		(подпись)
Руководитель выпускной квалификационной работы	Храпов С.С., к.ф.-м.н., доцент каф ИСКМ	
Рецензент	Еремин М.А., к.ф.-м.н., доцент каф. ТФ и ВП	

Волгоград 2021

Содержание

Введение

Целью данной работы является изучить и пройти все этапы разработки программного обеспечения при создание программного комплекса для расчета остаточных напряжений и деформаций металлоконструкций.

Для повышения качества и скорости разработки программного комплекса требуется декомпозировать все процессы и разбить их на несколько этапов.

Одна из первостепенных задач при старте разработки — это изучение предметной области. Избыточное изучение предметной области способствует уменьшению количества конфликтных ситуаций или проблем, которые могут возникнуть на этапе проектирования

Важным этапом разработки качественного программного обеспечения, является этап проектирования системы. На этом этапе требуется полностью определить всю спецификацию для разработки. Определить с какой целью создается программное обеспечение, оценить затраты и вложения, так же требуется утвердить алгоритмы и способы реализации. В результате составления информационной модели должно полностью сформироваться представление о работе готового программного обеспечения и вариантах его использования.

Первым этапом создания информационной модели является постановка задачи, и определение с какой целью создается программное обеспечение.

Вторым этапом является анализ объекта. В результате второго этапа выявляются составляющие объекты и расстановка связей между ними.

Во время третьего этапа строится информационная модель. Построения связано с задачами и целями объекта моделирования, все объекты включают в себя большое количество параметров и свойств, при построении выделяются значимые свойства.

По итогу в результате проектирования собирается информационная модель системы, представляющая собой модель организации работы системы.

Информационная модель включает в себя схематичное пояснение, каким образом работает программа и при каких входных данных происходит функционирование системы и получении выходных данных.

Информационная модель является полезным и важным инструментом, при создании сложных комплексов и систем.

Информационная модель позволяет проконтролировать технические вопросы и минимизировать возникновение неточностей и ошибок при реализации в будущем. Следующим и заключительным этапом в разработке будет переход на стадию написания кода. Написание кода это в большинстве случаев итерационный процесс, который требует максимальной внимательности и осторожности ведь любое даже незначительное отклонение от спроектированной архитектуры может привести к неработоспособности проекта и потери времени и средств на исправление. Поэтому при попытке изменения архитектуры приложения, требуется сначала вернуться на стадию проектирования и провести анализ заявленных изменений

Актуальность работы:

В настоящее в предприятия внедряются все больше новых технологий, которые требуют большего контроля качества для получения приемлемого качества изделий. Одной из таких технологий является аддитивное производство. С каждым годом все больше предприятий и организаций начинают использовать методы аддитивного производств. С ростом популярности аддитивного производства у потребителя появляются новые требования для повышения производительности станков. Оптимизация производства один из ключевых факторов, влияющих на общую прибыль предприятия, уменьшая число бракованных деталей, позволяет экономить средства и ресурсы, повышая общую производительность предприятия. Одним из вариантов оптимизации в сфере аддитивного производства это прогнозирования деформаций, которые могут возникнуть при производстве изделия. Значительно дешевле произвести компьютерное моделирование метода аддитивного производства и получить информацию о том какого

качества получится модель. Текущий рынок программного обеспечения для расчета остаточного напряжения предлагает программные комплексы для внедрения в крупные предприятия и цена у них соответствующая.

Задачи на преддипломную практику:

1) Изучить рекомендуемую научную литературу и ознакомиться с способами расчёта остаточного напряжения и деформаций металлоконструкций;
2) Создать информационною модель программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций
3) Реализовать программный комплекс для расчета остаточных напряжений и деформаций металлоконструкций

1 Изучение предметной области для создания программного комплекса расчета остаточных напряжений и деформаций металлоконструкций

1.1 Методы аддитивного производства изделий из мелкодисперсного металлического порошка

Аддитивное производство (АП) – технология создания трехмерных изделий на основе цифровой модели путем послойного добавления материала. Эта особенность делает возможным создания сложных изделий по смоделированной объемной модели без создания дополнительной оснастки или литьевых форм [6]. Изделия высокой сложности могут быть созданы в короткие сроки без дополнительных действий. Благодаря этому сейчас аддитивное производство принимается в качестве инновационной парадигмы проектирования и производства в различных сферах деятельности человека и используется для создания изделий [20].

Один из векторов развития аддитивного производства – гибридное изготовления деталей, включает в себя сам процесс послойного производства деталей и дополнительную механическую обработку. Гибридный способ изготовления подходит для производства больших деталей низкой и средней сложности [8].

1.1.1 Селективная лазерное спекание, обзор метода аддитивного производства

Селективное лазерное спекание (Selective laser sintering - SLS) - значимый метод в аддитивных технологиях. Осуществляется с помощью использования лазерных излучателей высокой мощности [14]. Специальный порошок, который имеется в камере, с помощью тонкого равномерного слоя помещается на рабочую область особым роликом, что помогает разравнивать порошок. Луч лазера

обрисовывает на поверхностном слое порошка силуэт модельного сечения. С помощью подвижного зеркала устанавливается направление луча лазера [27].

Лазерный луч представляет собой сфокусированный тепловой источник, в котором осуществляется спекание гранул материала, в результате чего создается металл или же твердый полимер, на месте, где проходил лазерный луч. Так образуется каждый новый слой намеченной детали.

Платформа рабочей камеры является подвижной и после того, как произвелась лазерная обработка слоя, она спускается вниз с целью нанести последующий слой порошка на предыдущий затвердевший слой. Одновременно дно в камере хранения порошка перемещается наверх. Валик разравнивателя наносит новый порошковый слой, накрывая прошлый слой, под влиянием лазера текущий слой спекается с прошлым. Процесс повторяется пока модель полностью не выполнится.

Таким образом, распечатываемое изделие вырастает снизу-вверх. В наличие поддержек нет необходимости, так как порошок, который еще не затвердел обволакивает изделие и в процессе 3D-печати удерживает его составляющие [37]. SLS технологии применимы во многих сферах: медицине, авиастроении, строении машин, космонавтике.

На рисунке 1 продемонстрирована схема устройства 3D-принтера технологии селективного лазерного спекания [11].

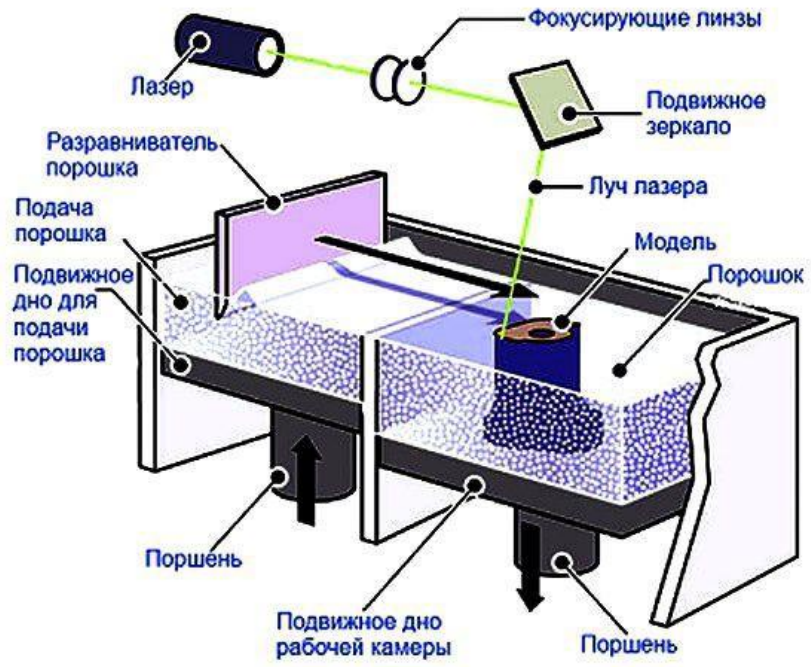

Рисунок 1 – Схема SLS 3D-принтера

Преимущества изготовления деталей селективным лазерным спеканием:

1) большая производительность: SLS-принтеры не требуют полного расплавления частиц материала, это способствует более быстрой работе, по сравнению с остальными порошковыми 3D-принтерами;
2) достоинства в механических свойствах выполненных деталей: хорошая прочность изделия, точность, поверхность обладает достаточным качеством;
3) благодаря объемной печатной камере существует возможность изготавливать изделия больших размеров или набор миниатюрных объектов за единственную печать;
4) процесс печати обеспечивает малое количество отходов, незадействованный материал можно применять в печати снова.

1.1.2 Обзор метода аддитивного производства, электронно-лучевая плавка

Электронно-лучевая плавка («Electron Beam Melting» или EBM) – метод аддитивного производства изделий из металла. Электронно-лучевой плавка сопоставима с технологией выборочной лазерной плавкой (SLS). Отличием является только в источнике энергии для плавки в EBM используются электронные излучатели вместо лазеров. В технологии электронно-лучевой плавки используются мощные пучки электронов для сплавления порошкового металла в вакуумной среде камеры [25].

Электронно-лучевая плавка создает детали особо высокой плотности и прочности. Сочетание высокой температуры и вакуума позволяет добиться уменьшения внутреннего напряжения. Благодаря уменьшения внутренних дефектов увеличивается прочность изделия, и дальнейшая термообработка изделия не требуется дальнейшая. На рисунке 2 изображена схема устройства 3D-принтера с технологией электронно-лучевой плавки.

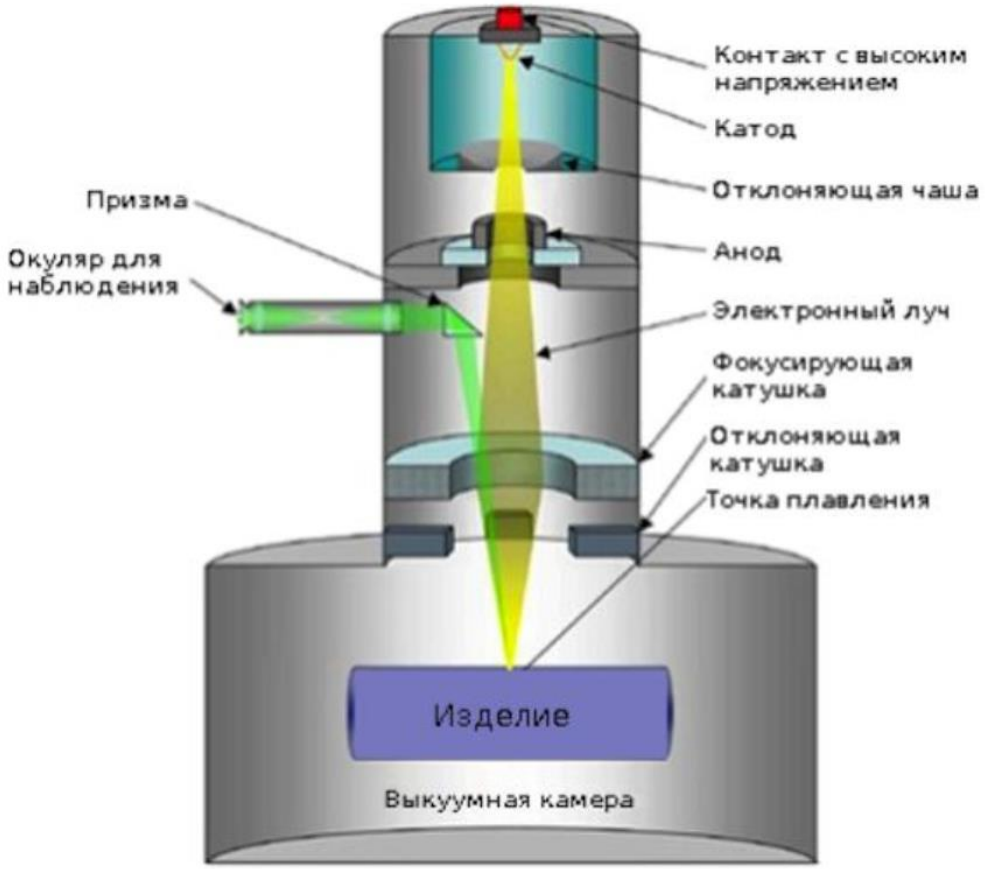

Рисунок 2 – Схема устройства EBM 3D-принтера

EBM обладает высокой скоростью построения модели, за счет высокой мощности излучателя. Основное различие технологий DMLS и EBM заключается в следующем:

1) разные источники энергии. В DMLS волокно-оптический лазер, а в EBM применяется электронный луч;
2) условия среды. Как упоминалось ранее, в DMLS печать протекает в среде защитных газов, а в EBM в вакууме.

Преимущества EBM:

1) низкий уровень внутренних дефектов;
2) хорошие свойства материала (а также усталостные свойства);
3) минимальные остаточные напряжения;
4) малое количество отходов.

Необработанные детали, полученные данным методом, имеют качество поверхности, сравнимую с деталями, полученными литьем под давлением [10].

1.1.3 Обзор метода аддитивного производства. Прямое лазерное спекание

Прямое лазерное спекание металлов (DMLS) – метод аддитивного производства изделий из металла. Технология DMLS очень схожа с технологиями SLS и SLM.

В технологии прямого спекания используется мелкодисперсный порошок. Технология заключается в том, что порошок поступает в камеру в количестве, которое необходимо для создания одного слоя. Далее, разравнивающий валик наносит слой и удаляет лишний. Затем, лазерная головка спекает частицы только что нанесенного порошка с предыдущем слоем, по контуру цифровой модели.

В качестве источника энергии для спекания металлического порошка используются оптоволоконные лазеры относительно высокой мощности порядка 200 Вт. Имеется возможность увеличить производительность за счет использования более мощных лазеров.

Технология DMLS подразумевает наличие поддержек для печатаемого изделия, само изделие должно быть закреплено на основании, а у всех выступающих элементов должны быть поддержки.

1.2 Качественная характеристика готовых изделий, произведенных методами аддитивного производства. Остаточное напряжение

В процессе создания деталей технологиями аддитивного производства в объеме материала появляются крупные градиенты температуры, а также технологические остаточные напряжения, которые приводят к повреждению формы изделия, искажению параметров объекта, как механических, так и эксплуатационных, разрушению детали во время самого производства [19]. Чтобы отработать режимы аддитивного производства, а также оптимизировать технологический

процесс следует произвести сначала построение процесса формирования изделия послойно, которое сводится к решению термоконверсионных задач с помощью большого количества различных вариантов. Остаточные напряжения — это деформация и пропорциональное ей напряжение в твердом теле при отсутствии внешних механически воздействий на него [7].

Остаточное напряжение появляется в изделии при термической обработке, механическом воздействие и других воздействиях.

Проявление остаточного напряжения при трансформации из жидкого агрегатного состояния в твердое мотивируется тем, что застывание изделия начинается с поверхности, которое сопровождается сжатиями и усадкой. Быстрое застывания внешнего слоя провоцируют возникновение внутреннего напряжения [23].

Остаточные напряжения могут быть специально созданы для получения каких-либо определенных конструктивных свойств или быть вредными. Негативные остаточные напряжения — это скрытый дефект. Для предотвращения и уменьшения остаточного напряжения осуществляется комплекс мер.

1.2.1 Причины возникновения остаточного напряжения при аддитивном производстве

При аддитивном производстве сложных металлических изделий слой металлического порошка поэтапно нагревается и охлаждается. После нанесения следующего слоя предыдущий подвергается повторному нагреву, в результате происходит сплавление двух слоев в единое целое. К окончанию печати все слои подвергались множественному изменению температуры, приводящему к появлению в детали остаточных деформаций [4].

Неравномерное нанесение порошка, а также циклическое тепловое сжатие и расширение провоцирует образование дефекта микроструктуры. На уровне

макроструктуры возможно появление дополнительных деформаций, обусловленных сложной формой детали или конструкцией опор. Эти и многие другие варианты событий могут стать причиной сбоев печати: деформации, нарушение заданных размеров, растрескивание, разломы [1].

Причинами появления дефектов, присуще для лазерного плавления, могут быть различные факторы, включая скорость перемещения и мощность лазера. Причинами технологических дефектов может быть слишком высокая или низкая скорость перемещение луча лазера и пучка электронов и слишком большая или маленькая выходная мощность лазера. При недостаточной мощности в зоне нанесения порошка фрагменты, оставшиеся нерасплавленными, образуют полости и поры произвольной формы. Если мощность слишком избыточна, появляются вкрапления газа, которые представляют собой круглые поры [33].

На процесс производства и качество выходной детали влияют множество факторов включающие в себя подачу газа, материал и другие факторы.

1.2.2 Способы минимизации остаточного напряжения при аддитивном производстве

При использовании технологии селективного лазерного плавления порошковый металл после расплавления быстро затвердевают, это приводит к усадке и появлению внутренних напряжений. Эти негативные последствия приводят к уменьшению прочности и максимально допустимой нагрузки, на которую будет рассчитано изделие. Внутренне напряжение в готовом изделии, полученном технологией селективного лазерного плавления, может значительно уменьшиться путем посттермической обработки.

Термическая обработка применяется для изменения структурно-фазового состояния материала с целью увеличения пластичности и прочности. Слишком высокая скорость прохождения лазерного луча по слою в селективном лазерном

плавлении содействует быстрому охлаждения материала и появлению нетрадиционной микроструктуры. В отдельных случаях такая микроструктура оказывается пагубное воздействие на изделие, для предотвращения негативных воздействий требуется термообработка, которая изменит структурно-фазовое состояние материала.

Количество микродефектов в структуре изделия, созданное технологией селективного лазерного плавления, можно ограничить, используя горячее изостатическое прессование (ГИП). Использование ГИП позволяет получать изделия с плотностью практически 100%. Поры существенно оказывают воздействие на механические свойства детали при циклических нагрузках. Применение ГИП заметно увеличивает усталостную прочность образцов, изготовленных методом селективного лазерного плавления, и приближает ее значение к значениям изделий, полученных традиционными методами.

Для устранения уменьшения количества дефектов используются разные схемы сканирования, т.е. обработки слоя лазером. Вместо сканирования всех слоев в одном направлении возможно применить послойное чередование направления движения лазера. Схема чередования сканирования изображена на рисунке 3

Рисунок 3 – Схемы чередования направлений движений лазера

Также может применяться метод штриховки «шахматной доски». Метод шахматной доски заключается в том, что проходы лазера производятся по принципу шахматной доски слой делится на клетки, и луч поочередно воздействует на черные и белые клетки перпендикулярно друг другу как на рисунке 4.

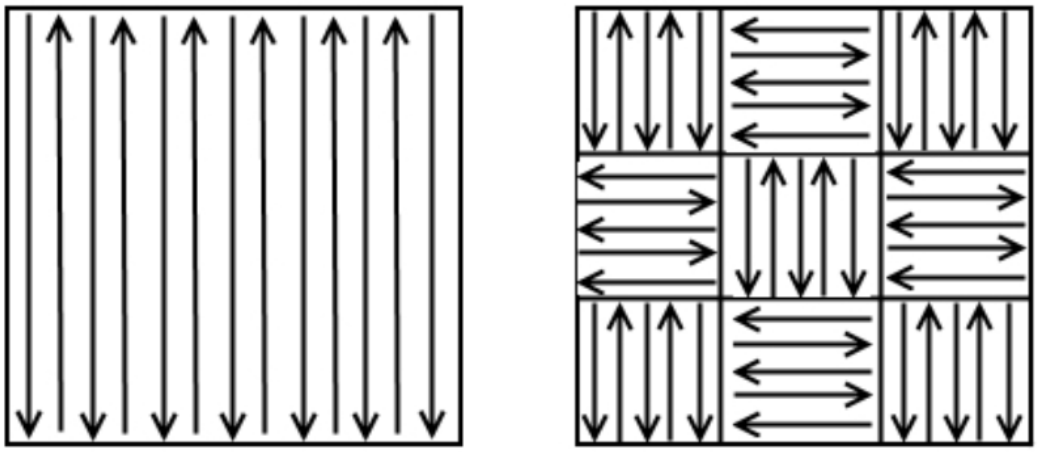

Рисунок 4 – Обычный метод сканирования и метод шахматной доски

Применение разных способов обработки способствует получению высокой плотности изделий (менее 1 % пористости). Одновременное использование нескольких методов способствует изготовлению изделия со сложной формой, при проведении контроля качества и механических испытаний, изделия по параметрам схожи с образцами, созданными путем литья.

1.2.3 Методы прогнозирования и определения остаточного напряжения при аддитивном производстве

Для определения и рассмотрения дефектов внутренней структуры изделий из большинства материалов могут использоваться томографы. Томограф чрезвычайно удобен и полезен для отработки технологий аддитивного производства. Томограф очень удобен для определения внутреннего качества изделия. Томо-

граф так же может использоваться для итогового контроля ответственных металлопорошковых изделий с повышенными требованиями герметичность и качество структуры материала.

Компьютерная томография делает возможным просмотреть внутреннюю структуру напечатанного изделия, определить наличие пор с высоким разрешением, получить объемное изображение изделия. Использование томограммы позволяет получить визуализацию сечения изделия и объемную модель в целом

Компьютерное моделирование дает возможность прогнозировать исход 3D-печати и рассчитывать выходные параметры готовой детали, тем самым уведомляет о возникновение проблем до начала печати. Последующее изменение трехмерной модели с учетом уже известных проблем обеспечивает успешное построение изделия высокой точности и качества [16, 26].

Одним из таких решений является ANSYS Additive Print.

ANSYS Additive Print – это производительный и понятный инструмент моделирования точной формы детали, получаемых в процессе 3D-печати. Этот инструмент способствует принятию обоснованных решений при решении задач аддитивного производства с использованием порошкового металла.

Additive Print имеет высокую точность прогнозирования:

- Итоговой формы изделия;
- Напряжений в каждом слое;
- Оптимального количества и расположения опор;
- Компенсации коробления (в STL-файле);
- Позволяет понять, как именно изделие будет деформироваться в процессе изготовления.
- Визуально показывает, как прогнозируемое коробление и остаточное напряжения отразятся на конечное изделие и поможет определить наиболее оптимальную ориентацию изделия и способы размещения опор.

1.2.3.1 Прогнозирование остаточных напряжений и деформаций в программе Ansys Additive Print

Первый этап при использовании программы Ansys Additive Print является подготовка модели к прогнозированию остаточных напряжения и деформаций

Для прогнозирования усадки и деформаций будет использоваться модель сложной стоматологической конструкции из металла. Модель сложной стоматологической конструкции изображена на рисунке 5.

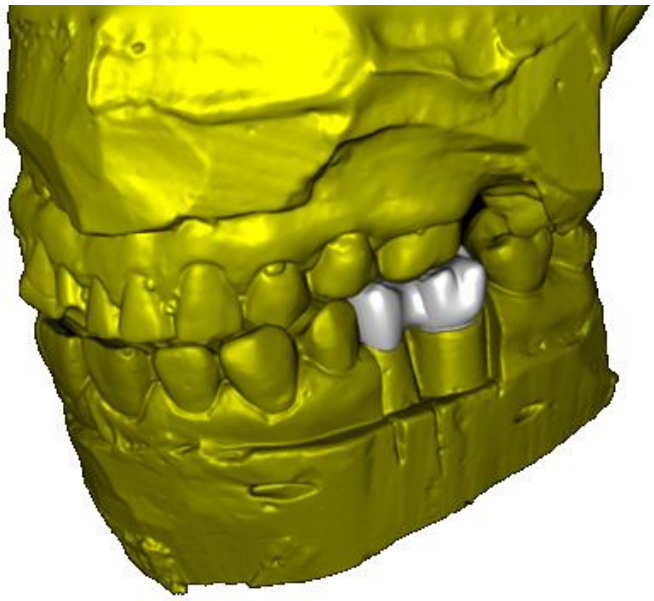

Рисунок 5 – Модель стоматологической конструкции

Для прогнозирования остаточного напряжений и деформаций в программном комплексе Ansys Additive Print потребуется подготовить модель, а именно уменьшить количество треугольников, то есть упростить модель. Максимальное количество треугольников способное обработать Ansys Additive Print является 50 тысяч. Упрощение модели будет производиться в программе Blender имеющая свободный доступ.

В первую очередь нужно импортировать модель формата STL в Blender как на рисунке 6.

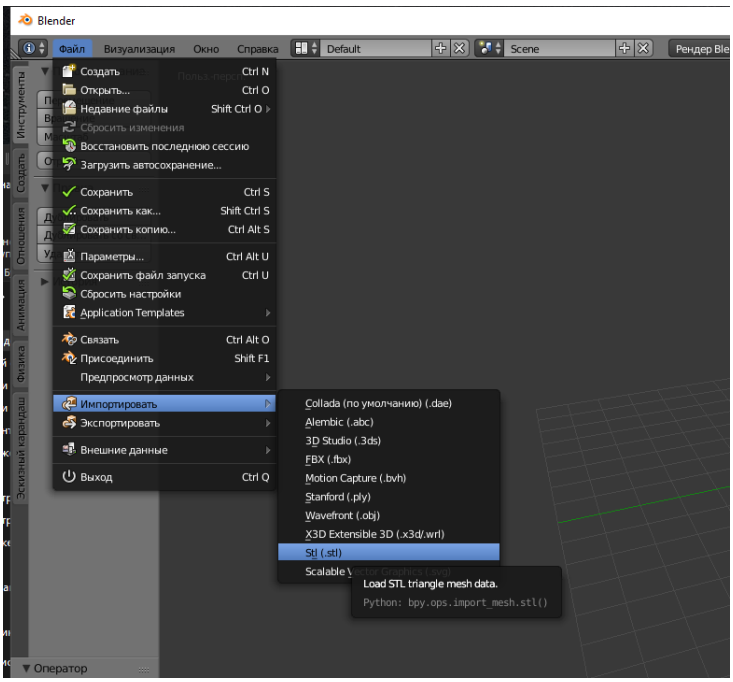

Рисунок 6 – Импортирование модели в Blender

Следующим шагом требуется применить модификатор аппроксимации к загруженной модели и подобрать требуемый коэффициент упрощения. На рисунке 7 изображено упрощение модели в Blender. В результате модель упрощается с 180 тысяч треугольников до 38 тысяч треугольников.

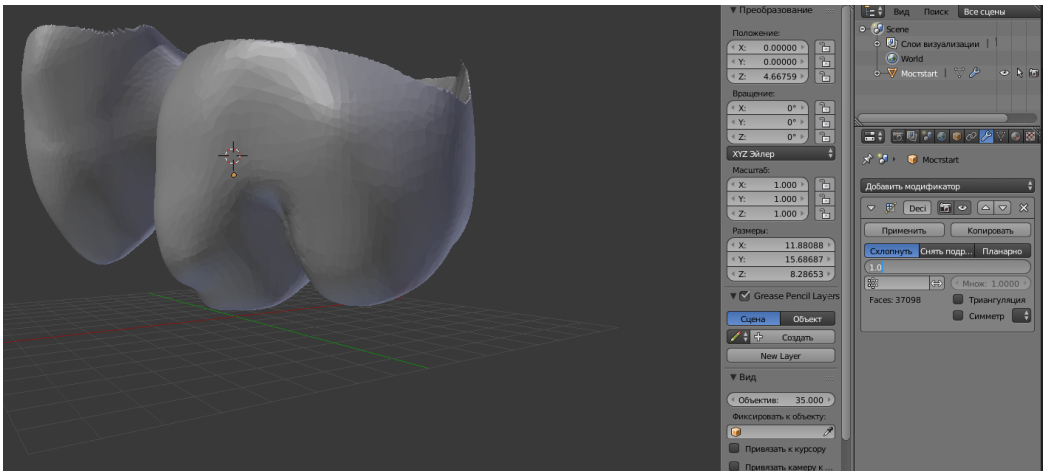

Рисунок 7 – Аппроксимация модели в Blender

Далее требуется экспортировать модель в формат STL и загрузить в раздел Parts программы Ansys Additive. На рисунке 8 изображен интерфейс раздела Parts.

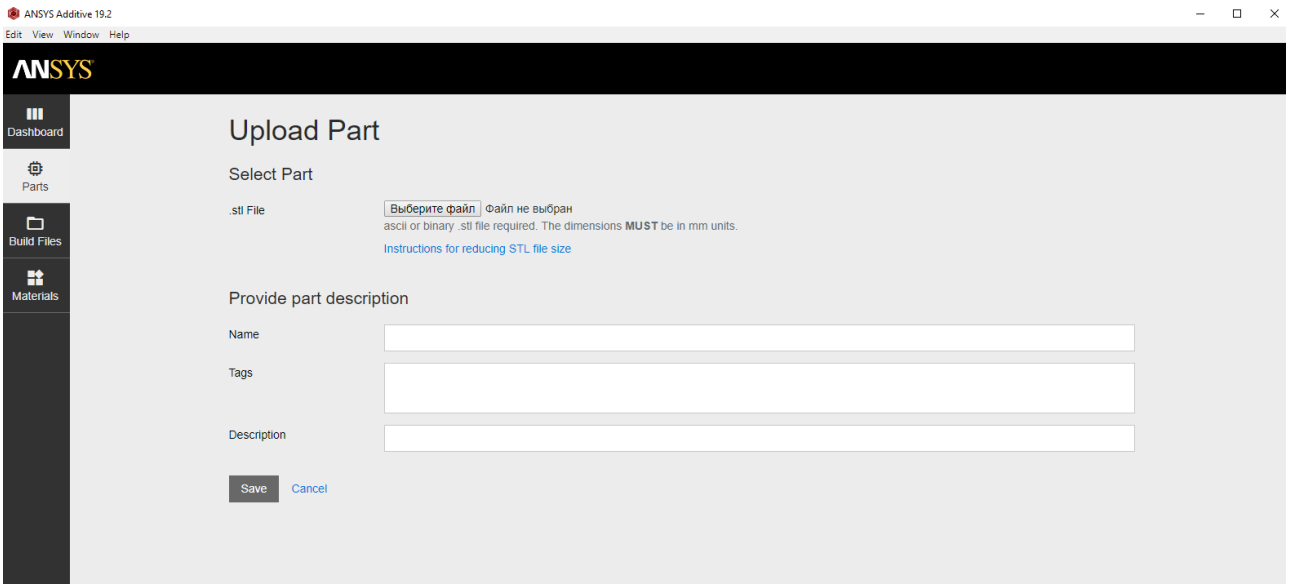

Рисунок 8 – Интерфейс раздела Parts в ANSYS Additive

Этап подготовки модели завершён и теперь можно перейти непосредственно к прогнозированию остаточных напряжений и деформаций.

После загрузки модели переходим в раздел Dashboard. В разделе Dashboard в поле Draft Simulation нажимаем New, в выпавшем поле выбираем требуемую функцию прогнозирования. На рисунке 9 изображен интерфейс программы Ansys Additive в разделе Dashboard.

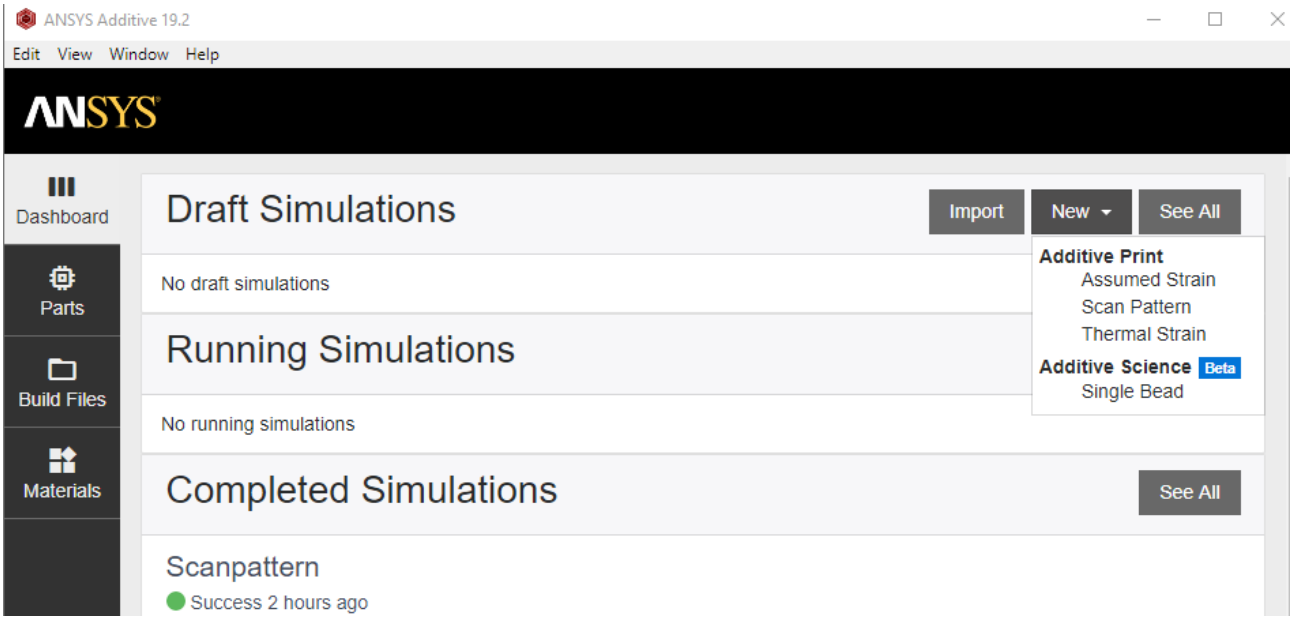

Рисунок 9 – Интерфейс программы Ansys Additive в разделе Dashboard

На выбор предоставляется три сценария расчетов Assumed Strain, Scan Pattern, Thermal Strain.

Assumed Strain прогнозирует возможные деформации, Scan Pattern подбирает оптимальные схемы сканирования (перемещение лазера), Thermal Strain прогнозирует деформации вызванные циклическими сменами температуры изготавливаемого изделия.

После выбора сценария требуется заполнить параметры печати, материала, поддержек и выбрать файлы вывода.

В поле geometry selection выбираем исследуемую модель и размер вокселя, размер вокселя влияет на точность спрогнозированной модели. На рисунке 10 изображен интерфейс выбора модели.

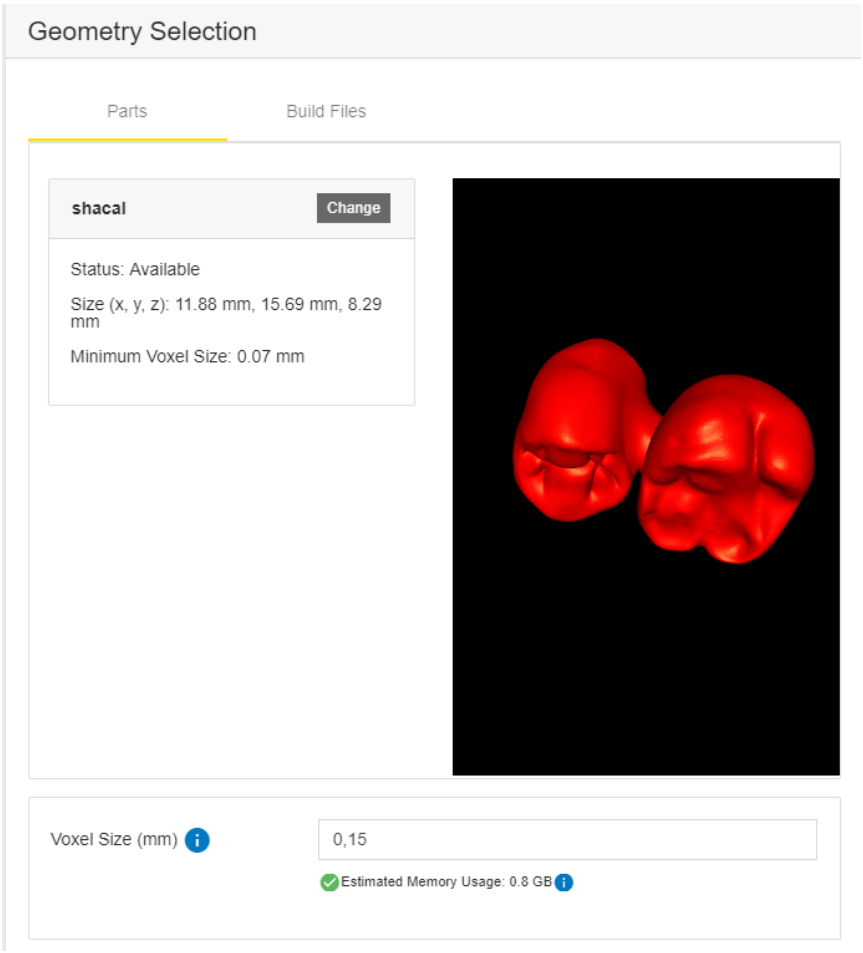

Рисунок 10 – Интерфейс поля Geometry Selection в Ansys Additive

В следующем окне Material Configuration выбираем металл, из которого будет производится изделие. На рисунке 11 изображен интерфейс выбора параметров материала печати.

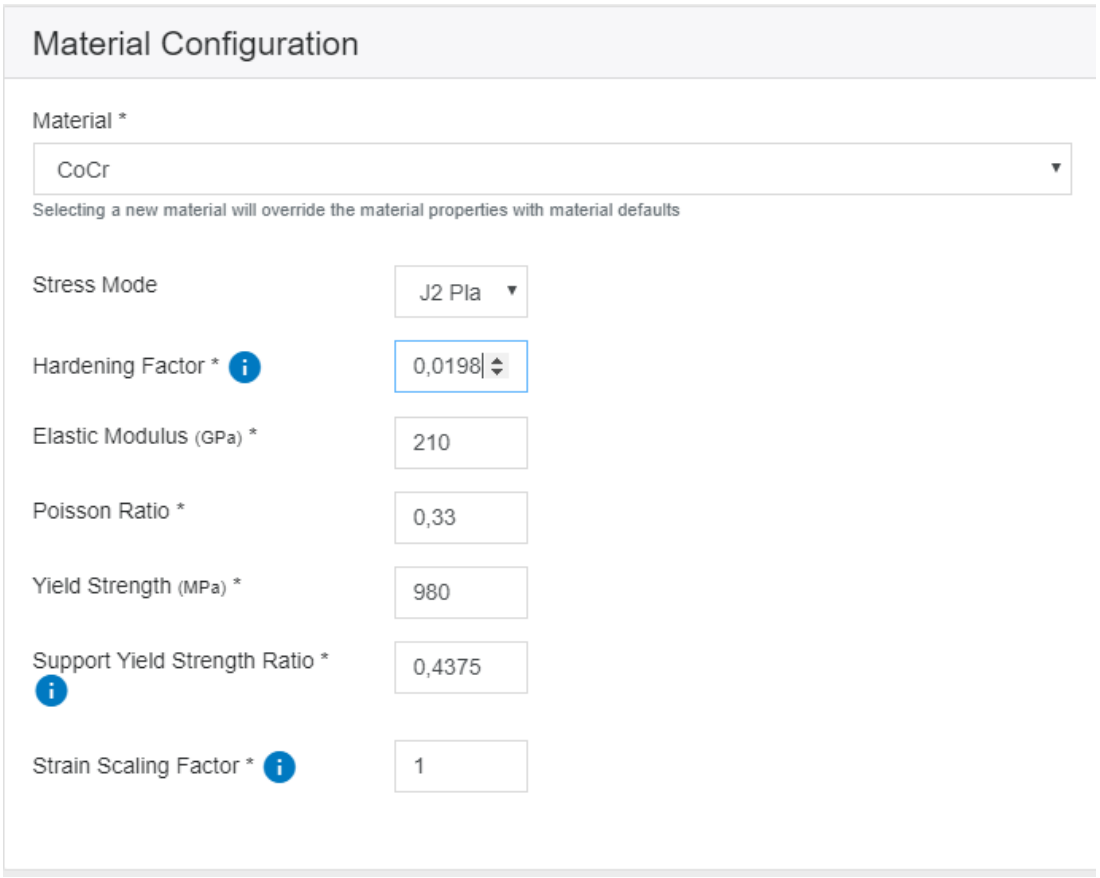

Рисунок 11 – Интерфейс поля Geometry Selection в Ansys Additive

Также есть возможность задать свои параметры материала изготовления.

Следующим шагом требуется указать параметры построения поддерживающих опор такие как максимальная и минимальная толщина стенок, минимальное высота опоры и т.п. На рисунке 12 изображен интерфейс выбора параметров поддержек и опор.

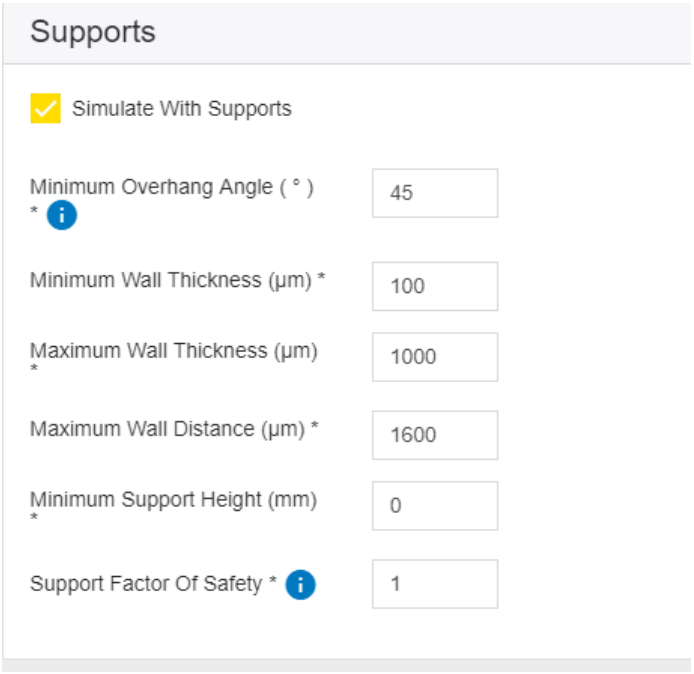

Рисунок 12 – Интерфейс поля Geometry Selection в Ansys Additive

Далее указываем требуемые файлы вывода как на рисунке 13 и нажимаем кнопку Start

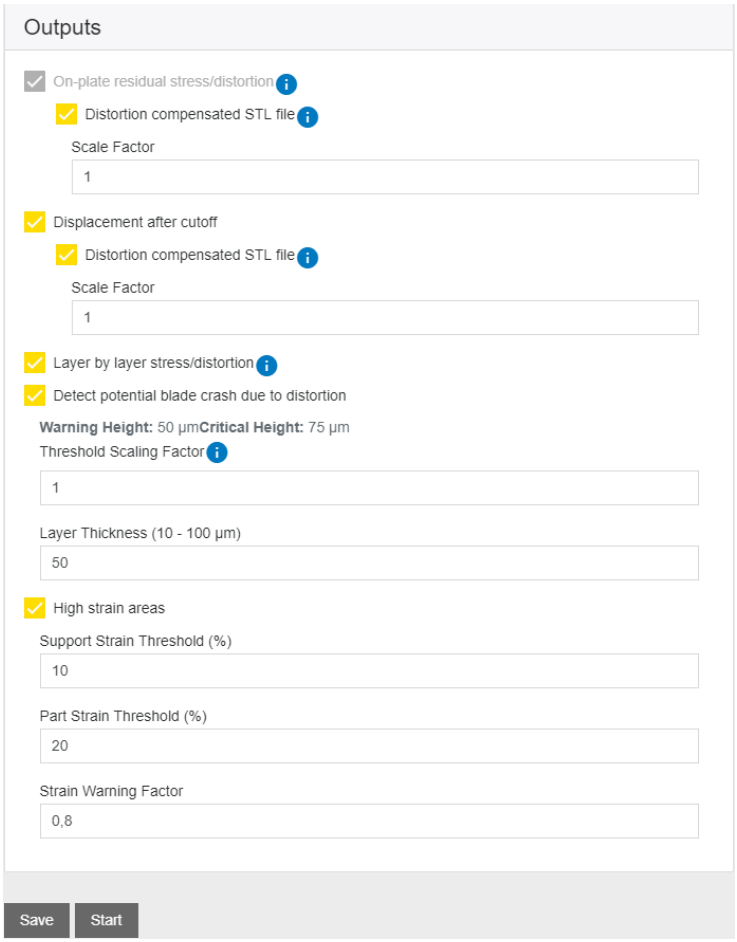

Рисунок 13 – Интерфейс поля Outputs в Ansys Additive

После нажатия на кнопку Start открывается окно прогнозирования, в котором отображаются данные проекта, статус выполнения. После выполнения расчетов в поле Output Files станет доступен список файлов вывода как показано на рисунке 14.

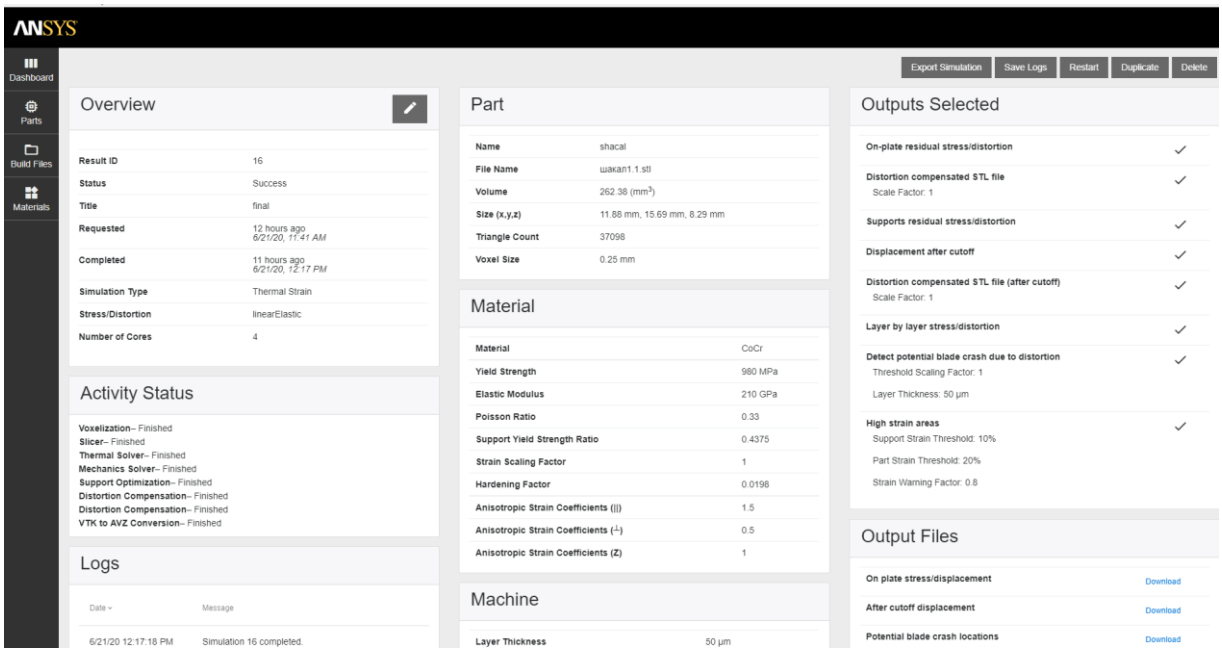

Рисунок 14 – Интерфейс окна прогнозирования в Ansys Additive

В Output Files есть целый список выходных данных, которые могут найти свое применение при оптимизации модели, помимо этого программа ANSYS Additive предлагает свою оптимизацию модели. Прямо Ansys посмотреть внутренние напряжения модели. На рисунке 15 изображён пример прогнозируемого дефекта.

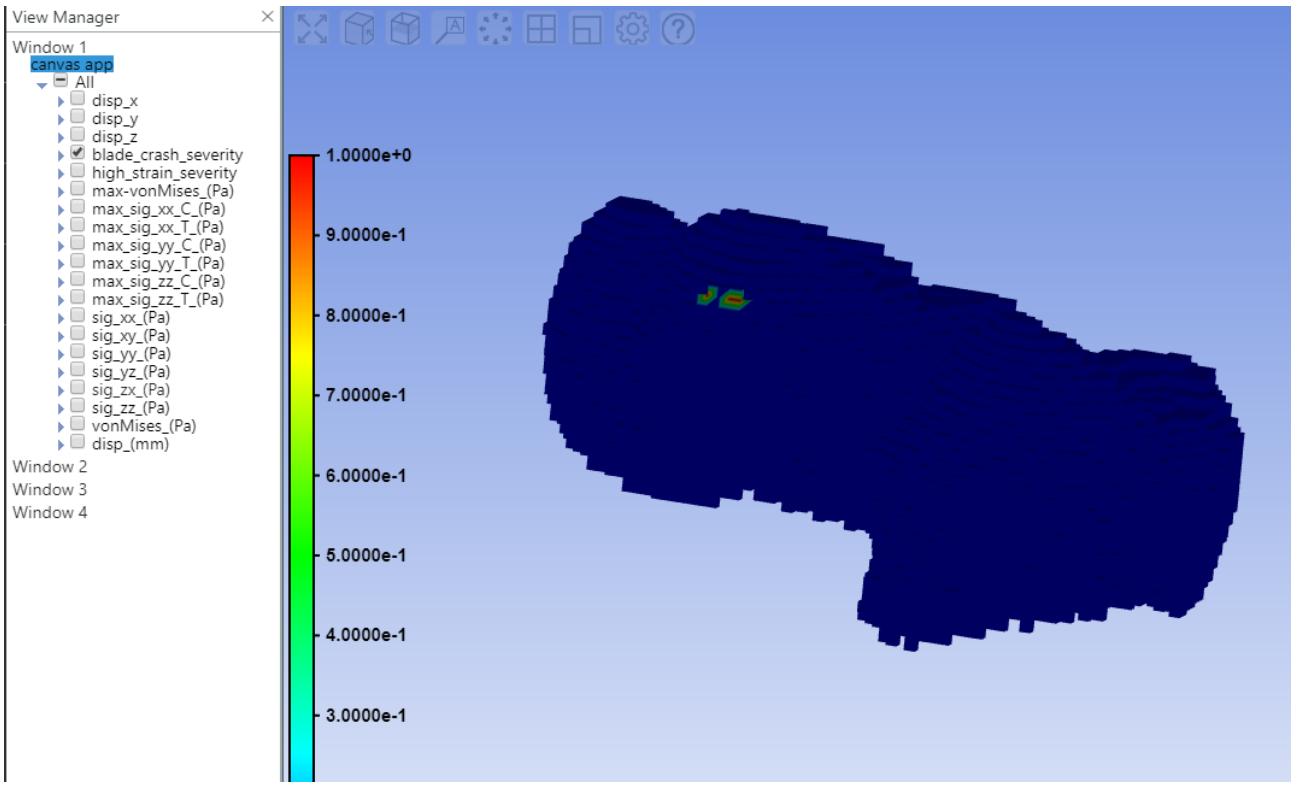

Рисунок 15 – Интерфейс просмотра выходного файла в ANSYS Additive

Кроме просмотра дефектов внешнего слоя в ANSYS Additive можно просматривать и внутренние дефекты, посредство просмотра сечения модели как изображено на рисунке 16.

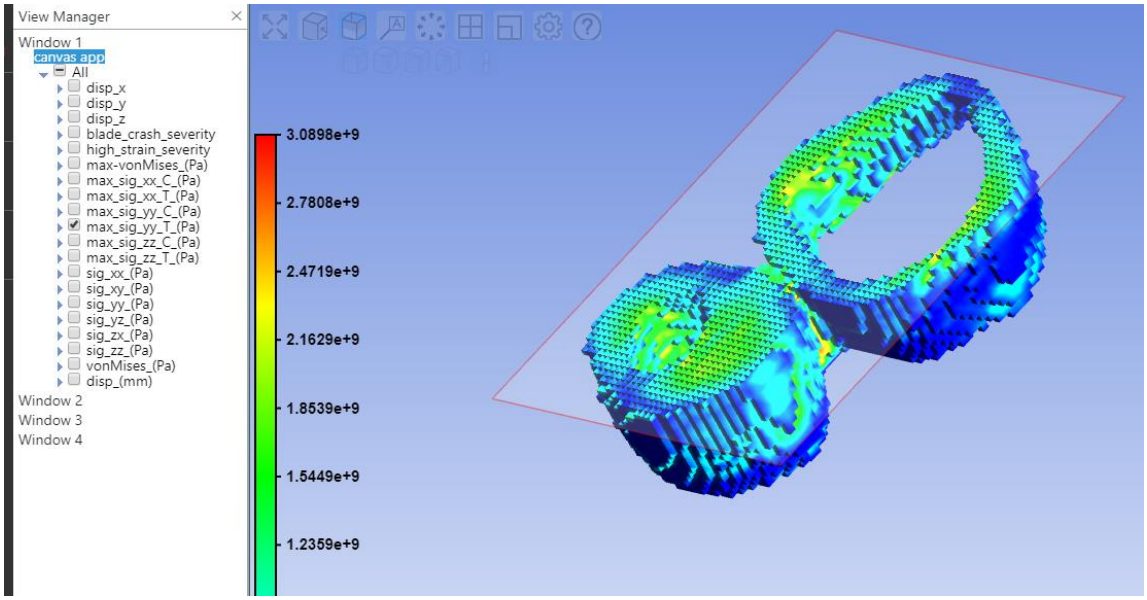

Рисунок 16 – Интерфейс просмотра сечения модели в ANSYS Additive

Помимо оптимизации начальной модели для печати, программа способна моделировать поддержки и опоры для изделия и вычисляет нагрузку на них подбирая оптимальные параметры. Пример опоры для печати изображена на рисунке 17.

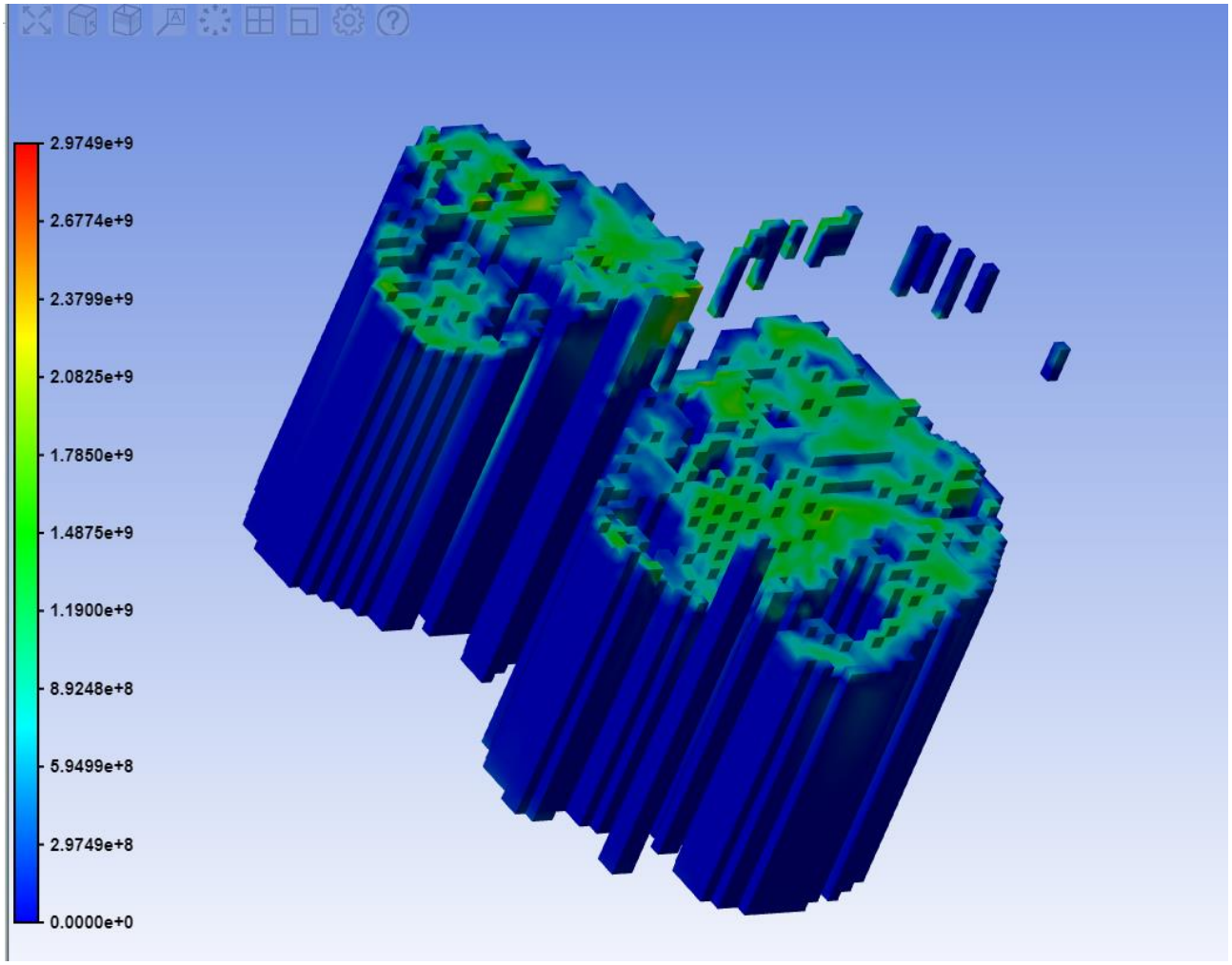

Рисунок 17 – Пример опор и поддержек

В итоге работы с ANSYS Additive были получены файлы параметров изделия и визуальные трехмерные модели с выделенными проблемными зонами. На основе полученных данных можно производить тестирование параметров печати изделия при этом не тратить лишний материал и экономить время и найти оптимальные параметры печати под каждое изделие.

1.3 Математическая модель процесса аддитивного производства и расчета остаточного напряжения

Для того чтобы программный комплекс для расчета остаточных напряжений и деформаций выполнял свои основные функции требуется смоделировать несколько физических процессов, к которым относится теплоперенос и распределение напряжений и деформаций. Для построения полей остаточного напряжения, вызванного циклическим нагреванием-остыванием модели теплоперенос должен рассчитываться каждую итерацию в каждой ячейке (вокселе). Математическая модель должна учитывать геометрию производимой модели на каждом шаге. Математическая модель должна учитывать множества параметров аддитивного производства: температуру источника, коэффициент теплопроводности материала, размер вокселя, скорость сканирования модели, время обработки одного вокселя и т.п.

1.3.1 Численная реализация процесса аддитивного производства металлоконструкций

Процесс аддитивного производства селективной лазерной плавки представляет собой непрерывный процесс нагрева мелкодисперсного металлического порошка на рабочей области в конкретной точке, то есть у нас есть несколько основных представлений материала которые будут иметь разный параметры и разный коэффициент теплопроводности. К основным типам материла будут относиться: мелкодисперсный порошок, сплавленный порошок, материал несплавливаемой подложки, воздух или инертная среда в которой обычно протекает процесс производства. Рабочая камера представляет собой замкнутую систему [12, 22].

Для моделирования теплопереноса будем использовать уравнение теплопроводности [24]:

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\mathrm{K}\frac{\partial T}{\partial x}\right) + q(t, x, T), \quad (1)$$

В этом уравнение описана нелинейная зависимость коэффициента K от T, где K – коэффициент теплопроводности, T – температура или энергия, x – пространственная координата, t – участок временного промежутка, q – источник энергии, температуры которой получает конкретный участок в зависимости от координаты и времени [24].

Для создания компьютерной математической модели требуется представить формулу (1) в численной форме.

$$T_i^{n+1} = T_i^n + \frac{\tau_n}{h^2}\left(\mathrm{K}^n_{i+\frac{1}{2}}(T^n_{i+1} - T^n_i) - \mathrm{K}^n_{i-\frac{1}{2}}\left(T^n_i - T^n_{i-1,}\right)\right) + q(n, i, T), \quad (2)$$

Где n – текущий время (номер итерации), i – пространственная координата, h – шаг пространственной координаты, τ_n- шаг по времени.

$$\mathrm{K}^n_{i+\frac{1}{2}} = \mathrm{K}\left(T_{i\pm\frac{1}{2}}\right), \quad (3)$$

где

$$T_{i\pm\frac{1}{2}} = \frac{T_{i\pm\frac{1}{2}} + T_i}{2}, \quad (4)$$

Шаг по времени τ_n для повышения сходимости системы должен рассчитываться динамически.

$$\tau_n = \mathrm{K} * \frac{h^2}{2 * K_{max}}, \quad (5)$$

где K_{max} – это максимальное значение коэффициента теплопроводности в сетке

$$K_{max} = MAX_i\big(K(T_i)\big), \quad (6)$$

где MAX – функция нахождения максимального элемента.

Формула (2) позволяет рассчитать перенос тепла в одномерном пространстве, но по условиям задачи требуется моделировать теплоперенос в трехмерном пространстве, поэтому добавим еще два измерения и преобразуем формулы (2-6) для трехмерного пространства.

$$T_{i,j,k}^{n+1} = T_{i,j,k}^{n} +$$
$$+\frac{\tau_n}{h^2}\left\{\begin{array}{c} \mathrm{K}_{i+\frac{1}{2},j,k}^{n}\left(T_{i+1,j,k}^{n} - T_{i,j,k}^{n}\right) - \mathrm{K}_{i-\frac{1}{2},j,k}^{n}\left(T_{i,j,k}^{n} - T_{i-1,j,k}^{n}\right) + \\ + \mathrm{K}_{i,j+\frac{1}{2},k}^{n}\left(T_{i,j+1,k}^{n} - T_{i,j,k}^{n}\right) - \mathrm{K}_{i,j+\frac{1}{2},k}^{n}\left(T_{i,j,k}^{n} - T_{i,j-1,k}^{n}\right) + \\ + \mathrm{K}_{i,j,k+\frac{1}{2}}^{n}\left(T_{i,j,k+1}^{n} - T_{i,j,k}^{n}\right) - \mathrm{K}_{i,j,k+\frac{1}{2}}^{n}\left(T_{i,j,k}^{n} - T_{i,j,k-1}^{n}\right) \end{array}\right\} +$$
$$+q(n,i,j,k,T), \quad (7)$$

где

$$\mathrm{K}_{i+\frac{1}{2},j,k}^{n} = \mathrm{K}\left(T_{i\pm\frac{1}{2},j,k}\right), \quad (8)$$

$$\mathrm{K}_{i,j+\frac{1}{2},k}^{n} = \mathrm{K}\left(T_{i,j+\frac{1}{2},k}\right), \quad (9)$$

$$\mathrm{K}_{i,j,k+\frac{1}{2}}^{n} = \mathrm{K}\left(T_{i,j,k+\frac{1}{2}}\right), \quad (10)$$

$$T_{i\pm\frac{1}{2},j,k} = \frac{T_{i\pm\frac{1}{2},j,k} + T_{i,j,k}}{2}, \quad (11)$$

$$\tau_n = \mathrm{K} * \frac{h^2}{2 * K_{max}}, \quad (12)$$

$$K_{max} = MAX_{i,j,k}\left(K\left(T_{i,j,k}\right)\right), \quad (13)$$

где i, j, k – пространственные координаты. Теперь все готово для построения компьютерной модели. Уравнения (7-13), позволят построить модель теплопереноса в трехмерном пространстве. В технологиях аддитивного производства металлических изделий, а именно в технологии селективного лазерного плавления под источником нагрева понимается лазер, который нагревает металлический мелкодисперсный порошок. Моделировать процесс импульса лазера процесс трудоемкий и малоэффективный. Источник нагрева будет находится прямо в участке обработки с заданной координатой, так будет наиболее оптимально моделироваться воздействие лазера на порошковый слой [3, 9].

1.3.2 Численная реализация расчета остаточного напряжения и деформаций металлоконструкций

Составим уравнения для осевых деформаций вокселя, где $\xi^x_{i,j,k}$– деформация по оси x, вокселе с координатами i, j, k; $\hat{\alpha}_{i,j,k}$ – тепловое расширение; h – размер вокселя; $\alpha_0(T)$ – коэффициент теплового расширения; $\Delta T_{i,j,k}$ – изменение температуры; $T_{i,j,k}$ – текущая температура; T_0 – начальная температура системы [12].

$$\xi^x_{i,j,k} = -h^2 \frac{\partial(\alpha_0 \Delta T)}{\partial x} = -h \frac{\hat{\alpha}_{i+1,j,k} - \hat{\alpha}_{i-1,j,k}}{2} \quad (14)$$

$$\xi^y_{i,j,k} = -h^2 \frac{\partial(\alpha_0 \Delta T)}{\partial y} = -h \frac{\hat{\alpha}_{i,j+1,k} - \hat{\alpha}_{i,j-1,k}}{2} \quad (15)$$

$$\xi^z_{i,j,k} = -h^2 \frac{\partial(\alpha_0 \Delta T)}{\partial z} = -h \frac{\hat{\alpha}_{i,j,k+1} - \hat{\alpha}_{i,j,k-1}}{2} \quad (16)$$

$$\hat{\alpha}_{i,j,k} = \alpha_0(T) * \Delta T_{i,j,k} \quad (17)$$

$$\Delta T_{i,j,k} = T_{i,j,k} - T_0 \quad (18)$$

ξ_{fatal} – показатель деформации при которой происходит перестроение кристаллической решетки. Если значение достигает ξ_{fatal}, то при остывании вокселя значение деформации не возвращается в исходное состояние при остывании.

Запишем уравнения тензорного представления деформаций [24].

$$u_{i,j,k} \rightarrow u^{xx}, u^{xy} = u^{yx}, u^{xz} = u^{zx}, u^{yy}, u^{yz} = u^{zy}, u^{zz} \quad (19)$$

$$u^{xx} = \frac{\partial \xi^x_{i,j,k}}{\partial x} = \frac{\xi^x_{i+1,j,k} - \xi^x_{i-1,j,k}}{2h} \quad (20)$$

$$u^{yy} = \frac{\partial \xi^y_{i,j,k}}{\partial y} = \frac{\xi^y_{i,j+1,k} - \xi^y_{i,j-1,k}}{2h} \quad (21)$$

$$u^{zz} = \frac{\partial \xi^z_{i,j,k}}{\partial z} = \frac{\xi^z_{i,j,k+1} - \xi^z_{i,j,k-1}}{2h} \quad (22)$$

$$u^{xy} = u^{yx} = \frac{1}{2}\left(\frac{\partial \xi^x_{i,j,k}}{\partial y} + \frac{\partial \xi^y_{i,j,k}}{\partial x}\right) =$$

$$= \frac{1}{4h}\left(\xi^x_{i,j+1,k} - \xi^x_{i,j-1,k} + \xi^y_{i+1,j,k} - \xi^y_{i-1,j,k}\right) \quad (23)$$

$$u^{xz} = u^{xy} = \frac{1}{2}\left(\frac{\partial \xi^x_{i,j,k}}{\partial z} + \frac{\partial \xi^z_{i,j,k}}{\partial x}\right) =$$

$$= \frac{1}{4h}\left(\xi^x_{i,j,k+1} - \xi^x_{i,j,k-1} + \xi^z_{i+1,j,k} - \xi^z_{i-1,j,k}\right) \quad (24)$$

$$u^{yz} = u^{zy} = \frac{1}{2}\left(\frac{\partial \xi^y_{i,j,k}}{\partial z} + \frac{\partial \xi^z_{i,j,k}}{\partial y}\right) =$$

$$= \frac{1}{4h}\left(\xi^y_{i,j,k+1} - \xi^y_{i,j,k-1} + \xi^z_{i,j+1,k} - \xi^z_{i,j-1,k}\right) \quad (25)$$

2 Разработка информационной модели программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций

Проектирование будущего продукта (приложения, программного комплекса) один из важнейших этапов создания конечного продукта для потребителя. На стадии проектирование можно спрогнозировать ошибки и сложности при разработке, проектирование способствует минимизировать вопросы, которые могут возникнуть при разработке. На стадии проектирования формируется конечное представление о продукте и предполагаемые его векторы развития. Проектирование позволяет предсказать предполагаемое время на разработку и составить график разработки. Этап проектирования важен как для разработчиков, так и для заказчиков.

2.1 Назначение и цели создания программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций

На этом этапе производится разработка информационной модели программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций. Разрабатываем программный комплекс включает себя модуль для расчета остаточного напряжения и деформаций металлоконструкций, модуль представления расчетов в воксельном виде с выделенными участками подверженными остаточному напряжения и деформациям, модуль визуализации модели в интерфейсе программы

Исходными данными для программного комплекса является STL файл, который содержит модель изделия, так же входными данными являются параметры печати и настройки принтера или станка.

2.2 Планирование процесса реализации программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций

Диаграмма Ганта — это вид столбчатых диаграмм с помощью, которой графически представляется план и график работ по проекту разработки. Диаграмма Ганта является одним из самых распространённых методов распределения временных ресурсов и планирования проекта. Диаграмма Ганта используется в приложениях по управлению проектами [5, 19]. Работа по проектированию происходила в программе Ganttpro.

Создание программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций было разделено на 5 задач, которые поделены на соответствующие подзадачи:

1) подготовка к разработке программного комплекса;
2) разработка информационной модели программного комплекса;
3) составление пояснительной записки;
4) разработка программного комплекса;
5) создание пояснительной записки;

Графическое представление основных задач изображено на рисунке 18

Рисунок 18 – Основные задачи разработки программного комплекса

Первая основная задача «Подготовка к разработке программного комплекса» в свою очередь делится на подзадачи:

1) обсуждение темы с научным руководителем;
2) создание листа задания научным руководителем;

а) определение целей и задач;

б) составление списка рекомендуемой литературы;

3) изучение предметной области;

а) изучение соответствующих источников литературы;

б) изучение аналогов;

в) определения возможностей программного комплекса;

г) определение модулей программного комплекса;

Графическое представление задачи «Подготовка к разработке программного комплекса» изображено на рисунке 19.

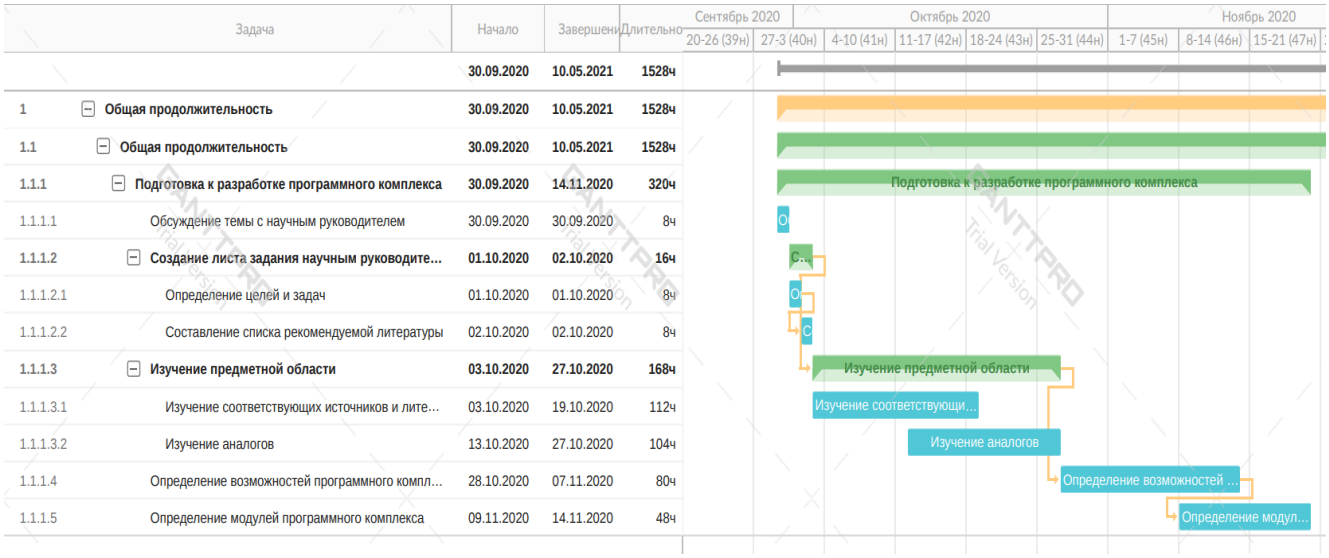

Рисунок 19 – Графическое представление задачи «Подготовка к разработке программного комплекса»

Вторая основная задача «Разработка информационной модели» в свою очередь делится на подзадачи:

1) разработка диаграммы ганта;

а) изучение нормативной документации;

б) декомпозиция задач и подзадач;

в) определение длительности выполнения задач;

г) определение условий планирования задач;

д) определение вех;

2) создание диаграммы idef0;
    а) изучение нормативной документации;
    б) создание контекстной диаграммы;
    в) выделение подпроцессов;
    г) декомпозиция процессов;
    д) определение последовательности функций;
    е) определение взаимосвязи элементов системы;
3) создание диаграммы dfd;
    а) изучение нормативной документации;
    б) создание контекстной диаграммы;
    в) декомпозиция процессов;
    г) декомпозиция потоков данных в системе;
4) создание диаграммы вариантов использования uml;
    а) изучение нормативной документации;
    б) разработка диаграммы;
5) создание диаграммы состояний uml;
    а) изучение нормативной документации;
    б) разработка диаграммы;
6) создание диаграммы классов uml;
    а) изучение нормативной документации;
    б) определение возможных классов ооп;
    в) определение взаимосвязей между классами;
    г) разработка диаграммы.

Графическое представление задачи «Разработка информационной модели» изображено на рисунке 20

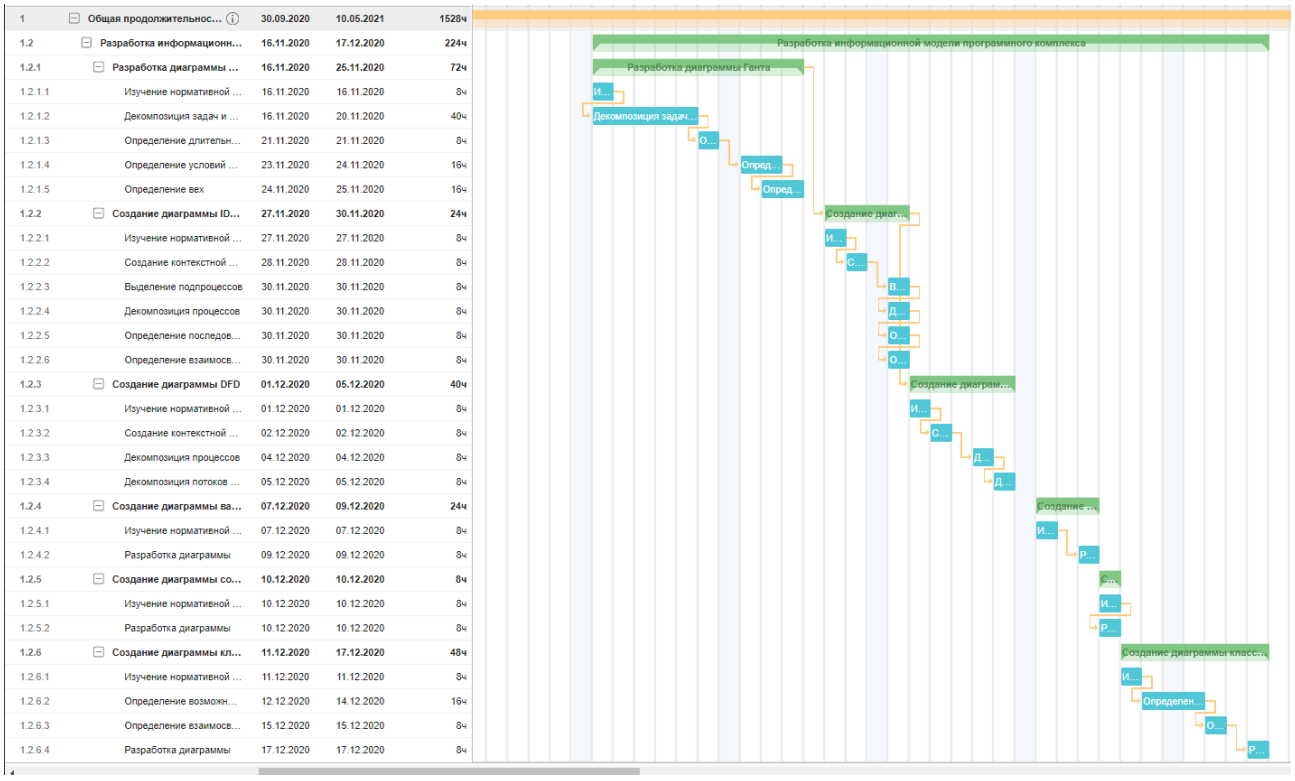

Рисунок 20 – Графическое представление задачи «Разработка информационной модели»

Следующая основная задача «Составление пояснительной записки НИР» в свою очередь делится на подзадачи:

1) составление пояснительной записки;
2) создание презентации;
3) сбор подписей;
4) прохождение нормконтроля.

Графическое представление задачи «Составление пояснительной записки НИР» изображено на рисунке 21.

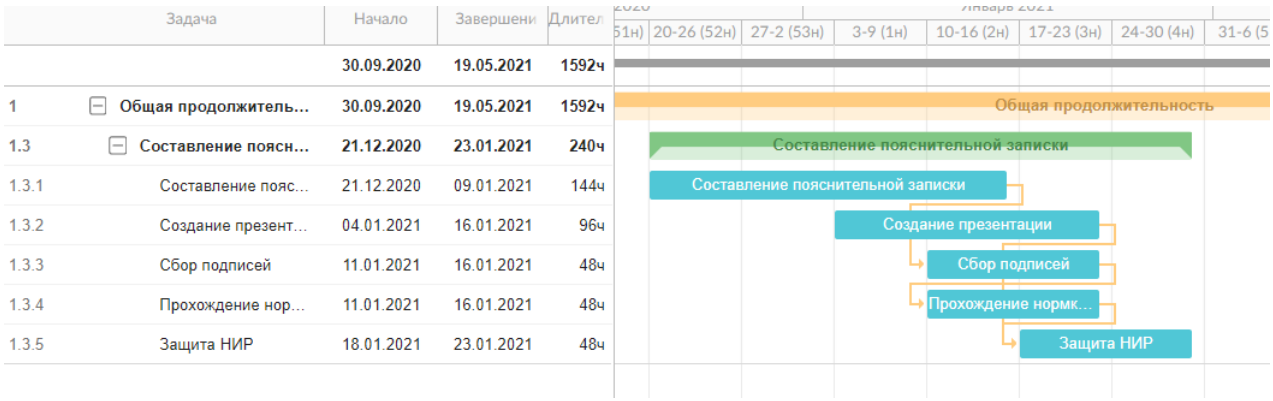

Рисунок 21 – Графическое представление задачи «Составление пояснительной записки НИР»

Следующая основная задача «Разработка программного комплекса» делится на подзадачи:

1) разработка программного комплекса;
   а) разработка пользовательского интерфейса;
      – разработка дизайна;
      – добавление функциональных элементов программы;
      – проверка работоспособности функциональных элементов;
      – внесение правок в разработанный модуль;
   б) разработка модуля обработки входной модели;
      – разработка алгоритма импорта модели;
      – реализация импорта модели;
      – разработка алгоритма вокселизации модели;
      – реализация алгоритма вокселизации модели;
   в) разработка модуля визуализации модели;
      – обработка загруженных моделей;
      – построение модели на 3d сцене;
      – внесение правок в разработанный модуль;
   г) разработка модуля для расчета остаточного напряжения и деформаций металлоконструкций;

– реализация расчета остаточного напряжения и деформаций;
– реализация графического представления расчетов;
– реализация управления выводом результирующей модели;
– внесение правок в разработанный модуль.

Графическое представление задачи «Разработка программного комплекса» изображено на рисунке 22.

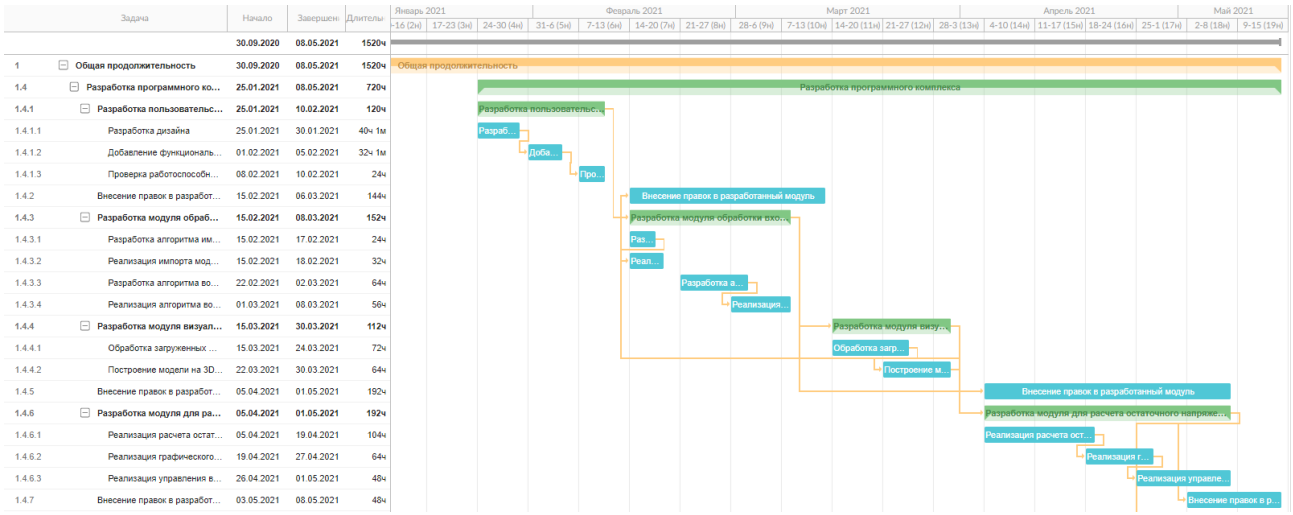

Рисунок 22 – Графическое представление задачи «Разработка программного комплекса»

2.3 Функциональное моделирование программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций

Методология функционального моделирования IDEF0 – это система принципов, положений и методов описания системы в целом как множества взаимозависимых действий. Функциональная точка зрения позволяет четко отделить аспекты назначения системы от аспектов ее физической реализации [2].

Главной целью методики IDEF0 является создание функциональной схемы системы, описывающей подробно все необходимые для работы процессы.

На рисунке 23 показана начальная диаграмма модели IDEF0 в Ramus Educational.

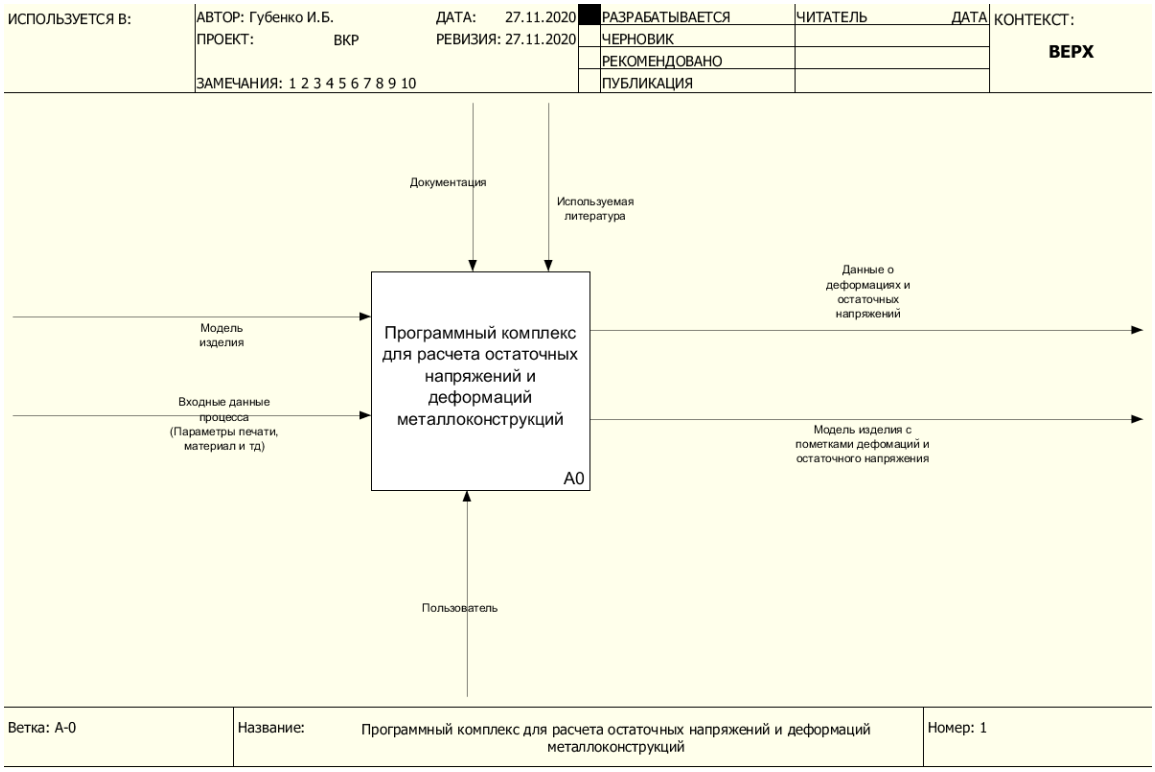

Рисунок 23 – Начальная диаграмма IDEF0

Центром диаграммы A0 является «Программный комплекс для расчета остаточных напряжений и деформаций металлоконструкций».

Компоненты функциональной модели:

1) входная стрелка: входные данные процесса (параметры печати, материал, температура и т.п.), модель изделия;
2) стрелка управления: документация и используемая литература;
3) выходная стрелка: данные о деформациях и остаточных напряжений, модель изделия с пометками деформаций и остаточного напряжения;
4) стрелка механизма использования: пользователь.

На рисунке 24 представлена декомпозиция главного процесса.

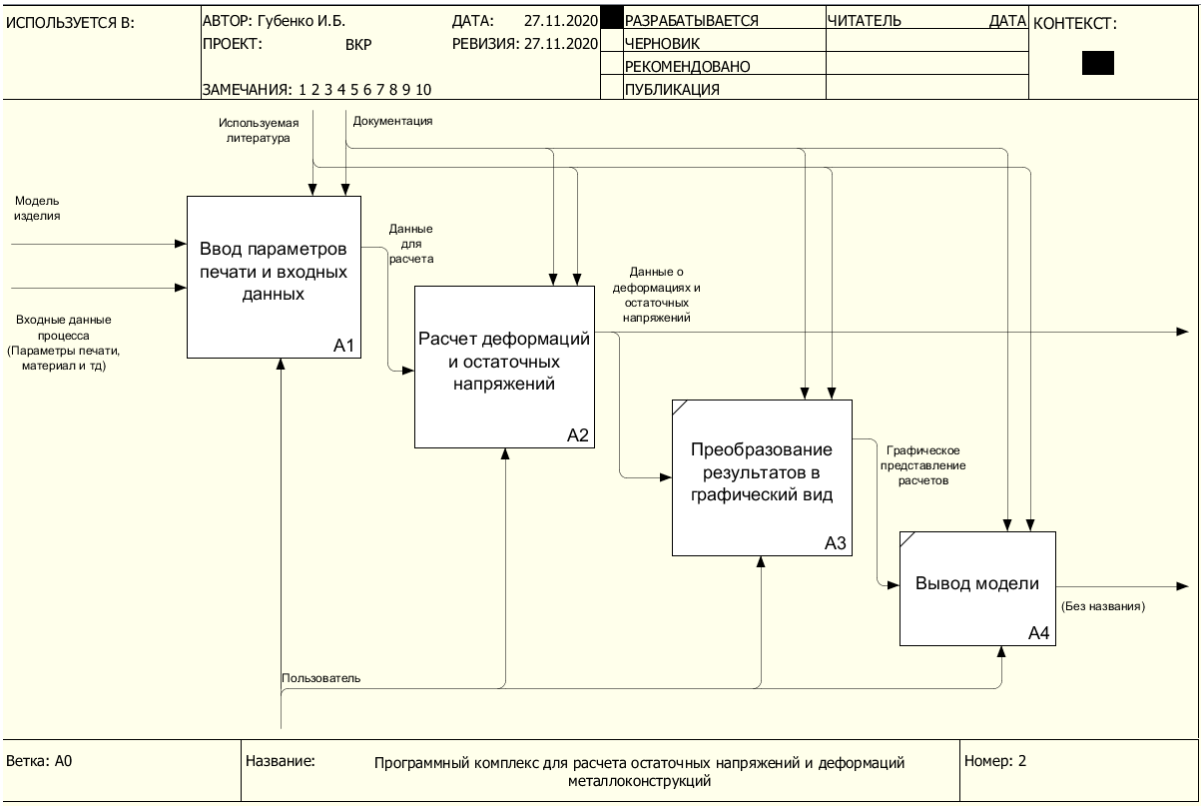

Рисунок 24 – Декомпозиция основного процесса

Декомпозиция основного процесса состоит из:

1) ввод параметров печати и входных данных;
2) расчет деформаций и остаточных напряжений;
3) преобразование результатов в графический вид;
4) вывод модели.

На рисунке 25 представлена декомпозиция процесса «Ввод параметров печати и входных данных».

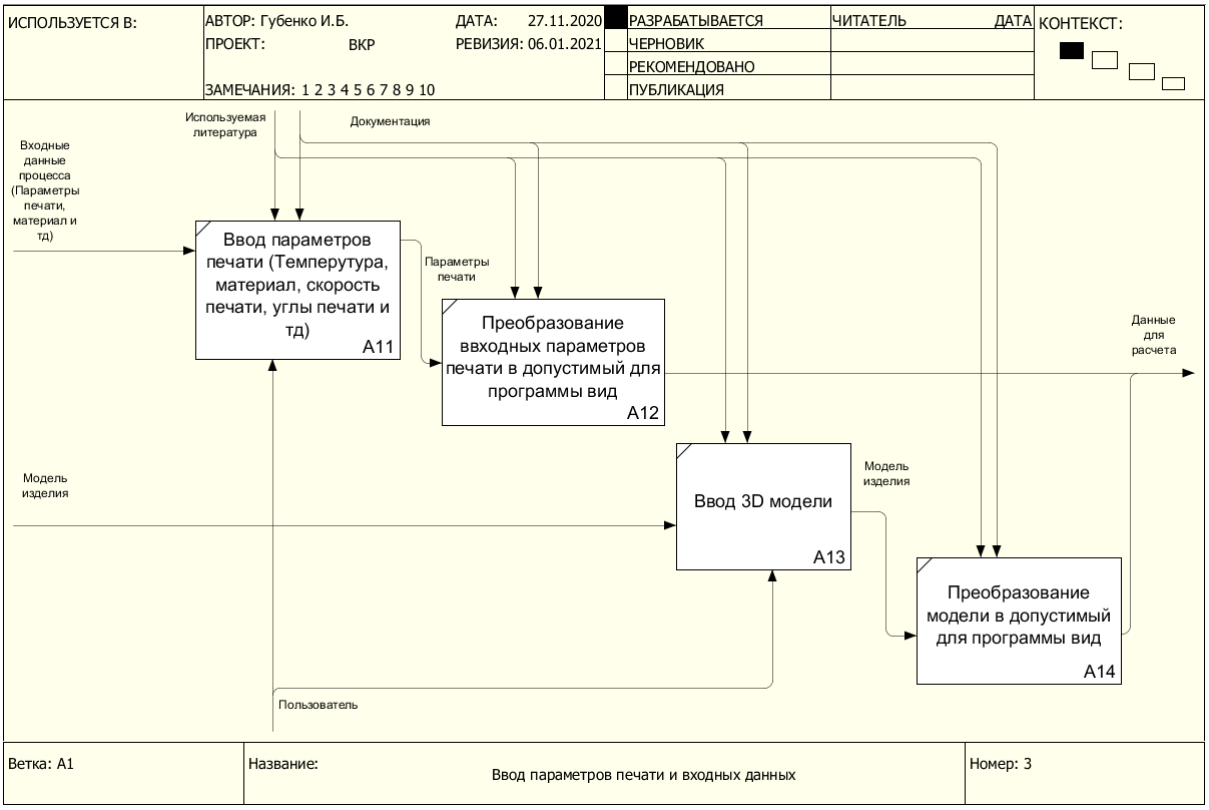

Рисунок 25 – Декомпозиция процесса «Ввод параметров печати и входных данных»

Декомпозиция основного процесса состоит из:

1) ввод параметров печати (температура, материал, скорость печати, углы печати и т.д.);
2) преобразование входных параметров печати в допустимый для программы вид;
3) ввод 3d модели;
4) преобразование модели в допустимый для программы вид.

На рисунке 26 представлена декомпозиция процесса «Расчет деформаций и остаточных напряжений».

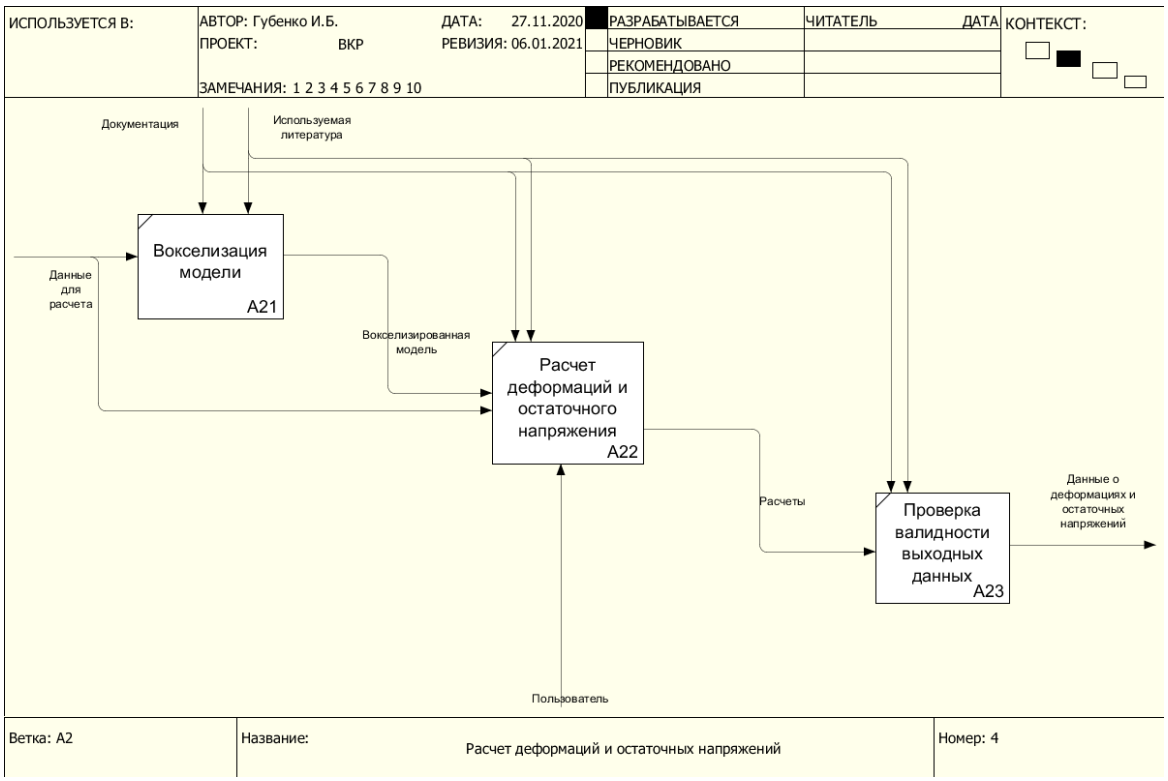

Рисунок 26 – Декомпозиция процесса «Расчет деформаций и остаточных напряжений»

Декомпозиция основного процесса состоит из:

1) вокселизация модели;
2) расчет деформаций и остаточного напряжения;
3) проверка валидности выходных данных.

2.4 Создание диаграммы потоков данных программного комплекса для расчета остаточных напряжений и деформаций металлоконструкций

Диаграмма потоков данных – это инструмент моделирования, который даёт возможность получить систему в виде сети функциональных процессов, соединенных один с другим стрелками данных. Диаграммы потоков данных (DFD) используются для описания движения документов и обработки информации как дополнение к IDEF0 [15].

Главная цель DFD – это показать, как каждый процесс преобразует свои входные данные в выходные и какие между ними взаимосвязи [21].

На рисунке 27 представлена контекстная диаграмма потоков данных

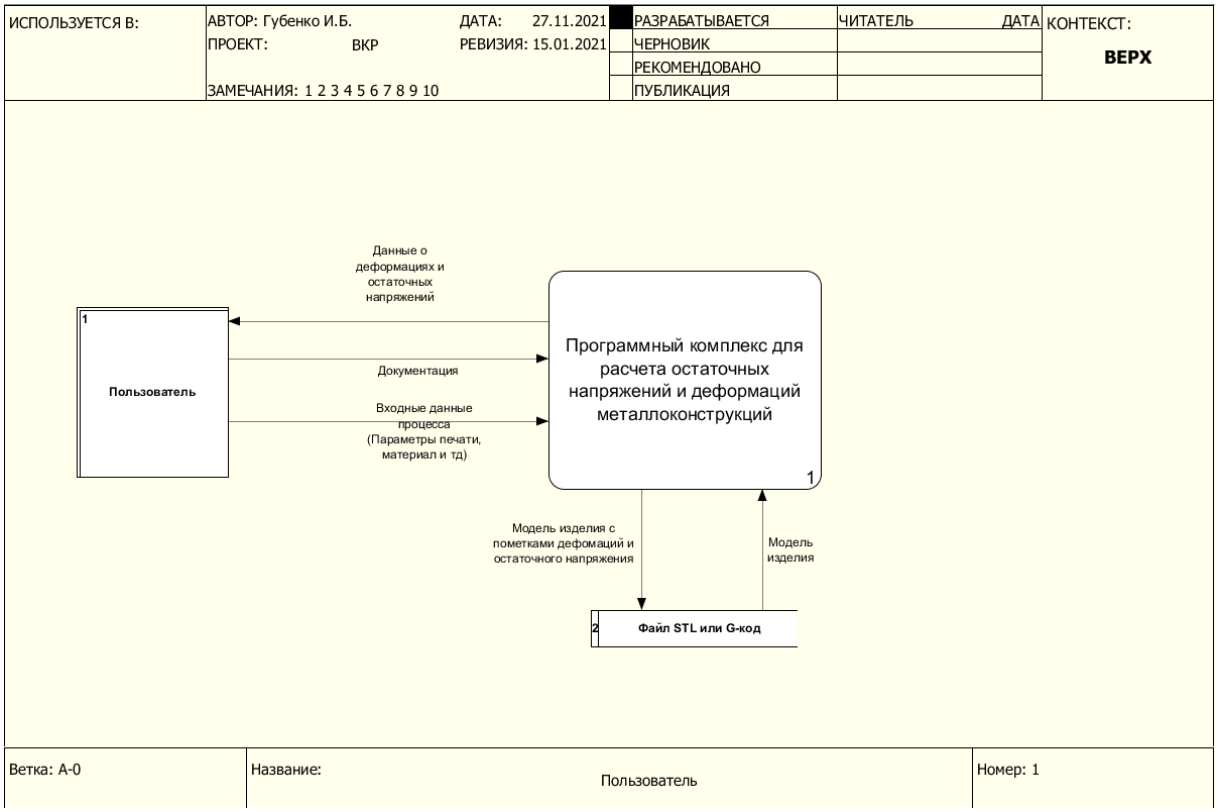

Рисунок 27 – Начальная диаграмма потоков данных

Диаграмма состоит из:

1) процесс: Программный комплекс для расчета остаточных напряжений и деформаций металлоконструкций;
2) внешняя сущность: Пользователь;
3) хранилище данных: Файл с моделью STL.

Наш основной процесс делится на совокупность небольших процессов, которые позволяют лучше понимать объект исследования, рассматриваемую область и сам основной процесс. На рисунке 28 представлена декомпозиция контекстной диаграммы.

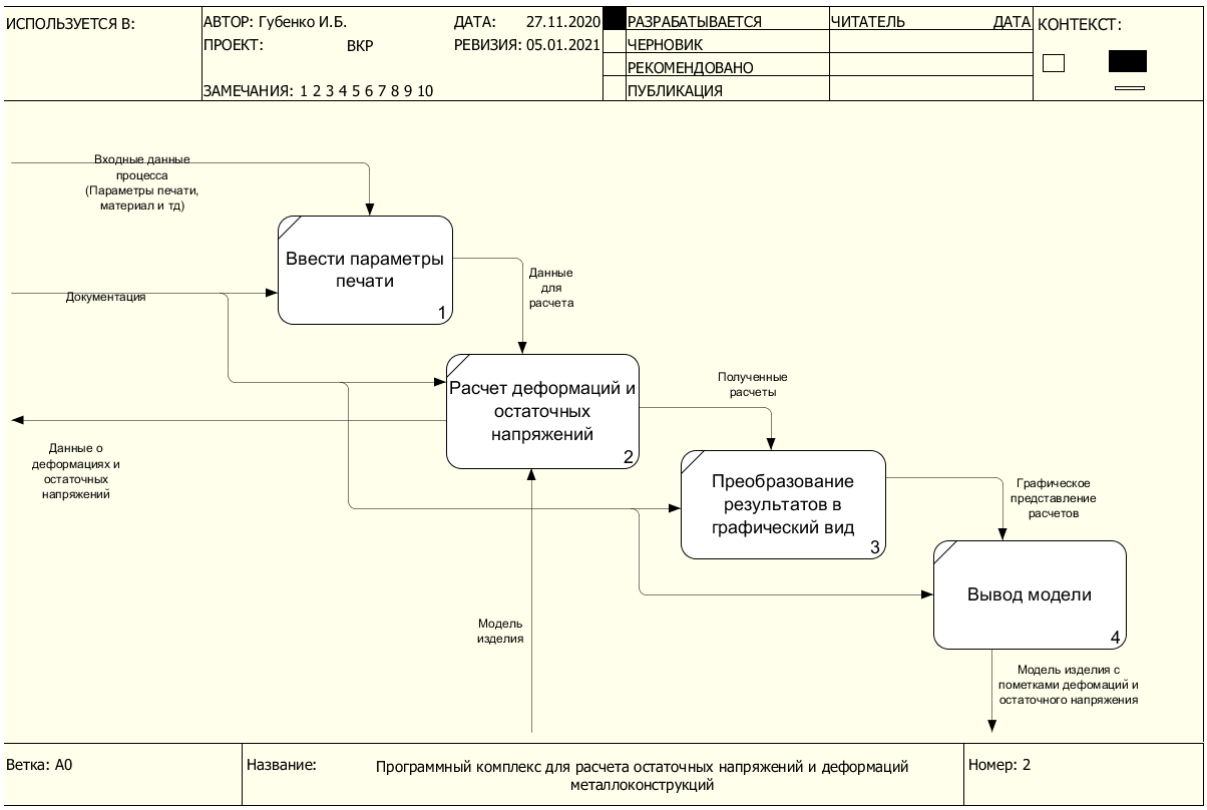

Рисунок 28 — Декомпозиция основного процесса

Декомпозиция контекстной диаграммы состоит из:

1) ввести параметры печати;
2) расчет деформаций и остаточных напряжений;
3) преобразование результатов в графический вид;
4) вывод модели.

2.6 Создание диаграммы классов UML программного комплекса для расчета остаточных напряжений и деформаций металлоконструкций

Диаграмма классов дает представление о программной системе в целом при помощи отображения классов, атрибутов и их взаимодействий. Диаграмма классов показывает типы объектов в системе и их виды отношений. Диаграмма классов устанавливает общее представление о системе. На рисунке 29 изображена диаграмма классов

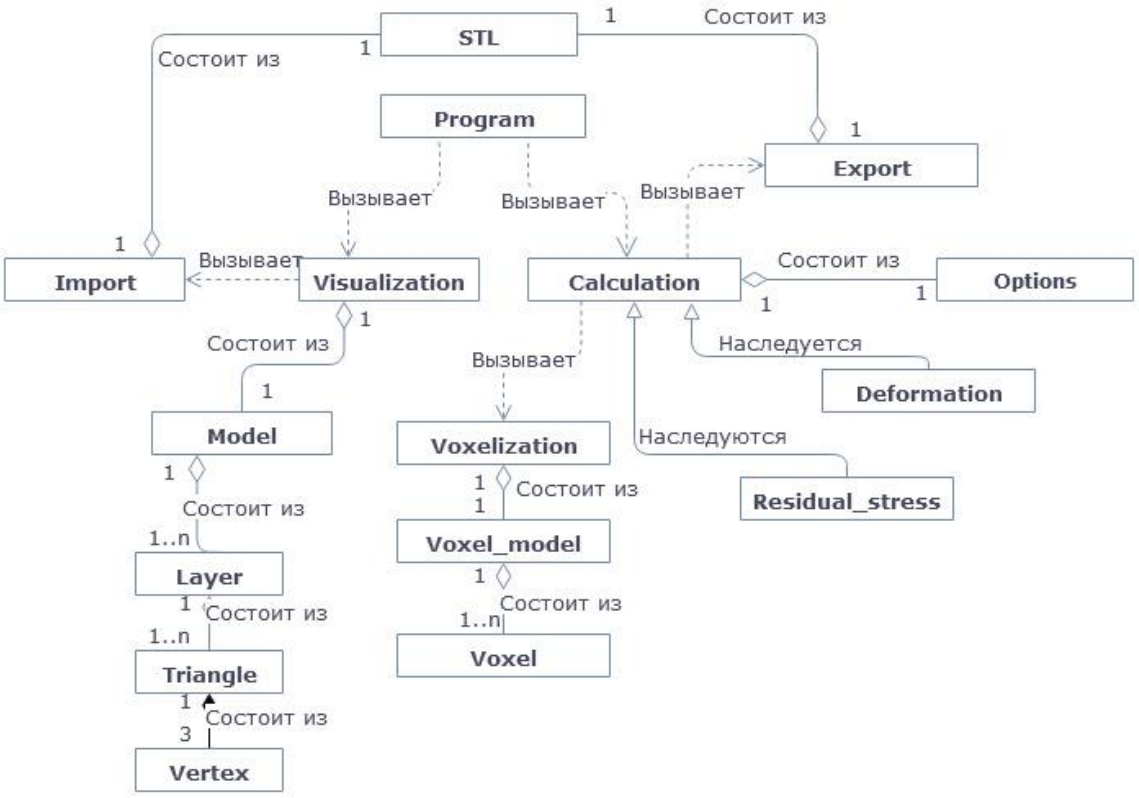

Рисунок 29 — Диаграмма классов

Класс Program стандартный класс, используется для взаимосвязи остальных действующих модулей системы и интерфейса.

Класс STL используется для представления, хранения и первичной обработки данных входящего или выходящего файла формата StereoLithography (STL).

Класс Import предназначен для импорта файлов из вне и для дальнейшей передачи файлов в другие модули системы.

Класс Export предназначен для экспорта файлов во внешнее хранилище.

Класс Visualization используется для представления графического представления входных и выходных файлов в интерфейсе программы.

Класс Model предназначен для хранения готовой для визуализации в интерфейсе модели, используется в классе Visualization. Состоит из объектов класса Layer.

Класс Layer предназначен для представления слоев в доступный для программы формат, включает в себя множество объектов класса Triangle

Класс Triangle предназначен для представления фасетов модели в удобный для программы формат. Состоит из трех объектов класса Vertex

Класс Vertex предназначен для представления вершин модели в доступный для программы вид. Главными полями являются X, Y, Z представляющие собой координаты вершины в трехмерном пространстве

Класс Calculation предназначен для вычислений остаточного напряжения и деформаций металлоконструкций, наследует два класса ResidaulStress и Deformation

Класс ResidaulStress является наследником класса Calculation предназначен для вычисления остаточного напряжения

Класс Deformation является наследником класса Calculation предназначен для вычисления деформаций металлоконструкций

Класс Options хранит в себе все параметры и настройки введенные с интерфейса программы. Параметры и настройки в связи с моделью будут использоваться как начальные данные для вычисления остаточного напряжения и деформаций металлоконструкций. К начальным параметрам относятся:

1) название проекта;
2) размер вокселя.

Далее перечисляются параметры

1) материал производства или его физические свойства (если нужного материала нет в списке);
2) модуль упругости (ГПа);
3) коэффициент Пуассона;
4) предел текучести (МПа);
5) коэффициент текучести опоры;
6) коэффициент масштабирования деформации;
7) коэффициенты анизотропной деформации;
8) коэффициенты анизотропной деформации;

9) коэффициенты анизотропной деформации.

Далее перечисляются параметры и настройки принтера или станка:

1) толщина слоя (10-100 мкм);
2) угол начального слоя;
3) угол поворота слоя;
4) ширина слоя;
5) мощность лазера;
6) скорость сканирования лазером;
7) температура опорной плиты;

Класс Voxelization предназначен для представления модели в воксельном формате для оптимизации вычислений и графического представления вычисленной модели и отображения остаточного напряжения, и деформаций. На рисунке 30 изображен пример воксельного представления модели.

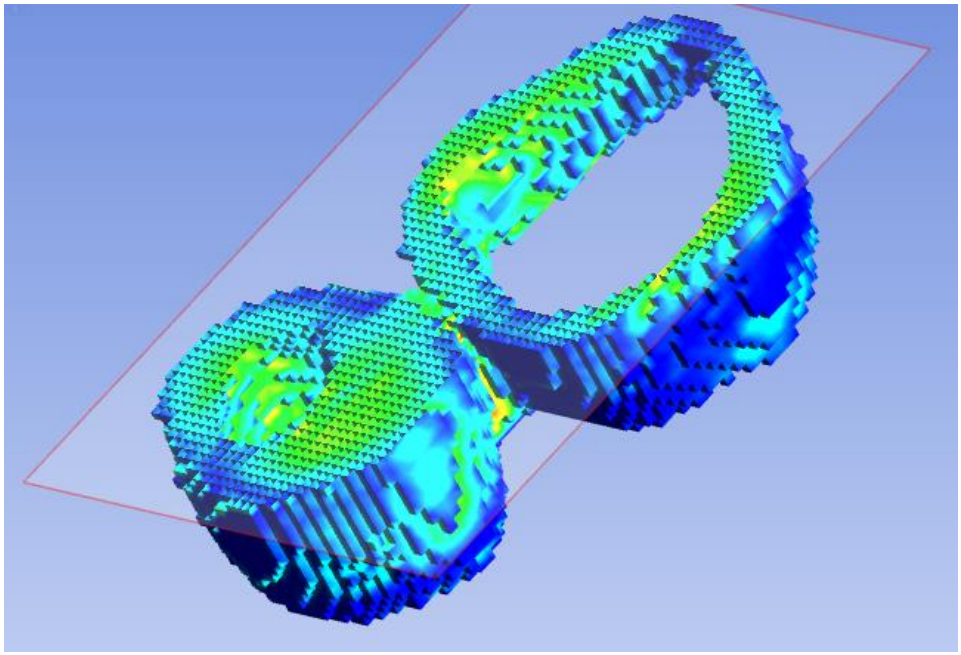

Рисунок 30 – Пример воксельного представления модели

Класс VoxelModel хранит в себе воксельную модель, состоит из множества объектов класса Voxel.

Класс Voxel хранит в себе данные и параметры одного вокселя.

2.7 Создание диаграммы вариантов использования UML программного комплекса для расчета остаточных напряжений и деформаций металлоконструкций

Диаграмма вариантов использования показывает взаимоотношение и зависимости группа вариантов использования и действующих лиц. Диаграмма вариантов использования является исходным концептуальным представлением системы в процессе ее проектирования и разработки [28]. Пример диаграммы использовании изображен на рисунке 31.

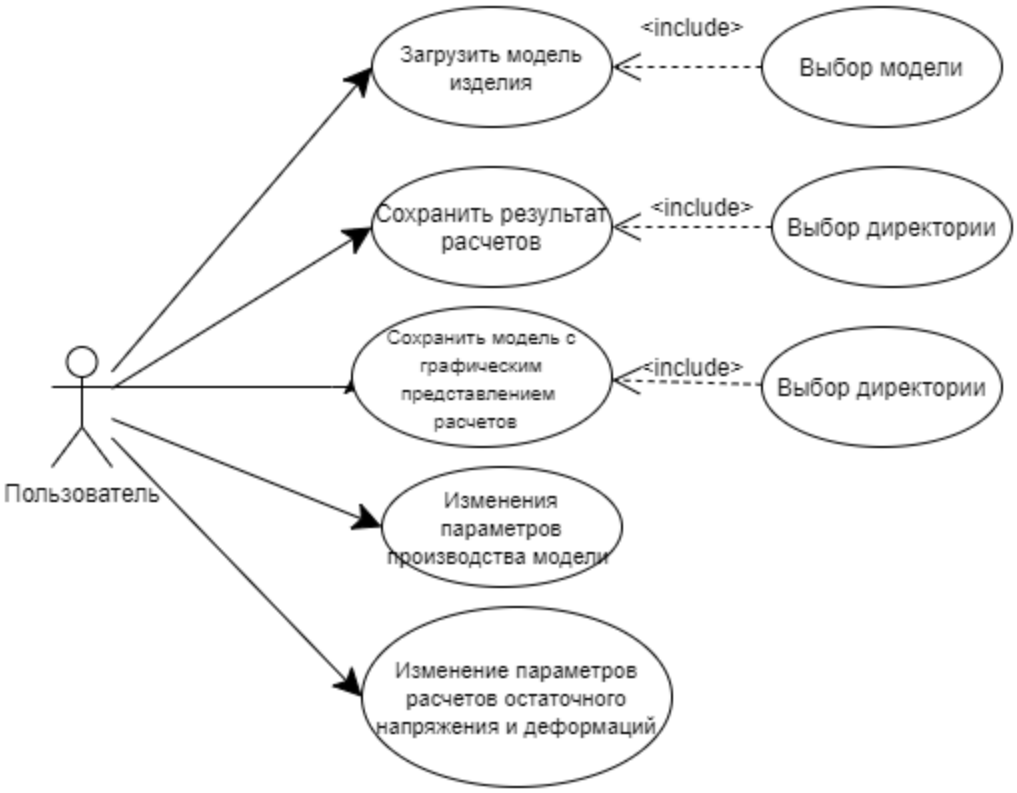

Рисунок 31 – Диаграмма вариантов использования.

Действующим лицом в диаграмме использования является пользователь. Во время работы пользователь может производить следующие действия:

1) загружать модель изделия из памяти компьютера в программу;

2) сохранять в указанную директорию результаты отчета в виде модели с графически выделенными участкам, которые подвержены остаточному напряжению и деформациям;
3) сохранить в указанную директорию результат расчетов в виде текстового файла или таблицы;
4) изменить параметры производства модели, от которых могут зависеть итоговые расчеты;
5) изменение параметров расчетов остаточного напряжения и деформаций.

3 Реализация программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций

Итоговая схема взаимодействия классов изображено на рисунке 32

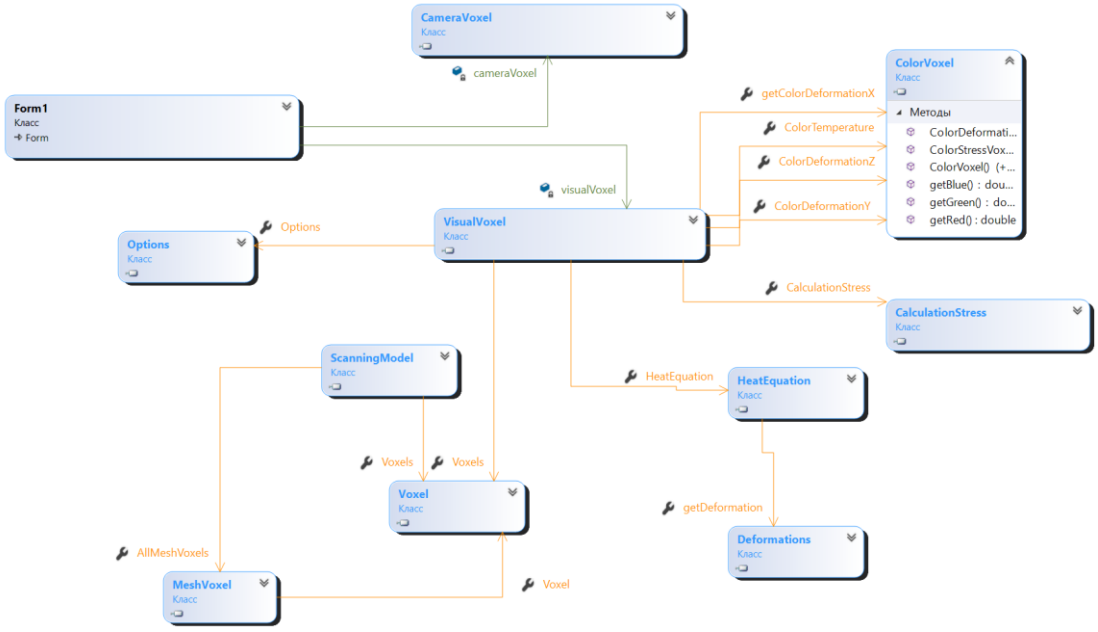

Рисунок 32 – Итоговая диаграмма классов.

3.1 Создание интерфейса программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций

Интерфейс это первое что видит конечный пользователь. Пользователь формирует свое первое мнение, только по интерфейсу. Интерфейс программного комплекса должен быть интуитивно понятен для любого пользователя и соответствовать требованиям UX/UI. Согласно UI (User Interface) требуется определить цветовую гамму, читабельность текста, удобство попадания на кнопки и прочие характеристики интерфейса. Также требуется продумать внешний облик интерфейса при выполнении этих действий. Согласно UX (User Experience) требуется определить целевую аудиторию приложения или программного комплекса и на основе анализа сделать интерфейс как можно более приспособленным для до-

стижения той цели, ради которой он был создан. Помимо этого, требуется позаботится о комфорте пользователя и его стимулировании на выполнение определенных действий.

Для создания наиболее оптимального интерфейса требуется сначала создать шаблон или прототип интерфейса. Создание прототипа интерфейса будет производиться в приложение Figma, которое находится в свободном доступе.

Согласно целям и требованием создания программного комплекса, интерфейс должен включать себе несколько сцен для визуального отображения результатов, а именно сцена формирования полей остаточного напряжения, сцена расчета теплопроводности во время процесса производства. Кнопки импорта и экспорта данных, кнопка старта расчетов и поля для ввода параметров.

На основе требований был создан прототип интерфейса программного комплекса в приложение Figma. На рисунке 33 изображен прототип интерфейса.

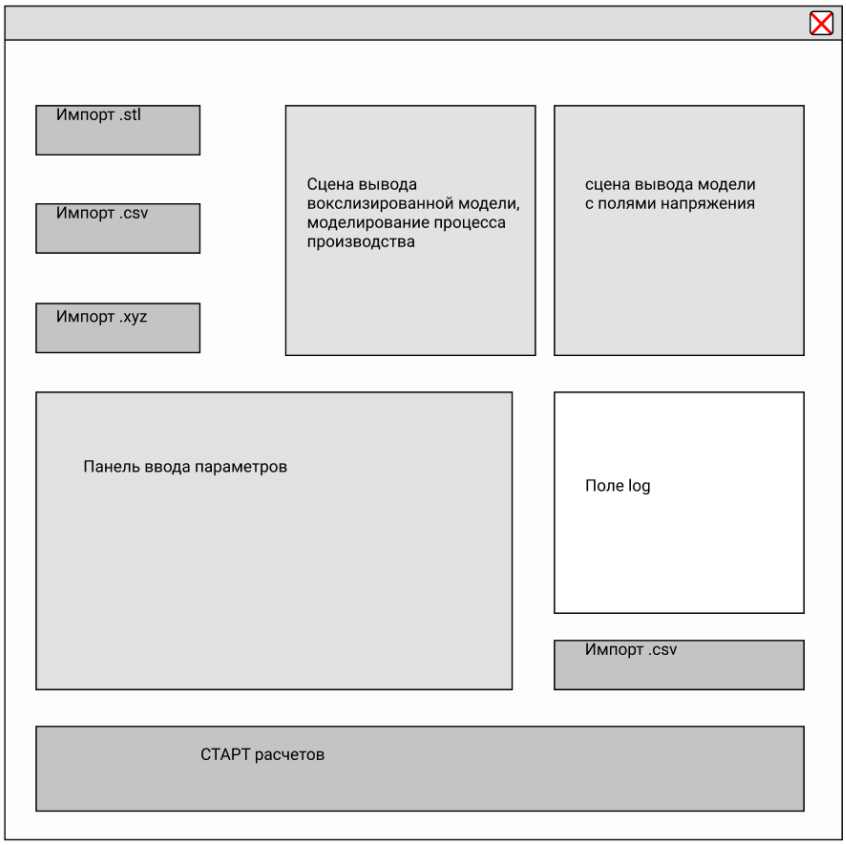

Рисунок 33 – Прототип интерфейса приложения

3.2 Создание модуля визуализации входной модели в воксельном формате

Воксельная визуализация требуется для полного понимания как будет происходить расчет. Воксель представляет собой куб с заданным размером ребра. Входная модель в формате .stl или .xyz (данный формат представляет собой текстовый файл с последовательным перечисление значений координат всех вокселей). Воксельная графика позволяет оптимизировать расчет. В модели воксель будет представлять собой объем мелкодисперсного порошка.

Для визуализации использовалась библиотека SharpGL, которая позволяет работать с OpenGL из Windows.Forms.

3.2.1 Создание библиотеки VisualVoxelLibrary для визуализации и управления камерой сцены

Для удобства работы с библиотекой и оптимизации иерархии кода была создана библиотека `VisualVoxelLibrary` которая состоит из классов `Camera-Voxel.cs` и `VisualVoxel.cs`.

Класс `VisualVoxel.cs` выступает в роли контейнера для класса `VoxelLibrary.Voxel.cs`. Класс `VisualVoxel.cs` включает в себя массив объектов класса `VoxelLibrary.Voxel.cs`, объект класса `VoxelLibrary.ColorVoxel.cs`.

Метод `drawVoxel(…)` принимает на вход аргументы: объект класса `VoxelLibrary.ColorVoxel.cs`, объект класса `VoxelLibrary.Voxel.cs` и производит визуализацию вокселя на сцене используя функции библиотеки SharpGL

Метод `drawSkeleton(…)` принимает на вход аргументы: объект класса `VoxelLibrary.ColorVoxel.cs`, объект класса `VoxelLibrary.Voxel.cs` и производит визуализацию каркаса вокселя на сцене используя функции библиотеки SharpGL

Метод `ImportXYZ()` считывает с файла формата .xyz данные и преобразует их в массив объектов класса `VoxelLibrary.Voxel.cs`

Метод `Visualization()` производит визуализацию на сцене всего массива объектов класса `VoxelLibrary.Voxel.cs`. Реализация метода `Visualization()` демонстрируется в листинге 1, полная реализация класса представлена в листинге А.1.

Листинг 1 – Метод `Visualization()`

```
        public void VisualizationTemperatyre3(OpenGL gl, OpenGL gl2, bool
boolPauseCalc,int countStep)
     {
         double temperature = 0;

         meshVoxels = scanningModel.getAllMeshVoxels();
         heatEquation.setScaningVoxels(scanningModel.SnakeScanning());

         if (boolPauseCalc)
         {

             if      (heatEquation.CountScanningVoxels<heatEquation.Scan-
ingVoxels.Length)
                 for (int i = 0; i < countStep; i++)
                 {
                     heatEquation.CalculationHeatEquation();
                 }
         }
         for   (int   z   =   scanningModel.getMinZ();   z   <=   scanning-
Model.getMaxZ(); z++)
         {
             for  (int  x  =  scanningModel.getMinX();  x  <=  scanning-
Model.getMaxX(); x++)
             {
                 for  (int  y  =  scanningModel.getMinY();  y  <=  scanning-
Model.getMaxY(); y++)
                 {

                     temperature = heatEquation.getTemperature(x, y, z);
                     colorTemperature.ColorStressVoxel(temperature);
                     colorDeformationX.ColorDeformationVoxel(heatEqua-
tion.getDeformationX(x, y, z));
                     colorDeformationY.ColorDeformationVoxel(heatEqua-
tion.getDeformationY(x, y, z));
                     colorDeformationZ.ColorDeformationVoxel(heatEqua-
tion.getDeformationZ(x, y, z));

                     if (heatEquation.boolMelted(x, y, z))
                     {
                         drawVoxel(gl, colorTemperature, meshVoxels[x, y,
z]);
                         drawVoxel(gl2,
```

```
                                        colorDeformationX,
                                        colorDeformationY,
                                        colorDeformationZ,
                                        meshVoxels[x, y, z]);
                    }
                    else
                    {
                      drawSkeleton(gl,  colorTemperature,  meshVoxels[x,
y, z]);
                      drawSkeleton(gl2, colorTemperature, meshVoxels[x,
y, z]);
                    }
                }
            }
        }
    }
```

3.2.2 Создание библиотеки VoxelLibrary для описания основных свойств вокселя

Для функционирования библиотеки `VisualVoxelLibrary` была создана библиотека `VoxelLibrary` состоящая из классов `ColorVoxel.cs` и `Voxel.cs`

Класс `Voxel.cs` описывает воксель, в классе прописаны поля координат вокселя, конструктор для создания объекта класса, а также геттеры и сеттеры для изменения и получения значений полей. Объекты класса `Voxel.cs` заполняют контейнеры для дальнейщего взаимодействия с каждым вокселем.

Класс `ColorVoxel.cs` позволяет индивидуально указывать цвет каждого вокселя, а также рассчитывает градиент в зависимости от минимальной температуры или напряжения в конкретном вокселе.

Класс `ColorVoxel.cs` состоит из полей, double red, double green, double blue задающие цвет вокселя, геттеров на каждое из полей.

При вызове конструктора `ColorVoxel(double sMin, double sMax)` происходит расчет шага по цвету для того что бы построить ранвномерный градиет в зависимости от минимального и максимального показателя температуры или уровня напряженности.

Метод `ColorStressVoxel (double valueStress)` изменят значения цвета воксля при конкретном показателе температуры или уровня напряжения

3.3 Создание модуля расчета остаточного напряжения и деформаций металлоконструкций

Модуль расчета состоит из нескольких классов `CalculationStress.cs`, `Deformations.cs`, `HeatEquation.cs`, `MeshVoxel.cs`, `Options.cs`, `ScanningModel.cs`.

Класс `MeshVoxel.cs` строит пространственную сетку на основе занятых вокселей.

Класс `Options.cs` передает параметры входных данных в классы расчета

Класс `ScanningModel.cs` описывает метод производства, то есть в какой последовательности будут спекаться воксели.

Класс `HeatEquation.cs` описывает процессы теплоотвода и нагрева рабочей зоны и всей системы в целом

Класс `Deformations.cs` рассчитывает деформации, вызванные циклическим нагревом.

3.3.1 Программная реализация численной модели процесса аддитивного производства

Для моделирования процессов аддитивного производства используется класс `HeatEquation.cs`. Для моделирования процессов теплообмена при изготовлении модели требуется учитывать температуру в каждой точке рабочей области, для это создан трехмерный массив `T_Next, T_Curent`.

Далее требуется решить уравнений (7-13).

Для решения уравнений (7-13) построим сетку коэффициентов теплопроводности в виде трехмерного массива, коэффициенты теплопроводности будут меняться в зависимости от состояния воксели и этапа его производства, выделим несколько коэффициентов теплопроводности: коэффициент теплопроводности мелкодисперсного порошка, коэффициент теплопроводности мелкодисперсного

порошка после спекания или сплавления, коэффициент теплопроводности воздуха или инертной среды в камере изготовления, коэфициент теплопроводности подложки на которой изготавливается деталь. . В листинге 2 демонстрируется реализация метода `CalculationHeatEquation()`,полная реализация класса представлена в листинге А.2.

Листинг 2 – Метод CalculationHeatEquation()

```
        public void CalculationHeatEquation(){

         //Расчет следующего значения температуры в сетке
         Parallel.For(1,K-1,
         k =>

         {
             for (int j = 1; j < J - 1; j++)
             {
                 for (int i = 1; i < I - 1; i++)
                 {
                     T_Next[i, j, k] = T_Curent[i, j, k] + (K_Val-
ues[i,j,k]*dt / (dx * dx)) * (
                                                 K_half_plus_i(i, j, k) *
(T_Curent[i + 1, j, k] - T_Curent[i, j, k])
                                                - K_half_minus_i(i, j, k)
* (T_Curent[i, j, k] - T_Curent[i - 1, j, k])
                                                +
                                                (K_half_plus_j(i, j, k) *
(T_Curent[i, j + 1, k] - T_Curent[i, j, k])
                                                - K_half_minus_j(i, j, k)
* (T_Curent[i, j, k] - T_Curent[i, j - 1, k]))
                                                +
                                                 (K_half_plus_k(i, j, k) *
(T_Curent[i, j, k + 1] - T_Curent[i, j, k])
                                                - K_half_minus_k(i, j, k)
* (T_Curent[i, j, k] - T_Curent[i, j, k - 1]))
                                                )
                                                + Q_source(t, i, j, k);

     //изменение агрегатного состояния (порошок превратился в металл)
                     if (T_Next[i, j, k] > T_fusion_titan)
                     {
                         K_Values[i, j, k] = D_metal;
                         boolPrinted[i, j, k] = true;
                     }
                 }
             }

         });
         for (int k = 0; k < K; k++)
         {
             for (int j = 0; j < J; j++)
             {
```

```
                for (int i = 0; i < I; i++)
                {
                    if (i == 0 || i == I - 1)
                        T_Next[i, j, k] = T_start;
                    if (j == 0 || j == J - 1)
                        T_Next[i, j, k] = T_start;
                    if (k == 0 || k == K - 1)
                        T_Next[i, j, k] = T_start;
                }
            }
        }
        /////
        deformation.CalculationDeformations(T_Next, boolPrinted);
                    T_Curent = T_Next;
        //измененение шага по времени для устойчивости
        dt = (dx * dx) / (2 * Tmax()) ;//tay
        t += dt;

    }
```

3.3.2 Программная реализация численной модели расчета остаточного напряжения и деформаций металлоконструкций

Для моделирования деформаций требуется учитывать процессы теплообмена и знать значение перепада температуры у всех вокселей в системе

Для моделирования деформаций при аддитивном производства используется класс `Deformations.cs`

Далее требуется программно решить уравнения (14-25).

Уравнения (14-25) прораммно решаются на основе результатов расчета процессов теплопроводности из класса `HeatEquation.cs`. В листинге 3 демонстрируется реализация метода `CalculationDeformations()`, полная реализация класса представлена в листинге А.3.

Листинг 3 – Метод CalculationDeformations()

```
      public void CalculationDeformations(double[,,]  T, bool[,,] bool-
Printed)
    {
    Parallel.For(0, K ,
    k =>
    //for (int k = 0; k < N; k++)
    {
```

```
            for (int j = 0; j < J; j++)
            {
                for (int i = 0; i < I; i++)
                {
                    if (boolPrinted[i, j, k])
                        alpha_cup[i, j, k] = alpha_0 * (T[i, j, k] - T_start);
                    //else
                    //    alpha_cup[i, j, k] = 0;

                }
            }
        });
        Parallel.For(1, K - 1,
        k =>
        {
            for (int j = 1; j < J - 1; j++)
            {
                for (int i = 1; i < I - 1; i++)
                {
                    Ex_Next[i, j, k] = -dx * ((alpha_cup[i + 1, j, k] -
alpha_cup[i - 1, j, k]) / 2);
                    Ey_Next[i, j, k] = -dx * ((alpha_cup[i, j + 1, k] -
alpha_cup[i, j - 1, k]) / 2);
                    Ez_Next[i, j, k] = -dx * ((alpha_cup[i, j, k + 1] -
alpha_cup[i, j, k - 1]) / 2);


                    if (Math.Abs(Ex_Curent[i, j, k]) > Math.Abs(E_crit))
                        if      (Math.Abs(Ex_Next[i,      j,      k])      <
Math.Abs(Ex_Curent[i, j, k]))
                            Ex_Next[i, j, k] = Ex_Curent[i, j, k];



                    if (Math.Abs(Ey_Curent[i, j, k]) > Math.Abs(E_crit))
                        if      (Math.Abs(Ey_Next[i,      j,      k])      <
Math.Abs(Ey_Curent[i, j, k]))
                            Ey_Next[i, j, k] = Ey_Curent[i, j, k];



                    if (Math.Abs(Ez_Curent[i, j, k]) > Math.Abs(E_crit))
                        if      (Math.Abs(Ez_Next[i,      j,      k])      <
Math.Abs(Ez_Curent[i, j, k]))
                            Ez_Next[i, j, k] = Ez_Curent[i, j, k];
                }
            }
        });
        Parallel.For(0, K, k =>
        {
            for (int j = 0; j < J; j++)
            {
                for (int i = 0; i < I; i++)
                {
                    Ex_Curent[i, j, k] = Ex_Next[i, j, k];
                    Ey_Curent[i, j, k] = Ey_Next[i, j, k];
                    Ez_Curent[i, j, k] = Ez_Next[i, j, k];
                }
```

```
    }
});
```

3.4 Функционирование программного комплекса для расчета остаточного напряжения и деформаций

Запускаем программный комплекс для расчета остаточных напряжений и деформаций CoRSaD (Calculation of residual stresses and deformations). Начальный интерфейс представляет собой две пустых сцены. Сцена слева визуализирует распределение температур в материале при изготовлении, правая сцена визуализирует температурные деформации, связанные с цикличным нагревом. На рисунке 34 изображен начальный интерфейс разработанной программы CoRSaD.

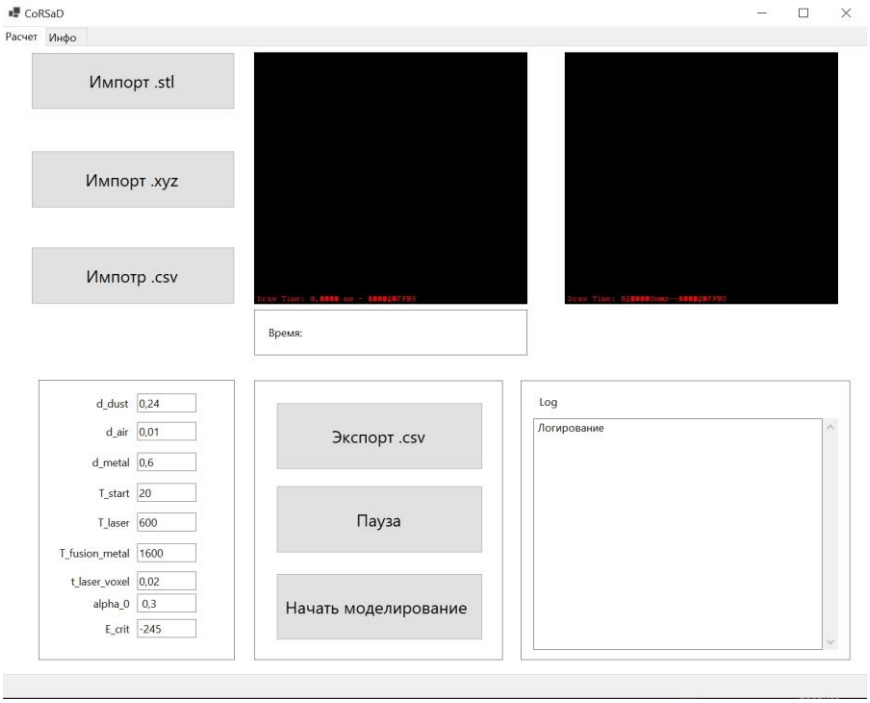

Рисунок 34 – Интерфейс при запуске программы

На следующем этапе требуется ввести начальные значения и импортировать модель в нужном формате. На рисунке 35 изображено окно импорта файла.

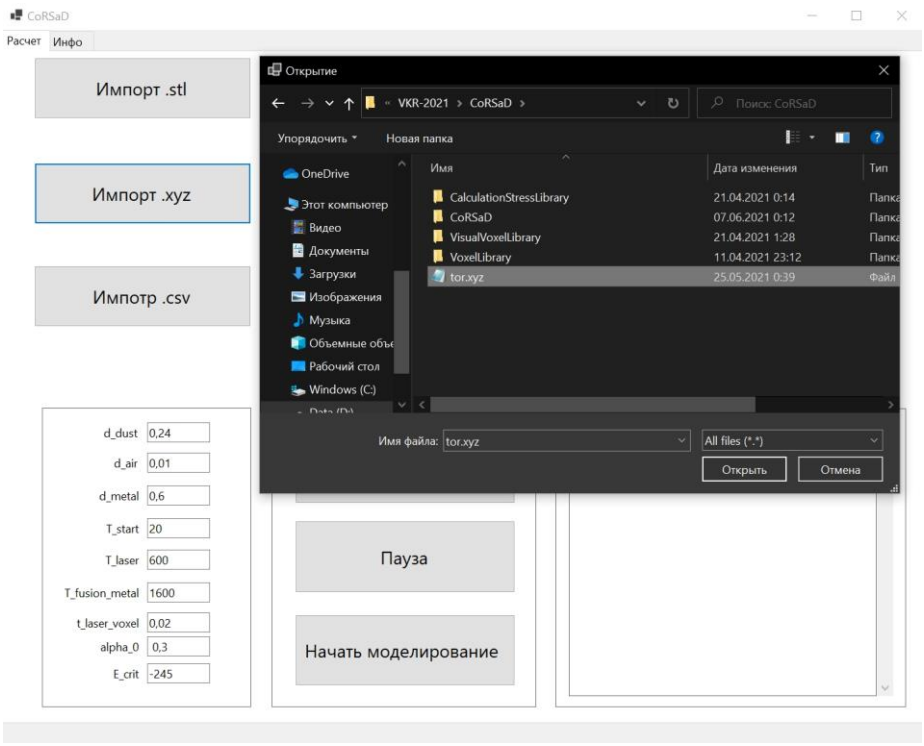

Рисунок 35 – Импорт файла

Далее нажимаем на кнопку «Начать моделирование». На рисунке 36 изображен интерфейс программы во время расчета

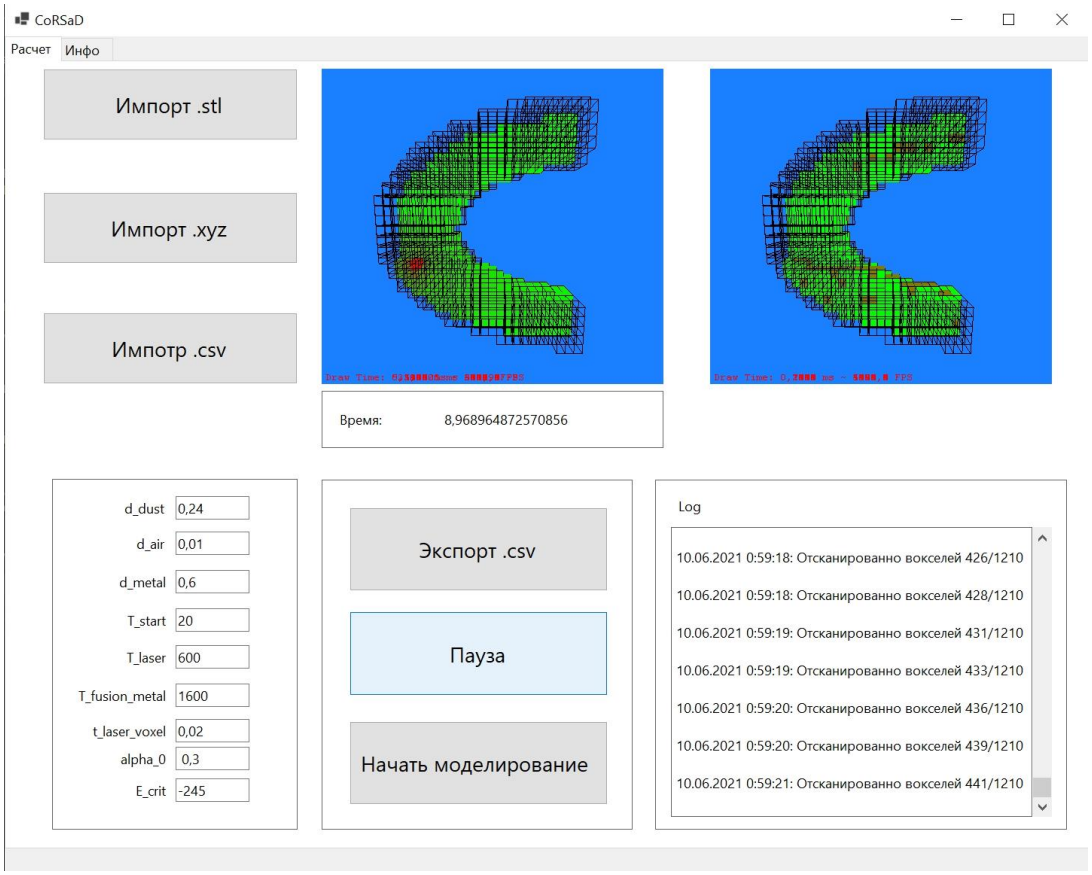

Рисунок 36 – Визуализация расчетов

В любой момент времени расчетов можно экспортировать данные в .csv. Для этого требуется нажать на кнопку «Экспортировать .csv», далее требуется выбрать имя и директорию сохранения файла. На рисунке 37 изображен этап экспорта.

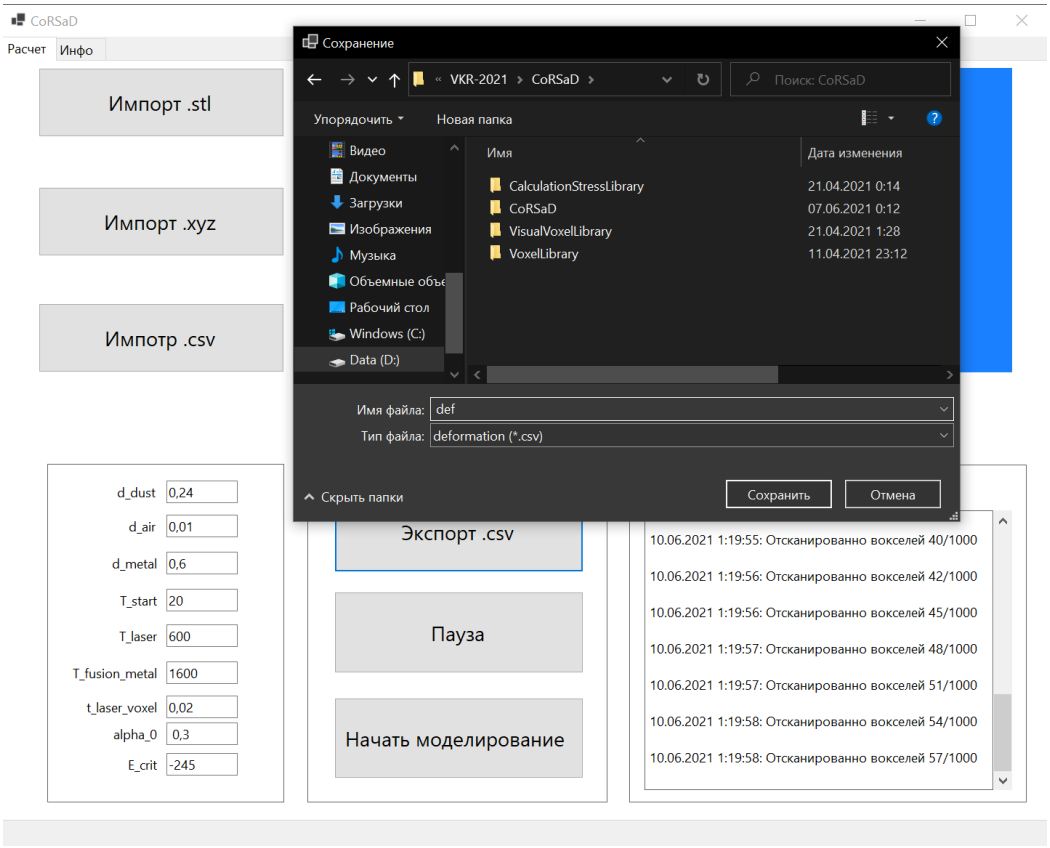

Рисунок 37 – Экспорт расчетов

В связи с тем, что изначально программа вычисляла данные в трехмерных сетках заданной величины, вычисление могли занимать продолжительный срок, для оптимизации вычислений было предпринято решение сделать динамическое определение сетки расчетов это существенно повысило производительность визуализации в 3D-сценах и скорости расчета в целом.

Далее двигаясь в сторону оптимизации расчетов было принято решение прибегнуть к использованию параллельного программирования, так как в программе присутствуют множество вложенных циклов, в которых гонка данных исключена, это дало приличный прирост к скорости расчета как на слабых, так и на более мощных ПК(Персональный компьютер).

Оптимизация аргументирована тем что изначально программный комплекс работал в однопоточном режиме и нагружал все один-два логических процессора, но после применения парадигм параллельного программирования все логические процессоры нагружались в равных долях.

Тест производительности осуществлялся на нескольких ПК.

Комплектующие ПК-1:

- Процессор Intel Core i5-10210U (2.11 GHz, 4 ядра, 8 логических процессоров)
- Видеокарта Nvidia Geforce MX250
- ОЗУ 8 Гб

Комплектующие ПК-2:

- Процессор Intel Xeon E5-2650 v3 (2.30 GHz, 20 ядра, 40 логических процессоров)
- Видеокарта Nvidia Quadro K6000
- ОЗУ 256 Гб

Рассчитаем скорость расчета на ПК-1 и сравним до оптимизации и после оптимизации. До оптимизации скорость расчета на модели в 2798 вокселей и с соотношением 1 такт визуализации на 100 тактов вычислений, равнялась 1 воксель в секунду. Данные вычислений и занятость логических процессоров изображены на рисунке 38.

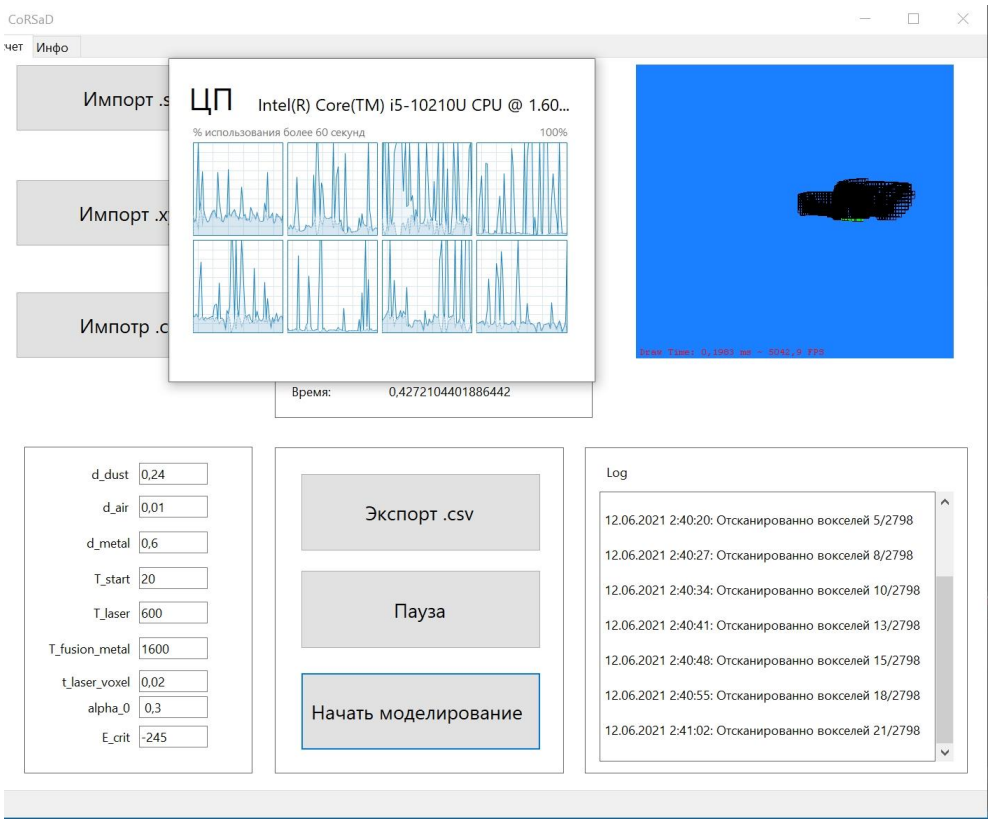

Рисунок 38 – Занятость логических процессоров до оптимизации

После оптимизации на аналогичных исходных данных скорость увеличилась до 17 вокселей в секунду, что является существенным приростом производительности. Данные вычислений и занятость логических процессоров изображены на рисунке 39.

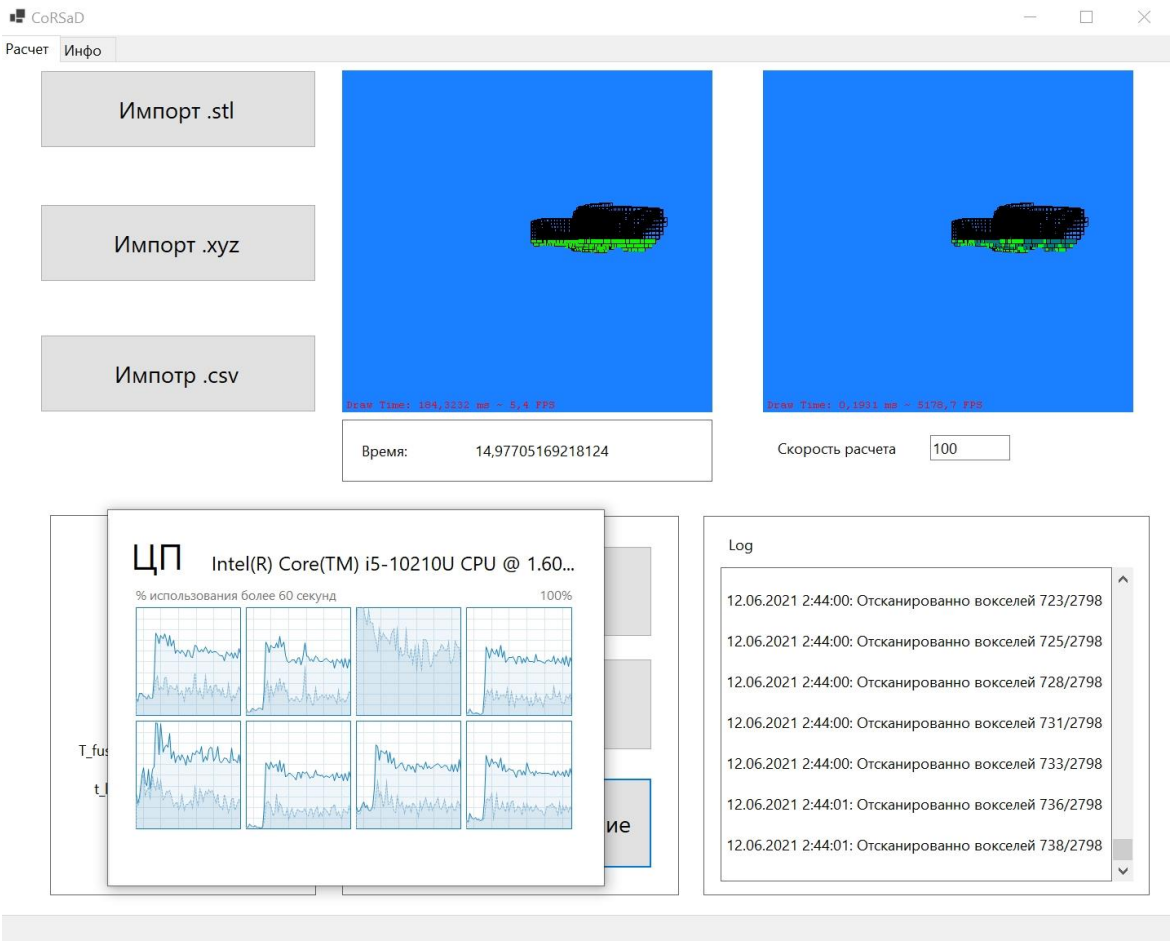

Рисунок 39 – Занятость логических процессоров после оптимизации

Так же оптимизации позволяет считать детализированные модели размером более 71000 вокселей. На рисунке 40 изображена занятость логических процессоров во время вычисления детализированной модели на ПК-2.

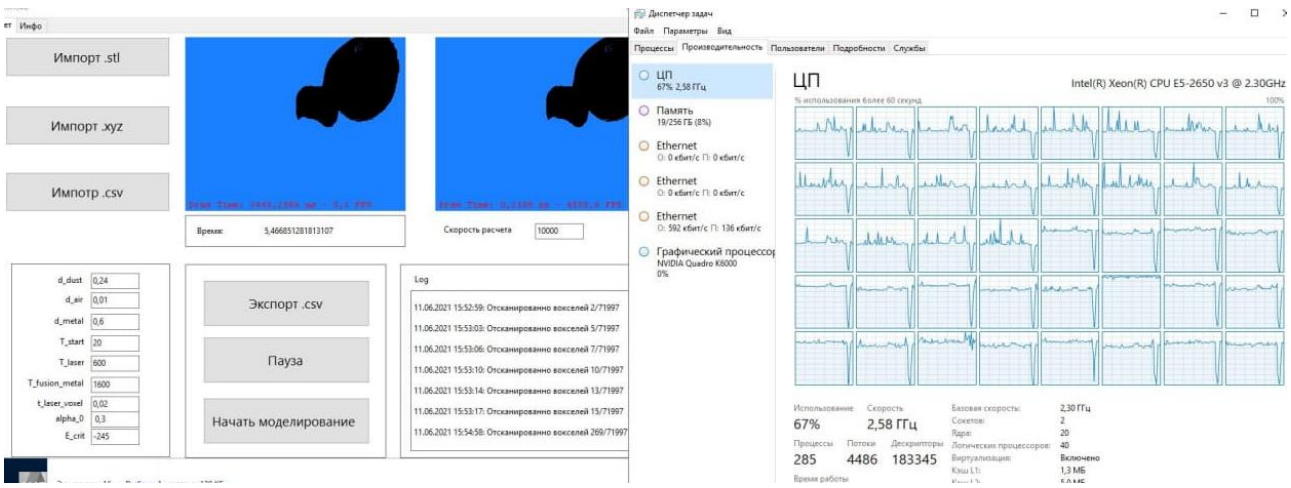

Рисунок 40 – Занятость логических процессоров на ПК-2

Далее произведем расчет для модели половины тора. Модель для расчета изображена на рисунке 41.

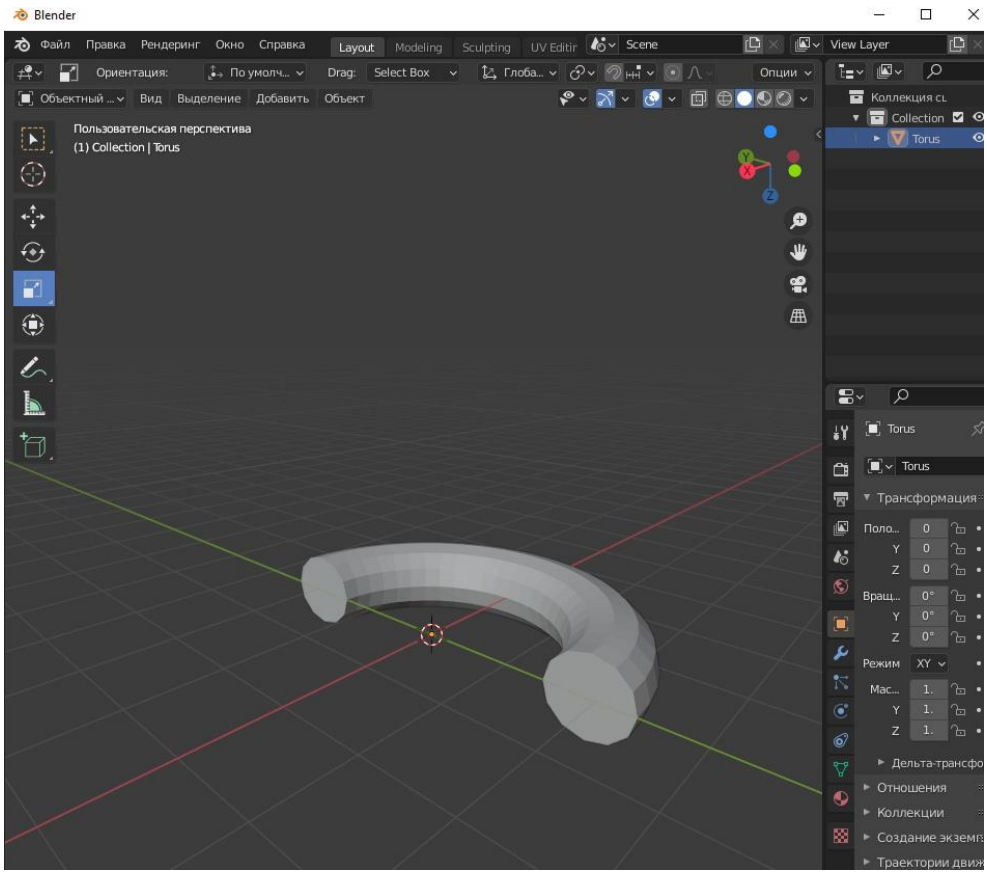

Рисунок 41– Модель для расчета

Далее произведем расчет в Ansys Additive Print. Результат расчета изображен на рисунке 42.

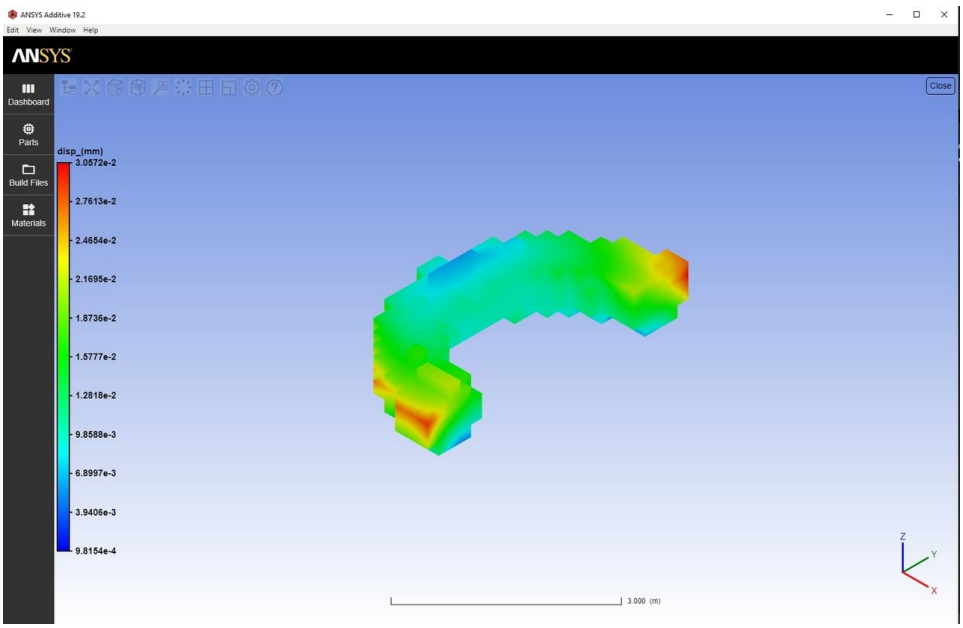

Рисунок 42– Результат расчета в Ansys Additive Print.

На рисунке 43 изображены результаты расчета в реализуемом программном комплексе CoRSaD.

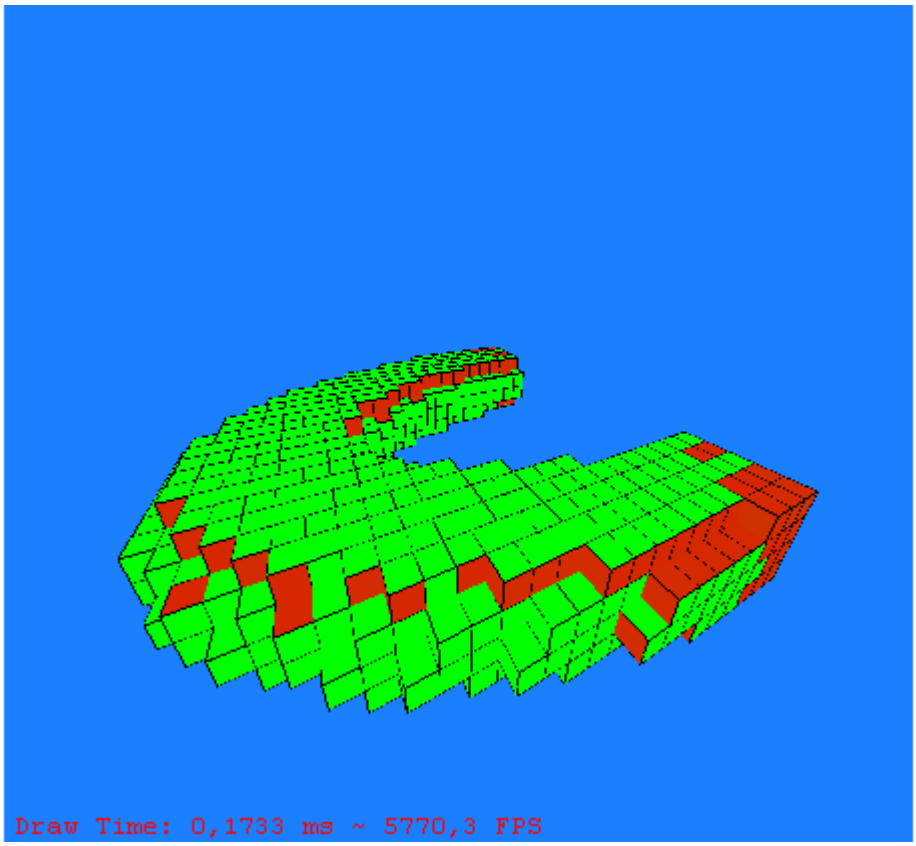

Рисунок 43– Результат расчета в CoRSaD.

Заключение

В рамках выполненной работы создана информационная модель программного комплекса для вычисления остаточного напряжения и деформаций металлоконструкций, реализован сам программный комплекс. Были изучены популярные технологии аддитивного производства сложных конструкций из металла и причины возникновения остаточного напряжения и деформаций, связанных с усадкой. Были рассмотрены способы минимизации деформаций и последствий остаточного напряжения

На основе изученного материала были получены следующие результаты:

1) изучены технологии аддитивного производства сложных металлических конструкций;
2) изучены способы устранения остаточных напряжений и усадочных деформаций;
3) создана функциональная модель программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций;
4) создана диаграмма потоков данных для программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций;
5) создана диаграмма состояний для программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций;
6) создана диаграмма вариантов использования для программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций;
7) создана диаграмма Ганта для контроля этапов разработки программного комплекса;
8) создана диаграмма классов программного комплекса для расчета остаточного напряжения и деформаций металлоконструкций;
9) разработана математическая модель модулей расчета остаточных напряжений и деформаций;

10) реализован программный комплекс для расчета остаточного напряжения и деформаций.

При выполнении работы производился поиск информации по темам: «Аддитивное производств», «SLM технология 3D-печати», «EBM технология 3D-печати», «DMLS технология 3D-печати», «Остаточные напряжения и деформации металлоконструкций», «Численные методы решения дифференциальных уравнений», «Построение математической модели аддитивного производства», «Создание информационной модели», «Диаграмма Ганта», «Функциональное моделирование IDEF0», «Диаграмма поток данных», «Диаграмма классов», «Диаграмма вариантов использования», «Требования UX/UI», «Визуализация OpenGL», «Парадигмы параллельного программирования» и проводить ее критический анализ, выявление необходимой информации для реализации задачи, что соответствует компетенции:

УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач.

Была спроектирована информационная модель программного комплекса, которая содержит диаграммы IDEF0 и DFD, в каждой из которых произведена декомпозиция выделенных процессов. В результате были определены задачи, необходимые для реализации программного комплекса, и разработан календарный график выполнения ВКР на основе диаграммы Ганта для планирования и управления задачами. Также использовалось рекомендуемое программное обеспечение в процессе создания выпускной квалификационной работы. Это соответствует освоению компетенции:

УК-2 Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений.

В процессе работы над программным комплексом в устной и письменной форме осуществлялась коммуникация с научным руководителем Храповым С.С., по вопросам выполнения выпускной квалификационной работы. Также производились консультации с преподавателями кафедры ИСКМ и сотрудниками ООО

«Теленово», специалистами в сфере 3D-моделирования, что соответствует освоению следующей компетенции:

УК-3 Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде.

В ходе выполнения работы в устной и письменной формах осуществлялась коммуникация с научным руководителем, с преподавателями кафедры ИСКМ в лице Титова А.В. по вопросу организации параллельных вычислений, Радченко В.П. по вопросам работы 3D-принтера технологии селективной лазерной плавки, с членами IT-сообщества в рамках русско- и англоязычных тематических форумов, что соответствует компетенции:

УК-4 Способен осуществлять деловую коммуникацию в устной и письменной формах на государственном языке Российской Федерации и иностранном(ых) языке(ах).

При выполнении работы была создана диаграмма Ганта, благодаря которой осуществлялось управление своим временем таким образом, чтобы этапы реализации программного комплекса и получение результатов работы производилось в установленные сроки, также были определены траектории саморазвития, что соответствует компетенции:

УК-6 Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни.

При разработке программного комплекса производилось создание его математической модели. Для ее создания применялись знания дисциплин математики и моделирования, естественнонаучные и общеинженерные знания, что соответствует освоению следующей компетенции:

ОПК-1 Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности.

В процессе изучения предметной области были использованы такие электронные библиографические базы, как Лань, Book.ru, elibrary и др. Отчет по выпускной квалификационной работы был сформирован посредством программного обеспечения LibreOffice, в программном обеспечение для создания трёхмерной компьютерной графики Blender создавались 3D-модели, разработка программного кода производилась в средах Microsoft Visual Studio и PyCharm, что соответствует компетенции:

ОПК-2 Способен использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности.

Для выполнения выпускной квалификационной работы была изучена предметная область расчета остаточных напряжений и деформаций металлоконструкций, произведенных методом селективной лазерной плавки, которая включает 3D-моделирование расчётной модели, создание и решение математической модели. Была разработана информационная модель, включающая диаграммы программного комплекса, созданные посредством программного обеспечения Ramus. Модули программного комплекса разрабатывались при помощи таких средств разработки, как Microsoft Visual Studio и PyCharm, также осуществлялся контроль версий проекта в системе контроля версий Git. Перечисленные задачи соответствуют освоению компетенции:

ОПК-3 Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности.

При выполнении работы был составлен отчет, включающий в себя информационную и математическую модели, которые содержат диаграммы, описывающие порядок выполнения задач программным комплексом и пользователем, и алгоритмы разработки частей, составляющих программный комплекс, что соответствует компетенции:

ОПК-4 Способен участвовать в разработке стандартов, норм и правил, а также технической документации, связанной с профессиональной деятельностью.

Для разработки программного комплекса была спроектирована его информационная модель, которая включает диаграммы DFD и IDEF0. Для создания диаграмм было установлено и использовано программное обеспечение Ramus, что соответствует компетенции:

ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем.

В рамках работы была разработана математическая модель программного комплекса, которая включает в себя алгоритм нахождения остаточных напряжений и деформации. По математической модели был разработан программный комплекс, позволяющий рассчитывать и визуализировать остаточные напряжения и деформации металлоконструкций по заданной модели в формате STL, что соответствует компетенции:

ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов.

В процессе реализации программного комплекса применялись принципы программного управления, которые заключались в применении команд, отвечающих за математические вычисления, применяемые для получения результатов расчетов, что соответствует компетенции:

ОПК-7 Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой.

При изучении предметной области, осуществлялся поиск и обработка необходимой информации, а также представление её в виде информационной и математической моделей программного комплекса, что соответствует компетенции:

ОПК-8 Способен осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий.

При выполнении работы было проведено исследование предметной области и проведена разработка модуля для расчёта остаточных напряжений и деформаций металлоконструкций, что соответствует компетенции:

ПК-1 Способен проводить научно-исследовательские и опытно конструкторские разработки.

Разработанный программный комплекс состоит из модуля визуализации модели в воксельном виде, модуля расчета остаточных напряжений и деформаций, графического интерфейса программного комплекса, что соответствует освоению компетенции:

ПК-2 Способен проводить интеграцию программных модулей и компонент.

При разработке программного комплекса производилось тестирование каждого модуля и сравнение получаемого результата с ожидаемым. Результаты обработки модели сравнивались с результатами обработки аналогов программного комплекса. Также производилось тестирование программного комплекса методом «Белого ящика». Также производилось тестирования производительности на разных устройствах. Производилось тестирование разработанного программного комплекса с использованием параметров и моделей для расчета, исходя из полученных результатов, принималось решение внесения изменений или исправления ошибок программного комплекса, что соответствует компетенции:

ПК-3 Способен разрабатывать тестовые случаи, проводить тестирование и исследовать результаты.

В процессе выполнения работы была составлена математическая и информационная модели для разработки программного, на основе которых реализован программный комплекс для расчета остаточных напряжений и деформаций металлоконструкций, что соответствует компетенции:

ПК-4 Способен создавать и анализировать требования на разработку программно-информационных систем и подсистем.

При выполнении данной работы была написана информационная модель программного комплекса для расчета остаточных напряжений и деформаций металлоконструкций, что соответствует компетенции:

ПК-5 Способен осуществлять концептуальное, функциональное и логическое проектирование программно-информационных систем.

Список литературы

1. **Башаров, Р. Р.** Анализ причин и источников возникновения остаточных напряжений / Башаров Р. Р., Кильметова Л. Р., Старовойтов С. В., Хадиуллин С. Х., Черников П. П. // Вестник УГАТУ, – 2018. – №4 (82).
2. **Бистерфельд, О. А.** Методология функционального моделирования IDEF0: учебно-методическое пособие / О. А. Бистерфельд – Ряз. гос. ун-т им. С. А. Есенина. — Рязань, 2008. — 48 с.
3. **Богданович, В. И.** Математическое моделирование процессов плавления порошка в технологии селективного лазерного сплавления / В. И. Богданович, М. Г. Гиорбелидзе, А. В. Сотов, Н. Д. Проничев, В. Г. Смелов, А. Агаповичев // Известия Самарского научного центра РАН. – 2017. – №4-1
4. **Ботвенко, С. И**. Пространственное распределение термических остаточных напряжений в пластине / Ботвенко С. И. // Вестник ИрГТУ. – 2013. – №12 (83).
5. **Васильева, А. О.** Что такое диаграмма Ганта? / А. О. Васильева, Е. О. Васильева // Редколлегия. – 2018. –168 с.
6. **Вехов, А. С.** Применение аддитивных технологий в современном производстве / А. С. Вехов, С. А. Титаренко // Решетневские чтения. – 2018.
7. **Гаврюшин, С. С.** Биомеханическое моделирование индивидуализированных имплантируемых изделий для реконструктивной хирургии / Гаврюшин С. С., Утенков В. М., Хрыков С. С. // Инженерный журнал: наука и инновации. – 2017. – №2 (62).
8. **Гончарова, О. Н.** Аддитивные технологии - динамично развивающееся производство / О. Н. Гончарова, Ю. М. Бережной, Е. Н. Бессарабов, Е. А. Кадамов, Т. М. Гайнутдинов, Е. М. Нагопетьян, В. М. Ковина // ИВД. – 2016. – №4 (43).
9. **Гордеев, Г. А.** Компьютерное моделирование селективного лазерного плавления высокодисперсных металлических порошков / Г. А. Гордеев, М.

Д. Кривилев, В. Е. Анкудинов // Вычислительная механика сплошных сред – Computational continuum mechanics, Т. 10, № 3, – 2017. – с. 293-312

10. **Гребенщикова, Т. Д.** Особенность 3d-печати из титана / Т. Д. Гребенщикова, М. Н. Краснова // The Scientific Heritage. – 2019. – №42-1 (42).
11. **Гришин, А. В.** Технология селективного лазерного спекания (SLS) // Научные исследования. – 2017. – №7 (18)
12. **Гусаров, А. В.** Расчёт остаточных напряжений при селективном лазерном плавлении порошков / А. В. Гусаров, И. С. Малахова-Зяблова, А. В. Пересторонина // Физика прочности и пластичности. – 2013. – № 11, с. 1501 – 1516
13. **Добрынин, А. С.** Формирование расписаний в задачах временного планирования / А. С. Добрынин, С. М. Кулаков, Р. С. Койнов, А. В. Грачёв // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. – 2014. – №. 4 – с. 103-109.
14. **Дынин, Н. В.** Влияние параметров процесса селективного лазерного сплавления на структуру алюминиевого сплава системы Al-Si-Mg / Н. В. Дынин, А. В. Заводов, М. С. Оглодков, Д. В. Хасиков // Труды ВИАМ. – 2017. – №10 (58).
15. **Ехлаков, Ю. П.** Управление программными проектами: учебник / Ю. П. Ехлаков. // – Томск: Изд-во Томск. гос. ун-та систем управления и радиоэлектроники, – 2015. – 216 с.
16. **Зайдес, С. А.** Механика формирования остаточных напряжений при поверхностном пластическом деформировании на основе динамического моделирования / С. А. Зайдес, А. В. Колесник // Вестник ИрГТУ. – 2017. – №1 (120).
17. **Каблов, Е. Н.** Металлопорошковые композиции жаропрочного сплава ЭП648 производства фгуп «ВИАМ» ГНЦ РФ в технологиях селективного лазерного сплавления, лазерной газопорошковой наплавки и высокоточного литья полимеров, наполненных металлическими порошками / Е. Н.

Каблов, А. Г. Евгенов, // Известия вузов. Машиностроение. – 2016. – №9 (678).

18. **Каратушин, С. И.** Моделирование и расчет остаточных напряжений в прокатных профилях / С. И. Каратушин, Д. А. Храмова, Н. А. Бильдюк // Известия вузов. Машиностроение. – 2017. – №6 (687).
19. **Каратушин, С. И.** Сравнительный анализ расчетных методов определения остаточных напряжений / С. И. Каратушин, Д. В. Спиридонов, Ю. А. Плешанова // Металлообработка. – 2016. – №4 (94).
20. **Ключко, А. Д.** Аддитивные технологии и эффективность их использования в производстве / А. Д. Ключко, Г. А. Гареева, Д. Р. Григорьева // Символ науки. – 2018. – №1-2.
21. **Копп, А. М.** Разработка подхода к анализу и оптимизации диаграмм потоков данных / А. М. Копп, Д. Л. Орловский. // ScienceRise - № 7, – 2017. – 33-42 с.
22. **Кривилев, М. Д.** Управление лазерным спеканием металлических порошковых смесей / М. Д. Кривилев, Е. В. Харанжевский, Г. А. Гордеев, В. Е. Анкудинов // Управление большими системами, Вып. 31, – 2010. – С. 299-322
23. **Курбатов, А. С.** Анализ задачи потери устойчивости тонкостенных конструкций, выполненных методом селективного лазерного спекания, при интенсивном нагреве / Курбатов А. С., Орехов А. А., Рабинский Л. Н. // Известия ТулГУ. Технические науки. – 2018. – №12
24. **Ландау, Л. Д.** Теория упругости / Л. Д. Ландау, Е. М. Лифшиц // Теоретическая физика – М.: Наука, – 1987. – №4. – 248 с.
25. **Осколков, А. А.** Передовые технологии аддитивного производства металлических изделий / А. А. Осколков, Е. В. Матвеев, И. И. Безукладников, Д. Н. Трушников, Е.Л. Кротова //Вестник ПНИПУ. Машиностроение, материаловедение. – 2018. – №3.

26. **Сметанников, О. Ю.** Исследование влияния параметров процесса 3D-наплавки проволочных материалов на формирование остаточных деформаций / Сметанников О. Ю., Максимов П. В., Трушников Д. Н., Пермяков Г. Л., Беленький В. Я., Фарберов А. С. // Вестник ПНИПУ. Механика. – 2019. – №2

27. **Султанова, Ф. Р.** Технология селективного лазерного спекания (SLS) / Ф. Р. Султанова, И. Э. Нам, С. Б. Мирзахакимов // Инновационная наука. – 2016. – №10-2.

28. **Тасваева, А. Н.** Диаграммы потоков данных и вариантов использования как инструменты проектирования информационных систем / А.Н. Тасваева // Модели, системы, сети в экономике, технике, природе и обществе. – 2012. – №2 (3).

29. **Ankudinov, V. E.** Numerical simulation of heat transfer and melting of Fe-based powders in SLM processing / V. E. Ankudinov, G. A. Gordeev, M. D. Krivilyov // IOP Conference Series: Materials Science and Engineering, SPTM-2017 conference. Vol. 192, Iss. 1, N art. 012026, – 2017. – 7 p.

30. **Gordeev, G. A.** Numerical simulation of selective laser melting with local powder shrinkage using FEM with the refined mesh / G. A. Gordeev, V. Ankudinov, E. V. Kharanzhevskiy, M. D. Krivilyov // European Physical Journal: Special Topics, Vol. 229, No. 2-3, 2020. Pp. 205-216.

31. **Gordeev, G. A.** Optimization of processing parameters in laser sintering of metallic powders / G. A. Gordeev, V. E. Ankudinov, M. D. Krivilyov, E. V. Kharanzhevskiy // IOP Conference Series: Materials Science and Engineering, 2011, Vol. 27, Iss. 1, N art. 012079, – 2011. – 7 p.

32. **Kruth, J. P.** Selective laser melting of iron-based powder. / J. P. Kruth, L. Froyen, J. Van Vaerenbergh, P. Mercelis, M. Rombouts, B. Lauwers //Journal of Materials Processing Technology, – 149(1-3), – 2004.

33. **Gerner, F.** Effect of Thermal Deformation on Part Errors in Metal Powder Based Additive Manufacturing Processes / P. Ratnadeep, A. Sam, F. Gerner// European Physical Journal: Special Topics, – 136(1-3), – 2014.

34. **Shutov, I. V.** Analysis of morphology and residual porosity in selective laser melting of Fe powders using single track experiments / I. V. Shutov, G. A. Gordeev, E. V. Kharanzhevskiy, M. D. Krivilyov // IOP Conference Series: Materials Science and Engineering, SPTM-2017 conference. Vol. 192, Iss. 1, N art. 012023, – 2017. – 10 p.
35. **Ushakova, E. S.** Modeling of the stress-strain state of rocket-space technology structural elements manufactured by using additive technologies / E. S. Ushakova // Сибирский журнал науки и технологий. – 2019. – №2.
36. **Wang, P.** Research on the fabricating quality optimization of the overhanging surface in SLM process. / P. Wang, D. Yang, Y. Yongqiang, S. Ziheng//The International Journal of Advanced Manufacturing Technology, – 65(9-12), – 2013.
37. **Wray, P.** Additive Manufacturing: Turning Manufacturing Inside Out, / Wray P. // Amer. Ceram. Soc. Bull., 2014, vol. 93, no. 3.

Приложение А

(обязательное)

Листинг программного комплекса CoRSaD

Листинг А.1 – Класс VisualVoxel

```
public class  VisualVoxel
{
    private OpenGL gl;
    private int[] voxelX;
    private int[] voxelY;
    private int[] voxelZ;


    private Voxel[] voxels;

    private int countVoxel;

    private int[,] voxelXYZ;

    private ColorVoxel colorTemperature = new ColorVoxel(0, 1000);
    private ColorVoxel colorDeformationX = new ColorVoxel(-245, 245);
    private ColorVoxel colorDeformationY = new ColorVoxel(-245, 245);
    private ColorVoxel colorDeformationZ = new ColorVoxel(-245, 245);


    public HeatEquation getHeatEquation { get => heatEquation; }

    public HeatEquation HeatEquation { get => heatEquation; set =>
heatEquation = value; }
    ScanningModel scanningModel;
    MeshVoxel[,,] meshVoxels;
    public VisualVoxel()
    {

    }
    public void optionsToHeatEquation(Options options)
    {
       // ImportXYZ();
        heatEquation   =   new   HeatEquation(options,   scanning-
Model.getMaxX(),scanningModel.getMaxY(),scanningModel.getMaxZ() );
    }

    double t = 0;



   private   CalculationStress  calculationStress  =  new  Calcula-
tionStress();

    Options options;
    HeatEquation heatEquation;
```

```
        //визуализация трехмерного уравнения теплопроводности
        public void VisualizationTemperatyre3(OpenGL gl, OpenGL gl2, bool
boolPauseCalc,int countSpeedStep)
        {
            double temperature = 0;

            meshVoxels = scanningModel.getAllMeshVoxels();
            heatEquation.setScaningVoxels(scanningModel.SnakeScan-
ning());

            if (boolPauseCalc)
            {

                if (heatEquation.CountScanningVoxels<heatEquation.Scan-
ingVoxels.Length)
                    for (int i = 0; i < countSpeedStep; i++)
                    {
                        heatEquation.CalculationHeatEquation();
                    }
            }


            for (int z = scanningModel.getMinZ(); z <= scanning-
Model.getMaxZ(); z++)
            {
                for (int x = scanningModel.getMinX(); x <= scanning-
Model.getMaxX(); x++)
                {
                    for (int y = scanningModel.getMinY(); y <= scanning-
Model.getMaxY(); y++)
                    {

                        //temperature = CalculationStress.AnalyticalSo-
lution(x+1,y+1,z+1,t);
                        temperature = heatEquation.getTemperature(x, y,
z);//x+1,y+1,z+1 так как в массиивк етсь 0 значения
                        colorTemperature.ColorStressVoxel(tempera-
ture);//разскоментить когда расчет сделаем
                        colorDeformationX.ColorDeformation-
Voxel(heatEquation.getDeformationX(x, y, z));
                        colorDeformationY.ColorDeformation-
Voxel(heatEquation.getDeformationY(x, y, z));
                        colorDeformationZ.ColorDeformation-
Voxel(heatEquation.getDeformationZ(x, y, z));

                        if (heatEquation.boolMelted(x, y, z))
                        {
                            drawVoxel(gl,    colorTemperature,    mesh-
Voxels[x, y, z]);
                            drawVoxel(gl2,
                                      colorDeformationX,
                                      colorDeformationY,
                                      colorDeformationZ,
                                      meshVoxels[x, y, z]);

                        }
                        else
                        {
```

```
                                drawSkeleton(gl,   colorTemperature,   mesh-
Voxels[x, y, z]);
                                drawSkeleton(gl2,  colorTemperature,  mesh-
Voxels[x, y, z]);
                            }
                        }
                    }
                }

            }

            public double getTimeHeatEquation()
            {
              return  heatEquation.getTime();
            }



            int maxX = 1;
            int maxY =1;
            int maxZ = 1;

            //convert_Data_from_FileXYZ  получает  на  вход  необаботанную
строку возвращает три масссива координат вокселей
            private void convert_Data_from_FileXYZ(string fileData)
            {
                string[] words = fileData.Split(' ');
                 this.countVoxel = words.Length / 3;

                Console.WriteLine(countVoxel);

                voxels = new  Voxel[countVoxel];
                voxelXYZ = new int[3, countVoxel];
                //voxelX = new int[k];
                //voxelY = new int[k];
                //voxelZ = new int[k];

                int j = 0;
                for (int i = 0; i < countVoxel; i += 1)
                {

                    voxels[i] = new  Voxel(Convert.ToInt32(words[j]),
                                           Convert.ToInt32(words[j + 1]),
                                           Convert.ToInt32(words[j + 2]) );
                    //voxelXYZ[0, j] = Convert.ToInt32(words[i]);
                    //voxelXYZ[1, j] = Convert.ToInt32(words[i + 1]);
                    //voxelXYZ[2, j] = Convert.ToInt32(words[i + 2]);

                    //voxelX[j] = Convert.ToInt32(words[i + 0]);
                    //voxelY[j] = Convert.ToInt32(words[i + 1]);
                    //voxelZ[j] = Convert.ToInt32(words[i + 2]);
                    j+=3;
                }

                                                        //           Con-
sole.WriteLine(voxelXYZ);
            }
            public  void ImportXYZ(StreamReader sr)
```

```
        {
            //string  path  =  @"D:\\StudentData\\VKR-2021\\Вокселизация
змеюка\\voxel pyton";//заменить на патч
            //StreamReader sr = new StreamReader($"{path}\\tor.xyz");

            string numbers = sr.ReadToEnd();
            numbers = numbers.Replace("\r", "").Replace("\n", " ");
            convert_Data_from_FileXYZ(numbers);


            scanningModel   =   new   ScanningModel(voxels);//выполняется
каждый кадр , нужно  упростить

        }
        private void drawPolygon(OpenGL gl, float r, float g, float b,
float x1, float y1, float z1,
                                                                    float
x2, float y2, float z2,
                                                                    float
x3, float y3, float z3,
                                                                    float
x4, float y4, float z4)
        {
            gl.Begin(OpenGL.GL_POLYGON);
            gl.Color(r, g, b);
            gl.Vertex(x1, y1, z1);
            gl.Vertex(x2, y2, z2);
            gl.Vertex(x3, y3, z3);
            gl.Vertex(x4, y4, z4);
            gl.End();
        }

        //рисуем воксель по точке(центр куба) обект gl ,входные rгб цвет,
координаты центров

        private void drawVoxel(OpenGL gl, ColorVoxel color, MeshVoxel
voxel /*float xc, float yc, float zc*/)
        {
            if (voxel.getBoolScanned())
            {
                drawSkeleton(gl,color,voxel);

                float sizeVoxel = 1;
                float hfs = sizeVoxel / 2;
                float dsqrt = (float)Math.Sqrt(hfs * hfs);
                //рисование полигонов
                gl.Begin(OpenGL.GL_QUADS);
                gl.Color(color.getRed(),  color.getGreen(),  color.get-
Blue(),255);

                float xc = voxel.getVoxelX();
                float yc = voxel.getVoxelY();
                float zc = voxel.getVoxelZ();

                //верхняя грань
                gl.Vertex(xc - dsqrt, yc + hfs, zc - dsqrt);
                gl.Vertex(xc + dsqrt, yc + hfs, zc - dsqrt);
                gl.Vertex(xc + dsqrt, yc + hfs, zc + dsqrt);
```

```
                gl.Vertex(xc - dsqrt, yc + hfs, zc + dsqrt);


                //нижняя грань
                gl.Vertex(xc - dsqrt, yc - hfs, zc - dsqrt);
                gl.Vertex(xc + dsqrt, yc - hfs, zc - dsqrt);
                gl.Vertex(xc + dsqrt, yc - hfs, zc + dsqrt);
                gl.Vertex(xc - dsqrt, yc - hfs, zc + dsqrt);


                //левая грань
                gl.Vertex(xc - hfs, yc - dsqrt, zc - dsqrt);
                gl.Vertex(xc - hfs, yc + dsqrt, zc - dsqrt);
                gl.Vertex(xc - hfs, yc + dsqrt, zc + dsqrt);
                gl.Vertex(xc - hfs, yc - dsqrt, zc + dsqrt);


                //правая грань
                gl.Vertex(xc + hfs, yc - dsqrt, zc - dsqrt);
                gl.Vertex(xc + hfs, yc + dsqrt, zc - dsqrt);
                gl.Vertex(xc + hfs, yc + dsqrt, zc + dsqrt);
                gl.Vertex(xc + hfs, yc - dsqrt, zc + dsqrt);

                //передняя грань
                gl.Vertex(xc - dsqrt, yc - dsqrt, zc + hfs);
                gl.Vertex(xc + dsqrt, yc - dsqrt, zc + hfs);
                gl.Vertex(xc + dsqrt, yc + dsqrt, zc + hfs);
                gl.Vertex(xc - dsqrt, yc + dsqrt, zc + hfs);

                //задняя грань
                gl.Vertex(xc - dsqrt, yc - dsqrt, zc - hfs);
                gl.Vertex(xc + dsqrt, yc - dsqrt, zc - hfs);
                gl.Vertex(xc + dsqrt, yc + dsqrt, zc - hfs);
                gl.Vertex(xc - dsqrt, yc + dsqrt, zc - hfs);

                gl.End();
                //gl.Flush();
                //рисование контуров


                gl.Finish();
            }
        }
        private void drawVoxel(OpenGL gl, ColorVoxel colorX,ColorVoxel
colorY,ColorVoxel colorZ, MeshVoxel voxel /*float xc, float yc, float zc*/)
        {
            if (voxel.getBoolScanned())
            {
                drawSkeleton(gl, colorX, voxel);

                float sizeVoxel = 1;
                float hfs = sizeVoxel / 2;
                float dsqrt = (float)Math.Sqrt(hfs * hfs);
                //рисование полигонов
                float xc = voxel.getVoxelX();
                float yc = voxel.getVoxelY();
                float zc = voxel.getVoxelZ();
                ///////
```

```
                    drawSkeleton(gl, colorX, voxel);
                    gl.Begin(OpenGL.GL_QUADS);
                    gl.Color(colorX.getRed(), colorX.getGreen(), colorX.get-
Blue(), 255);

                    //левая грань
                    gl.Vertex(xc - hfs, yc - dsqrt, zc - dsqrt);
                    gl.Vertex(xc - hfs, yc + dsqrt, zc - dsqrt);
                    gl.Vertex(xc - hfs, yc + dsqrt, zc + dsqrt);
                    gl.Vertex(xc - hfs, yc - dsqrt, zc + dsqrt);


                    //правая грань
                    gl.Vertex(xc + hfs, yc - dsqrt, zc - dsqrt);
                    gl.Vertex(xc + hfs, yc + dsqrt, zc - dsqrt);
                    gl.Vertex(xc + hfs, yc + dsqrt, zc + dsqrt);
                    gl.Vertex(xc + hfs, yc - dsqrt, zc + dsqrt);

                    gl.End();
                    /////////
                    gl.Begin(OpenGL.GL_QUADS);
                    gl.Color(colorY.getRed(), colorY.getGreen(), colorY.get-
Blue(), 255);

                    //верхняя грань
                    gl.Vertex(xc - dsqrt, yc + hfs, zc - dsqrt);
                    gl.Vertex(xc + dsqrt, yc + hfs, zc - dsqrt);
                    gl.Vertex(xc + dsqrt, yc + hfs, zc + dsqrt);
                    gl.Vertex(xc - dsqrt, yc + hfs, zc + dsqrt);


                    //нижняя грань
                    gl.Vertex(xc - dsqrt, yc - hfs, zc - dsqrt);
                    gl.Vertex(xc + dsqrt, yc - hfs, zc - dsqrt);
                    gl.Vertex(xc + dsqrt, yc - hfs, zc + dsqrt);
                    gl.Vertex(xc - dsqrt, yc - hfs, zc + dsqrt);
                    gl.End();

                    gl.Begin(OpenGL.GL_QUADS);
                    gl.Color(colorZ.getRed(), colorZ.getGreen(), colorZ.get-
Blue(), 255);
                    //передняя грань
                    gl.Vertex(xc - dsqrt, yc - dsqrt, zc + hfs);
                    gl.Vertex(xc + dsqrt, yc - dsqrt, zc + hfs);
                    gl.Vertex(xc + dsqrt, yc + dsqrt, zc + hfs);
                    gl.Vertex(xc - dsqrt, yc + dsqrt, zc + hfs);

                    //задняя грань
                    gl.Vertex(xc - dsqrt, yc - dsqrt, zc - hfs);
                    gl.Vertex(xc + dsqrt, yc - dsqrt, zc - hfs);
                    gl.Vertex(xc + dsqrt, yc + dsqrt, zc - hfs);
                    gl.Vertex(xc - dsqrt, yc + dsqrt, zc - hfs);

                    gl.End();
                    //gl.Flush();
                    //рисование контуров
```

```
                gl.Finish();
            }
        }
        private void drawSkeleton(OpenGL gl, ColorVoxel color, MeshVoxel
voxel /*float xc, float yc, float zc*/)
        {
            if (voxel.getBoolScanned())
            {

                float sizeVoxel = 1;
                float hfs = sizeVoxel / 2;
                float dsqrt = (float)Math.Sqrt(hfs * hfs);
                //рисование полигонов

                float xc = voxel.getVoxelX();
                float yc = voxel.getVoxelY();
                float zc = voxel.getVoxelZ();
                //gl.Flush();
                //рисование контуров

                gl.Begin(OpenGL.GL_LINES);
                gl.Color(0, 0, 0);
             //    gl.LineWidth(1000000);
                // glColor3d(0, 1, 0);
                // glBegin(GL_LINE_STRIP);
                //верхняя грань


                gl.Vertex(xc - dsqrt, yc + hfs, zc - dsqrt);
                gl.Vertex(xc + dsqrt, yc + hfs, zc - dsqrt);

                gl.Vertex(xc + dsqrt, yc + hfs, zc + dsqrt);
                gl.Vertex(xc - dsqrt, yc + hfs, zc + dsqrt);


                //нижняя грань
                gl.Vertex(xc - dsqrt, yc - hfs, zc - dsqrt);
                gl.Vertex(xc + dsqrt, yc - hfs, zc - dsqrt);

                gl.Vertex(xc + dsqrt, yc - hfs, zc + dsqrt);
                gl.Vertex(xc - dsqrt, yc - hfs, zc + dsqrt);


                //левая грань
                gl.Vertex(xc - hfs, yc - dsqrt, zc - dsqrt);
                gl.Vertex(xc - hfs, yc + dsqrt, zc - dsqrt);

                gl.Vertex(xc - hfs, yc + dsqrt, zc + dsqrt);
                gl.Vertex(xc - hfs, yc - dsqrt, zc + dsqrt);

                //правая грань
                gl.Vertex(xc + hfs, yc - dsqrt, zc - dsqrt);
                gl.Vertex(xc + hfs, yc + dsqrt, zc - dsqrt);

                gl.Vertex(xc + hfs, yc + dsqrt, zc + dsqrt);
                gl.Vertex(xc + hfs, yc - dsqrt, zc + dsqrt);

                //передняя грань
```

```
                gl.Vertex(xc - dsqrt, yc - dsqrt, zc + hfs);
                gl.Vertex(xc + dsqrt, yc - dsqrt, zc + hfs);

                gl.Vertex(xc + dsqrt, yc + dsqrt, zc + hfs);
                gl.Vertex(xc - dsqrt, yc + dsqrt, zc + hfs);

                //задняя грань
                gl.Vertex(xc - dsqrt, yc - dsqrt, zc - hfs);
                gl.Vertex(xc + dsqrt, yc - dsqrt, zc - hfs);

                gl.Vertex(xc + dsqrt, yc + dsqrt, zc - hfs);
                gl.Vertex(xc - dsqrt, yc + dsqrt, zc - hfs);

                //передняя грань
                gl.Vertex(xc - dsqrt, yc - dsqrt, zc - hfs);
                gl.Vertex(xc - dsqrt, yc - dsqrt, zc + hfs);

                gl.Vertex(xc + dsqrt, yc + dsqrt, zc - hfs);
                gl.Vertex(xc + dsqrt, yc + dsqrt, zc + hfs);

                //задняя грань
                gl.Vertex(xc - dsqrt, yc + dsqrt, zc - hfs);
                gl.Vertex(xc - dsqrt, yc + dsqrt, zc + hfs);

                gl.Vertex(xc + dsqrt, yc - dsqrt, zc - hfs);
                gl.Vertex(xc + dsqrt, yc - dsqrt, zc + hfs);

                gl.End();
                gl.Finish();
            }
        }
    }
```

Листинг А.2 – Класс HeatEquation

```
    public class HeatEquation
    {
        private static double D_dust = 0.24,//сплавленного материала
                              D_air=0.01,
                              D_metal=0.6,

                              dx = 1,//шаг
                              dy = 1,
                              dz = 1;

        private double t = 0,
                       dt = (dx * dx) / (2) * T_start,//tay
                       t_laser_voxel  =  0.02,//время  обработки  одного
вокселя лазером
                       t_curent_laser_voxel = 0;//время для таймера

        private static double T_start = 20,//температур окружающей среды
                              T_laser = 600,
                              T_fusion_titan=1600,
                              T_fusion_metal;
```

```
        private static int
                        a ,b ,N,I,J,K;

        double[,,] T_Next;
        double[,,] T_Curent;
        double[,,] K_Values;
        bool[,,] boolPrinted;//ячейки которые которые прошли обработку

        int[][] scaningVoxels;
        int countScanningVoxels=0;

        private Deformations deformation;
        public Deformations getDeformation { get => deformation; }
        public int[][] ScaningVoxels { get => scaningVoxels; set => scan-
ingVoxels = value; }
        public int CountScanningVoxels { get => countScanningVoxels; set
=> countScanningVoxels = value; }

        public  HeatEquation(Options  option,int  Icount,int  Jcount,int
Kcount)
        {
            a = 0;
           // b = 59;//заменить на voxelmaxXYZ
            I = (int)(((double)(Icount - a)) / dx) + 2;
            J = (int)(((double)(Jcount - a)) / dx) + 2;
            K = (int)(((double)(Kcount - a)) / dx) + 2;

            T_Next = new double[I,J,K];
            T_Curent = new double[I,J,K];//4измерение время
            K_Values = new double[I, J, K];
            boolPrinted = new bool[I,J,K];
            //вводим воксели по порядку сканировнаия для источника
            D_air = option.get_D_air;
            D_metal = option.get_D_metal;
            T_start = option.get_T_start;//температур окружающей среды
            T_laser = option.get_T_laser;
            T_fusion_metal = option.get_T_fusion_metal;
            t_laser_voxel = option.get_t_laser_voxel;
            // Инициализация массивов в соответсвии с начальными услови-
ями

            Parallel.For(0, K,
             k =>
             //for (int k = 0; k < K; k++)
             {
               for (int j = 0; j < J; j++)
               {
                   for (int i = 0; i < I; i++)
                   {
                       T_Curent[i, j, k] = T_start;//начальная всех эле-
ментов и окружающей среды

                       K_Values[i, j, k] = D_dust;


                       if (i == 0 || i == I - 1)
                           K_Values[i, j, k] = D_air;
```

```
                        if (j == 0 || j == J - 1)
                            K_Values[i, j, k] = D_air;
                        if (k == 0 || k == K - 1)
                            K_Values[i, j, k] = D_air;

                        boolPrinted[i, j, k] = false;
                    }
                }
            });

            deformation = new Deformations(option,dx,I,J,K);

        }
        public void setScaningVoxels(int[][] scaningVoxels){
             this.scaningVoxels = scaningVoxels;
        }

        public void CalculationHeatEquation(){
            //Указываем источники их начальную температуру и время из
водействия

            //Расчет следующего значения температуры в сетке
            Parallel.For(1, K - 1,
            k =>
            //for (int k = 1; k < K - 1; k++)
            {
                for (int j = 1; j < J - 1; j++)
                {
                    for (int i = 1; i < I - 1; i++)
                    {
                        T_Next[i, j, k] = T_Curent[i, j, k] + (K_Val-
ues[i,j,k]*dt / (dx * dx)) * (
                                                                K_half_plus_i(i,   j,
k) * (T_Curent[i + 1, j, k] - T_Curent[i, j, k])
                                                               - K_half_minus_i(i, j,
k) * (T_Curent[i, j, k] - T_Curent[i - 1, j, k])
                                                               +
                                                               (K_half_plus_j(i, j, k)
* (T_Curent[i, j + 1, k] - T_Curent[i, j, k])
                                                               - K_half_minus_j(i, j,
k) * (T_Curent[i, j, k] - T_Curent[i, j - 1, k]))
                                                               +
                                                                (K_half_plus_k(i,   j,
k) * (T_Curent[i, j, k + 1] - T_Curent[i, j, k])
                                                               - K_half_minus_k(i, j,
k) * (T_Curent[i, j, k] - T_Curent[i, j, k - 1]))
                                                               )
                                                               + Q_source(t, i, j, k);

                        //изменение агрегатного состояния (порошок пре-
вратился в металл)
                        if (T_Next[i, j, k] > T_fusion_titan)
                        {
                            K_Values[i, j, k] = D_metal;
                            boolPrinted[i, j, k] = true;
                        }
                    }
                }
```

```
    });

    //перезаписываем граничные условия в next
   Parallel.For(0, K,
   k =>
   //for (int k = 0; k < K; k++)
   {
        for (int j = 0; j < J; j++)
        {
            for (int i = 0; i < I; i++)
            {
                if (i == 0 || i == I - 1)
                    T_Next[i, j, k] = T_start;
                if (j == 0 || j == J - 1)
                    T_Next[i, j, k] = T_start;
                if (k == 0 || k == K - 1)
                    T_Next[i, j, k] = T_start;
            }
        }
    });
    /////

    deformation.CalculationDeformations(T_Next, boolPrinted);


    ////
                T_Curent/*[i, j, k]*/ = T_Next/*[i, j, k]*/;



    //измененение шага по времени для устойчивости
    dt = (dx * dx) / (2 * Tmax()) ;//tay
    t += dt;
    // Console.WriteLine(T_Curent[N-1, N-1, N-1]);
}


public double Tmax(){
    double max=0;

    foreach(double t in T_Curent)
    {
        max = t > max ? t : max;

    }
    return max;
}
//метод определения источника
public double Q_source(double t,int i, int j, int k)
{

    double q = 0;
    if(countScanningVoxels< scaningVoxels.Length)
        if(scaningVoxels[countScanningVoxels][0]+1==i)
            if(scaningVoxels[countScanningVoxels][1]+1==j)
                if(scaningVoxels[countScanningVoxels][2]+1==k)
                    if (t-t_curent_laser_voxel < t_laser_voxel)
```

```
                    {
                        q = T_laser;
                    }
                    else
                    {
                        t_curent_laser_voxel = t;
                        countScanningVoxels += 1;
                    }

        return q;
    }

    public double K_half_plus_i( int i, int j, int k)
    {
        double K_half = 0;

        K_half =  (T_Curent[ i, j, k] + T_Curent[ i + 1, j, k]) / 2;
        return K_half;
    }
    public double K_half_minus_i( int i, int j, int k)
    {
        double K_half = 0;

        K_half =  (T_Curent[i, j, k] + T_Curent[ i - 1, j, k]) / 2;
        return K_half;
    }
    public double K_half_plus_j( int i, int j, int k)
    {
        double K_half = 0;

        K_half= (T_Curent[ i, j, k] + T_Curent[ i, j+1, k]) / 2;
        return K_half;
    }
    public double K_half_minus_j( int i, int j, int k)
    {
        double K_half = 0;

        K_half = (T_Curent[ i, j, k] + T_Curent[ i, j-1, k]) / 2;
        return K_half;
    }
    public double K_half_plus_k( int i, int j, int k)
    {
        double K_half = 0;

        K_half= (T_Curent[ i, j, k] + T_Curent[ i , j, k + 1]) / 2;
        return K_half;
    }
    public double K_half_minus_k( int i, int j, int k)
    {
        double K_half = 0;

        K_half = (T_Curent[ i, j, k] + T_Curent[ i , j, k - 1]) / 2;
        return K_half;
    }
    public double getTemperature(int x, int y, int z)
    {
        return T_Curent[x,y,z];
    }
```

```
        public double getDeformationX(int x, int y, int z)
        {
            return deformation.getDeformationX(x,y,z);
        }
        public double getDeformationY(int x, int y, int z)
        {
            return deformation.getDeformationY(x, y, z);
        }
        public double getDeformationZ(int x, int y, int z)
        {
            return deformation.getDeformationZ(x, y, z);
        }
        public double getTime()
        {
            return t;
        }


        public int toNormalization(int ijk)
        {
            return ijk * 10;
        }

        //если вокслель расплавился то поменялся коэфциент теплопровод-
ности ()
        public bool boolMelted(int i ,int j, int k) //расплавился воксель
или нет
        {
            return boolPrinted[i, j, k];
        }
    }
```

Листинг А.3 – Класс Deformations

```
    public class Deformations
    {
        //норм видно при  alpha_0 = 0.3; E_crit=-245; куб 10на10


        private double[,,] Ex_Curent;
        private double[,,] Ey_Curent;
        private double[,,] Ez_Curent;

        private double[,,] Ex_Next;
        private double[,,] Ey_Next;
        private double[,,] Ez_Next;

        private double[,,] alpha_cup;
        private double alpha_0 ;//коэфициент теплового расшире


        private double E_crit;//изменить на значение
        private int N,I,J,K;
        private double dx;
```

```
        private double T_start;

        public Deformations(Options options, double dx,int I,int J,int
K)
        {
            this.I = I;
            this.J = J;
            this.K = K;
            this.dx = dx;
            T_start = options.get_T_start;
            E_crit = options.get_E_crit;
            alpha_0 = options.get_Alpha_0;

            Ex_Curent = new double[I, J, K];
            Ey_Curent = new double[I, J, K];
            Ez_Curent = new double[I, J, K];

            Ex_Next = new double[I,J,K];
            Ey_Next = new double[I,J,K];
            Ez_Next = new double[I, J, K];

            alpha_cup = new double[I, J, K];

            Parallel.For(0, K,
            k =>
            //for (int k = 0; k < N; k++)
            {
                for (int j = 0; j <J; j++)
                {
                    for (int i = 0; i < I; i++)
                    {
                        Ex_Curent[i, j, k] = 0;
                        Ey_Curent[i, j, k] = 0;
                        Ez_Curent[i, j, k] = 0;

                        Ex_Next[i, j, k] = 0;
                        Ey_Next[i, j, k] = 0;
                        Ez_Next[i, j, k] = 0;

                        alpha_cup[i, j, k] = 0;
                    }
                }
            });



        }
        public void CalculationDeformations(double[,,]  T)
            {

                for (int k = 0; k < N; k++)
                {
                    for (int j = 0; j < N; j++)
                    {
                        for (int i = 0; i < N; i++)
                        {
                            alpha_cup[i,j,k]  =  alpha_0  *  T[i,j,k]  -
T_start;
```

```
                            if (i == 0 || i == N - 1)
                                alpha_cup[i, j, k] = 0;
                            if (j == 0 || j == N - 1)
                                alpha_cup[i, j, k] = 0;
                            if (k == 0 || k == N - 1)
                                alpha_cup[i, j, k] = 0;
                        }
                    }
                }

                for (int k = 1; k < N - 1; k++)
                {
                    for (int j = 1; j < N - 1; j++)
                    {
                        for (int i = 1; i < N - 1; i++)
                        {
                            Ex_Next[i, j, k] = -dx * ((alpha_cup[i + 1,
j, k] - alpha_cup[i - 1, j, k]) / 2);
                            Ey_Next[i, j, k] = -dx * ((alpha_cup[i, j +
1, k] - alpha_cup[i, j - 1, k]) / 2);
                            Ez_Next[i, j, k] = -dx * ((alpha_cup[i, j, k
+ 1] - alpha_cup[i, j, k - 1]) / 2);


                            if (Math.Abs(Ex_Curent[i, j, k]) > E_crit)
                                if   (Math.Abs(Ex_Next[i,   j,   k])   <
Math.Abs(Ex_Curent[i, j, k]))
                                    Ex_Next[i, j, k] = Ex_Curent[i, j,
k];


                            if (Math.Abs(Ey_Curent[i, j, k]) > E_crit)
                                if   (Math.Abs(Ey_Next[i,   j,   k])   <
Math.Abs(Ey_Curent[i, j, k]))
                                    Ey_Next[i, j, k] = Ey_Curent[i, j,
k];

                            if (Math.Abs(Ez_Curent[i, j, k]) > E_crit)
                                if   (Math.Abs(Ez_Next[i,   j,   k])   <
Math.Abs(Ez_Curent[i, j, k]))
                                    Ez_Next[i, j, k] = Ez_Curent[i, j,
k];
                        }
                    }
                }
                Ex_Curent = Ex_Next;
                Ey_Curent = Ey_Next;
                Ez_Curent = Ez_Next;
            }
        public void CalculationDeformations(double[,,] T, bool[,,] bool-
Printed)
        {
            Parallel.For(0, K,
            k =>
            //for (int k = 0; k < N; k++)
            {
                for (int j = 0; j < J; j++)
```

```
                {
                    for (int i = 0; i < I; i++)
                    {
                        if (boolPrinted[i, j, k])
                            alpha_cup[i, j, k] = alpha_0 * (T[i, j, k] -
T_start);
                        //else
                        //    alpha_cup[i, j, k] = 0;

                    }
                }
            });


            Parallel.For(1, K - 1,
            k =>
            //for (int k = 1; k < N - 1; k++)
            {
                for (int j = 1; j < J - 1; j++)
                {
                    for (int i = 1; i < I - 1; i++)
                    {


                        Ex_Next[i, j, k] = -dx * ((alpha_cup[i + 1, j,
k] - alpha_cup[i - 1, j, k]) / 2);
                        Ey_Next[i, j, k] = -dx * ((alpha_cup[i, j + 1,
k] - alpha_cup[i, j - 1, k]) / 2);
                        Ez_Next[i, j, k] = -dx * ((alpha_cup[i, j, k +
1] - alpha_cup[i, j, k - 1]) / 2);


                        if    (Math.Abs(Ex_Curent[i,    j,    k])    >
Math.Abs(E_crit))
                            if    (Math.Abs(Ex_Next[i,    j,    k])    <
Math.Abs(Ex_Curent[i, j, k]))
                                Ex_Next[i, j, k] = Ex_Curent[i, j, k];



                        if    (Math.Abs(Ey_Curent[i,    j,    k])    >
Math.Abs(E_crit))
                            if    (Math.Abs(Ey_Next[i,    j,    k])    <
Math.Abs(Ey_Curent[i, j, k]))
                                Ey_Next[i, j, k] = Ey_Curent[i, j, k];



                        if    (Math.Abs(Ez_Curent[i,    j,    k])    >
Math.Abs(E_crit))
                            if    (Math.Abs(Ez_Next[i,    j,    k])    <
Math.Abs(Ez_Curent[i, j, k]))
                                Ez_Next[i, j, k] = Ez_Curent[i, j, k];



                    }
```

```
                }
            });


            Parallel.For(0, K, k =>
            //for (int k = 0; k < N; k++)
            {
                for (int j = 0; j < J; j++)
                {
                    for (int i = 0; i < I; i++)
                    {
                        Ex_Curent[i, j, k] = Ex_Next[i, j, k];
                        Ey_Curent[i, j, k] = Ey_Next[i, j, k];
                        Ez_Curent[i, j, k] = Ez_Next[i, j, k];
                    }
                }
            });

            // ExporttoFileCSV();

        }
        public double getDeformationX(int x, int y, int z)
        {
            return Ex_Curent[x,y,z];
        }
        public  void ExportDeformationsToCSV(StreamWriter myStream)
        {

                for (int k = 1; k < K - 1; k++)
                {
                    for (int j = 1; j < J - 1; j++)
                    {
                        for (int i = 1; i < I - 1; i++)
                        {
                            myStream.Write(Ex_Curent[i, j, k].ToString()
+ ";");
                            myStream.Write(Ey_Curent[i, j, k].ToString()
+ ";");
                            myStream.Write(Ez_Curent[i, j, k].ToString()
+ ";\n");

                        }
                    }
                }

        }
        public double getDeformationY(int x, int y, int z)
        {
            return Ey_Curent[x,y,z];
        }
        public double getDeformationZ(int x, int y, int z)
        {
            return Ez_Curent[x,y,z];
        }

    }
```