|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |
| ФАКУЛЬТЕТ «Робототехника и комплексная автоматизация»  КАФЕДРА «Компьютерные системы автоматизации производства»  **РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ  ***НА ТЕМУ:***  «Автоматизированная система мониторинга воспринимаемых металлоконструкцией нагрузок»  Студент РК9-72Б Ю.А. Бетуганов  (Подпись, дата)  Руководитель КП И.А. Мещихин  (Подпись, дата)  Нормоконтролер С. Ю. Дудников  (Подпись, дата)  2023 г. | |
|  | |

**РЕФЕРАТ**

В ходе настоящей работы потребуется создать такую систему, которая будет определять силы, действующие на индивидуализированную металлоконструкции и полученные данные предоставлять пользователю для анализа.

Цель проекта:

Разработать автоматизированную систему мониторинга воспринимаемых металлоконструкцией нагрузок.

Задачи:

1. Разработка модели конструкции для расчёта напряжённого деформированного состояния;
2. Разработка суррогатной модели конструкции для оценки напряженно деформированного состояния в процессе эксплуатации;
3. Разработать программу для определения лучших точек для крепления датчиков;
4. Разработать программу для визуального отображения результатов физического воздействия на металлоконструкцию.

**СОДЕРЖАНИЕ**

[РЕФЕРАТ 2](#_heading=h.3znysh7)

[СОДЕРЖАНИЕ 3](#_heading=h.2et92p0)

[ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 5](#_heading=h.tyjcwt)

[ВВЕДЕНИЕ 6](#_heading=h.3dy6vkm)

[1. Предпроектное обследование 7](#_heading=h.1t3h5sf)

[1.1. Объект автоматизации 7](#_heading=h.4d34og8)

[2. Концептуальное проектирование 10](#_heading=h.17dp8vu)

[2.1. Общая концепция системы 10](#_heading=h.3rdcrjn)

[2.2. Структурная схема системы 10](#_heading=h.26in1rg)

[2.3. Функциональная модель процесса автоматизации мониторинга износа металлоконструкций 11](#_heading=h.lnxbz9)

[2.4. Диаграмма целей 15](#_heading=h.35nkun2)

[3. Техническое задание 17](#_heading=h.1ksv4uv)

[3.1. Общие сведения 17](#_heading=h.44sinio)

[3.2. Требования к программе 17](#_heading=h.2jxsxqh)

[4. Техническое проектирование 19](#_heading=h.z337ya)

[4.1. Определение хронологии платформы автомобильных весов 19](#_heading=h.3j2qqm3)

[4.2. Задание набора управляющих параметров 19](#_heading=h.1y810tw)

[4.3. Расчетная схема металлоконструкции 20](#_heading=h.4i7ojhp)

[4.4. Модуль сбора данных 21](#_heading=h.2xcytpi)

[4.5. Модуль определения действительных напряжений и остаточного ресурса 22](#_heading=h.3whwml4)

[4.5.1. Алгоритм работы модуля определения действительных напряжений 2](#_heading=h.2bn6wsx)2

[4.5.2. Алгоритм работы модуля определения остаточного ресурса 24](#_heading=h.qsh70q)

[4.6. Модуль графического интерфейса 25](#_heading=h.3as4poj)

4.7 Объединение всех программных модулей 26

[5. Выбор системы инженерного расчета и языка программирования 2](#_heading=h.1pxezwc)7

[5.1. Выбор системы инженерного расчета и языка программирования 2](#_heading=h.49x2ik5)7

[5.2. Создание расчета конструкции 2](#_heading=h.2p2csry)7

[5.3. Расчет переходных матриц для по результатам анализа конструкции 30](#_heading=h.147n2zr)

[5.4. Создание программы для отображения нагрузок и износа 3](#_heading=h.3o7alnk)1

[ЗАКЛЮЧЕНИЕ](#_heading=h.4f1mdlm) 34

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_heading=h.2u6wntf) 35

[ПРИЛОЖЕНИЕ А](#_heading=h.19c6y18) 36

**ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

КЭ – конечный элемент.

КЭ-модель – конечно-элементная модель.

АСМ – Автоматизированная система мониторинга.

АЦП – Аналого-цифровой преобразователь.

ЯП– язык программирования

**ВВЕДЕНИЕ**

В сфере эксплуатации металлоконструкций немаловажным аспектом является мониторинг износа, конечной целью которого является более точное определение времени обследования на коррозионный и усталостный износ.

На данный момент времени нет оборудования, которое может определить действительный износ конструкций, но подобное оборудование поможет определять фактический уровень усталости материала. Использование такого рода оборудования поможет как регулировать необходимое число обследований металлоконструкций, так и упреждать их поломки.

Для создания такой системы мониторинга необходимо определение характерных точек, в которых мы сможем производить замеры силовых факторов.

**1. Предпроектное обследование**

**1.1. Объект автоматизации**

Так как для определения характерных точек и построения системы мониторинга необходима модель самой металлоконструкции. В данном курсовом проекте будет рассмотрена вышка сотовой связи (рисунок 1) с целью демонстрации потенциала данной системы.

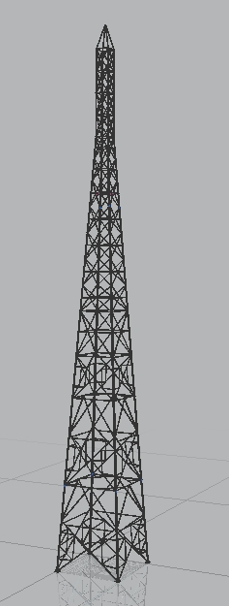


Рисунок 1 – Вышка сотовой связи

Антенно-мачтовое сооружение (АМС) — общее название для антенной опоры, входящей в состав оборудования площадки размещения радиоэлектронных средств (РЭС) связи.

По типу антенно-мачтовые сооружения различаются на свободностоящие — трубостойки, столбы, башни, и мачтового типа, с оттяжками.

Как правило, антенно-мачтовые сооружения имеют несущие металлические конструкции из стали и фундамент из армированного бетона (рисунок 2 и рисунок 3).



Рисунок 2 – Башня сотовой связи, смонтированная из металлических ферм



Рисунок 3 – Башня сети цифрового телерадиовещания ФГУП «РТРС» с характерной стойкой излучающих на передачу антенн на вершине сооружения.

**1.2. Теория прочности материалов**

Основной теорией, используемой в данной работе является теория прочности материалов при циклически меняющих напряжениях.

Циклическое напряжение (цикл напряжения) в теории прочности - это повторяющееся сложное напряженное состояние, которое происходит в технических конструкциях и материалах при их эксплуатации. Цикл напряжения характеризует то, как напряжение в материале изменяется со временем в процессе работы или воздействия какого-нибудь механического или физического процесса. Усталость материала при циклическом напряжении может привести к трещинам и разрушению. Поэтому важно анализировать циклические напряжения в конструкциях и разрабатывать материалы, которые устойчивы к такому воздействию.

Выделяют определенные виды циклов напряжения:

1. Симметричный цикл – максимальное и минимальное напряжения равны по абсолютной величине и противоположны по знаку

2. Асимметричный цикл – максимальное и минимальное напряжения не равны по абсолютной величине, при этом асимметричный цикл может быть знакопеременным или знакопостоянным;

а) Знакопеременный цикл – максимальное и минимальное напряжения не равны по абсолютной величине и противоположны по знаку

б) Знакопостоянный цикл – максимальное и минимальное напряжения не равны по абсолютной величине и имеют одинаковый знак

1. Отнулевой (пульсирующий) цикл – максимальное или минимальное напряжения равны нулю

Для определения остаточного ресурса в условиях реальной эксплуатации необходимо определить математический аппарат, необходимый для приведения асимметричных циклов напряжения к симметричным.

Испытания при симметричном цикле (при R=–1) оказываются наиболее простыми с точки зрения их реализации. Однако в реальных конструкциях подавляющее число деталей работает при асимметричном нагружении. Поэтому, чтобы обеспечить корректность расчета, необходимы сведения о пределах выносливости для любой асимметрии цикла.

В расчетной практике обычно пользуются двумя типами диаграмм: диаграммой предельных напряжений и диаграммой предельных амплитуд. В данной работе будет рассматриваться только диаграмма предельных напряжений.

Диаграмма предельных напряжений (также известная как диаграмма Смита) строится, как минимум, по трем режимам нагружения (по трем точкам), для каждого из которых определяют предел выносливости :

Первый режим (точка 1) – обычный симметричный цикл нагружения (R=–1, =0, =σ–1, =–σ–1);

Второй режим (точка 2) – асимметричный цикл, как правило, отнулевой (R=0, = σ0/2, =σ0, =0);

Третий режим (точка 3) – простое статическое растяжение (R=1, === ).

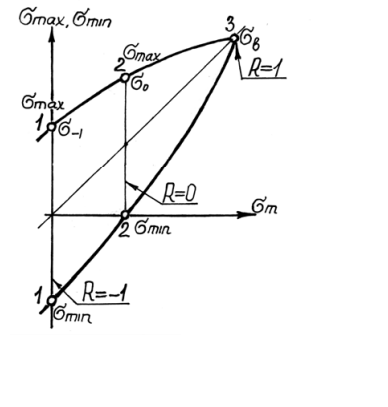
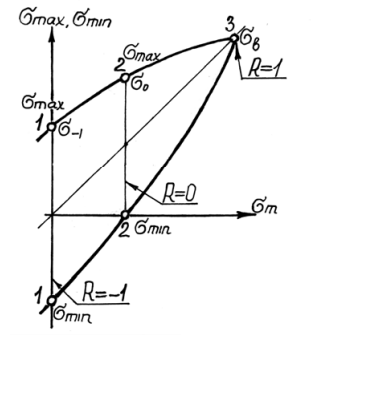


Рисунок 4 – диаграмма Смита.

Полученные точки соединяем плавной линией (1–2–3), ординаты точек которой соответствуют пределам выносливости материала при различных значениях коэффициента асимметрии цикла.

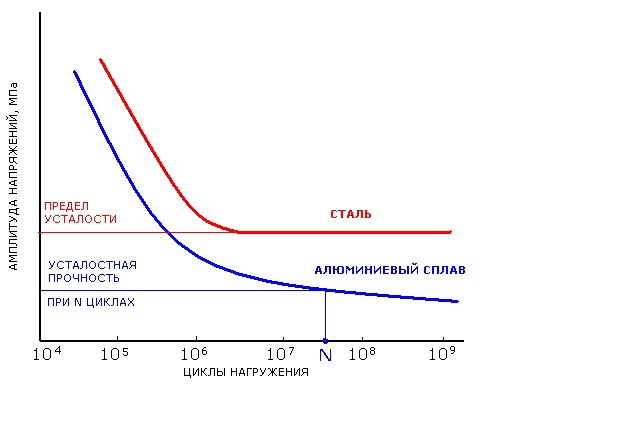
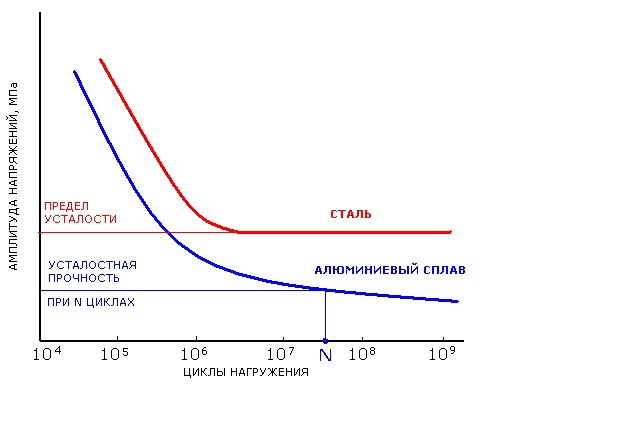
Легко показать, что луч, проходящий под углом β через начало координат диаграммы предельных напряжений, характеризует циклы с одинаковым коэффициентом асимметрии R:

Тогда, для определения предела выносливости при заданной асимметрии цикла R нужно по приведенной формуле вычислить величину угла β и провести луч под этим углом до пересечения с линией 1–2–3, ордината точки пересечения и даст нам искомый предел выносливости .

Опуская перпендикуляр на ось абсцисс, найдем среднее напряжение цикла , а на пересечении перпендикуляра со второй ветвью диаграммы – минимальное напряжение .

**1.2. Запас прочности материалов**

Число циклов, выдержанное деталью до разрушения, меняется от 5–10(например, многократный изгиб проволоки) до миллионов. Для математического представления используется кривая усталости.

Кривая усталости - это графическое представление зависимости действительного числа циклов прочности материала от амплитуды колебаний напряжений. 

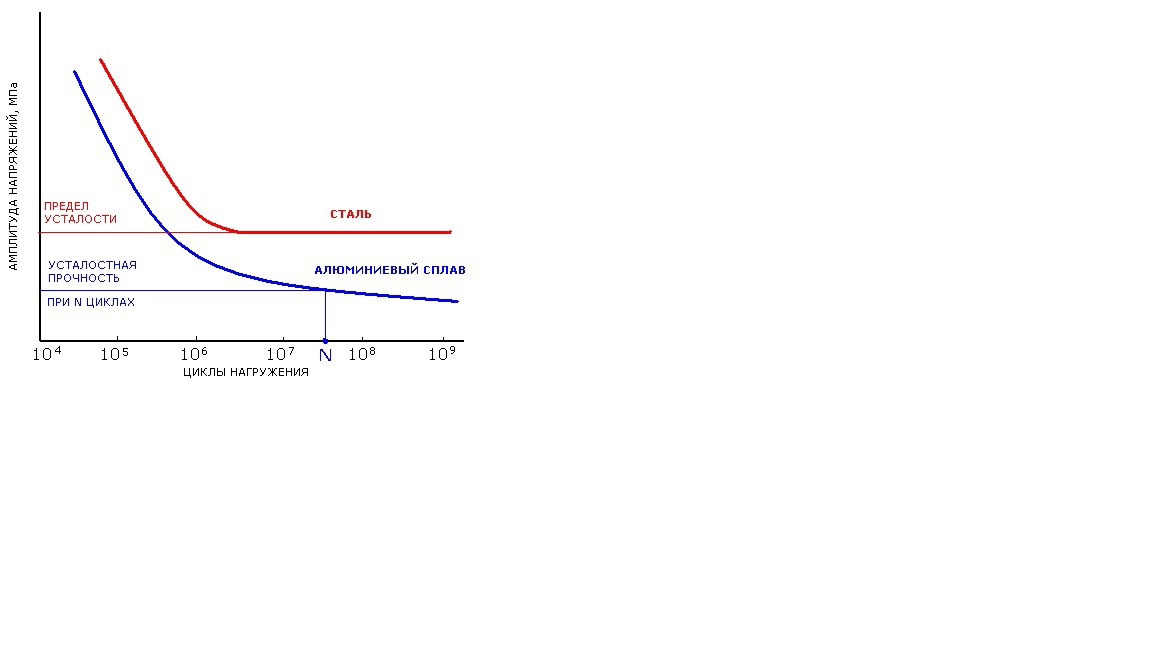


Рисунок 5 – Кривая усталости.

На начальном этапе прочность материала снижается медленно, затем скорость падает, пока прочность не достигает минимального значения, называемого пределом выносливости. После этого прочность продолжает падать, но уже медленнее, и наконец достигает окончательного значения, которое представляет собой остаточную прочность материала.

Кривая усталости может использоваться для прогнозирования срока службы изделий и для оптимизации конструкции и изготовления материалов, которые будут использоваться в условиях циклических нагрузок.

**2. Концептуальное проектирование**

**2.1. Общая концепция системы**

В общем виде система представляет из себя набор измерительных устройств и программных модулей, исполняющих определение силовых факторов, напряжений возникающих в конструкции. Предварительно, для системы необходимо определить характерные точки для крепления датчиков, в которых датчик сможет определять наиболее точно воздействия определённых силовых факторов. Система получает информацию о значении силовых факторов, действующих на конструкцию, которая будет использоваться как основа для расчёта напряжений в элементах объекта. На основе полученных результатов создаётся отчёт, в котором есть информация об оставшемся ресурсе металлоконструкции.

**2.2. Структурная схема системы**

В данном подразделе приведено описание структурной схемы автоматизированной система мониторинга воспринимаемых металлоконструкцией нагрузок (рисунок 6). Система состоит из модуля получения информации, модуля сбора данных и модуля определения остаточного ресурса и смещений элемента. Модуль определения остаточного ресурса и смещений элемента, в свою очередь, состоит из двух подмодулей: модуля определения остаточного ресурса и модуля определения смещений элемента.

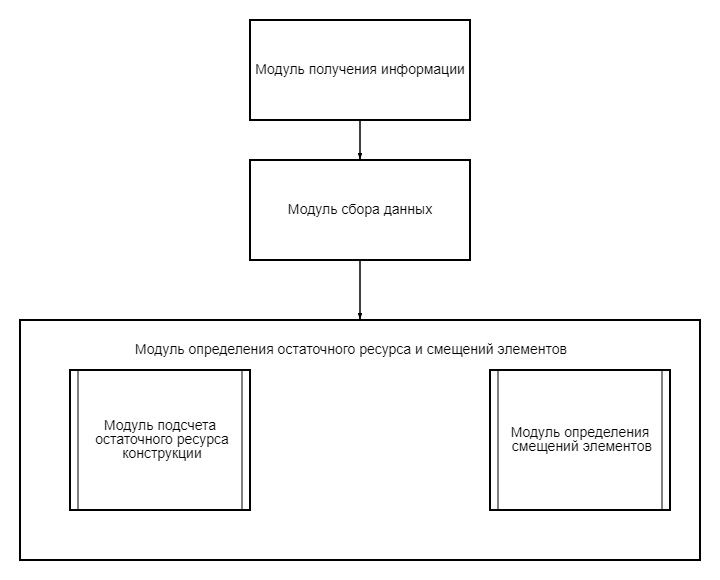


Рисунок 6 – Структурная схема системы

**2.3. Функциональная модель процесса автоматизации проектирования платформенных автомобильных весов**

Создание АСМ будет производиться в рамках мероприятий по автоматизации процесса мониторинга износа металлоконструкций. В данном подразделе производится функциональное моделирование процесса автоматизации с помощью нотации IDEF0. Для автоматизации процесса мониторинга износа металлоконструкций (рисунок 7) в качестве механизмов исполнения будут задействованы NX 12.0, Processing, MATLAB и специалист по жизненному циклу. На вход будет подаваться топология системы мониторинга состояния металлоконструкций. Все операции будут осуществляться с учетом критериальных ограничений, единой системы конструкторской и технологической документации. В результате проведенной работы должны получиться АСМ , конструкторская и технологическая документация.

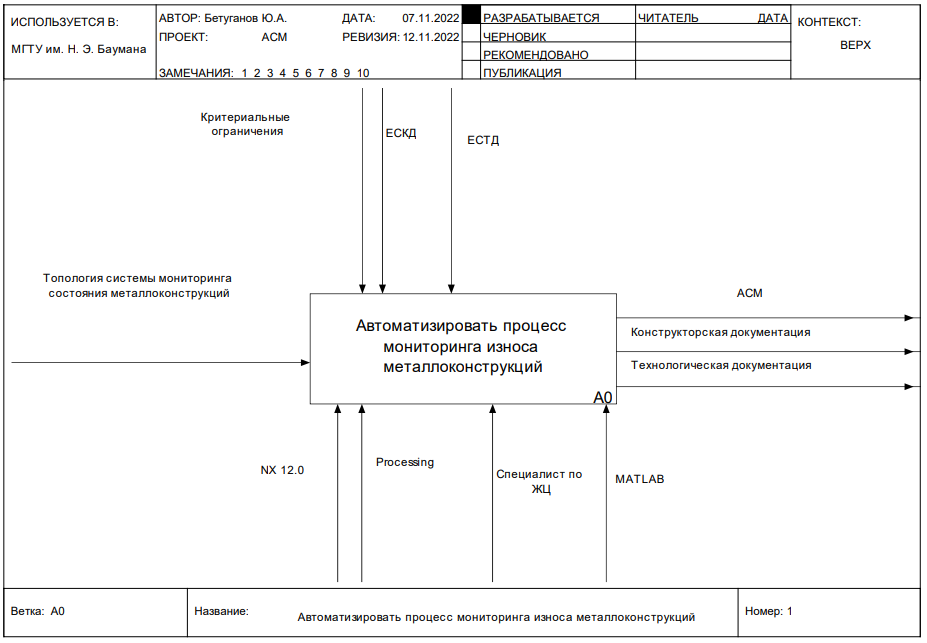


Рисунок 7 – Контекстная диаграмма IDEF0

Изначально определяется набор управляющих параметров и критериев качества, на основе которых будет производиться поиск оптимального решения. С учётом этих параметров и на базе заданной топологии будет разработана АСМ. Получившаяся система будет использована для нахождения действующих сил и напряжений в элементах конструкций, на основе которых будет создана конструкторская и технологическая документация для получившейся системы (рисунок 8).

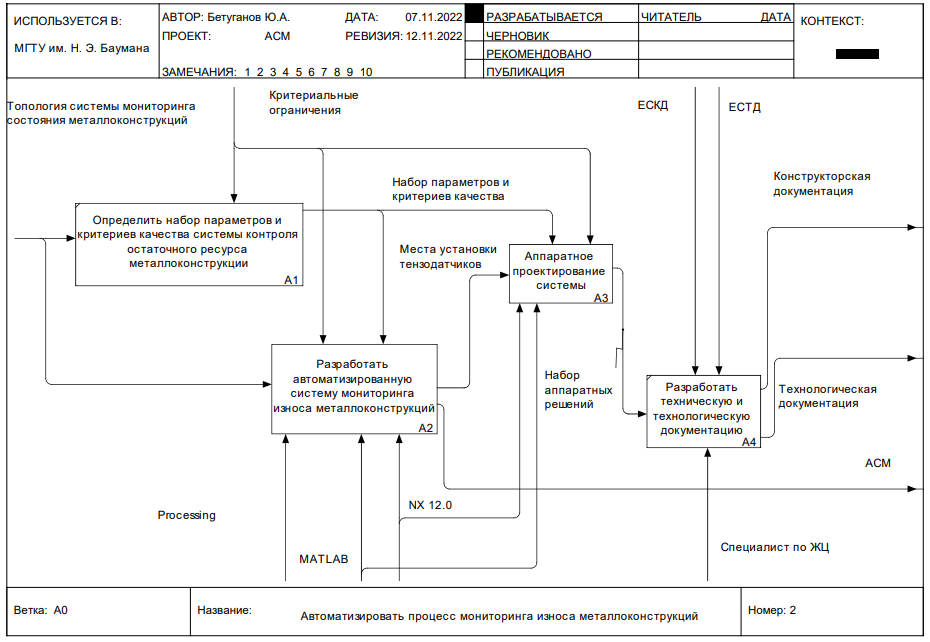


Рисунок 8 – Декомпозиция блока «Автоматизировать процесс проектирования платформенных автомобильных весов»

Для АСМ будут разработаны модуль сбора данных, модуль определения действительных напряжений, модуль расчета остаточного ресурса и модуль графического интерфейса, после чего все модули будут объединены в единую систему (рисунок 9). Разработка модуля критериев качества будет осуществляться на базе заданной топологии мониторинга с учетом значений управляющих параметров.

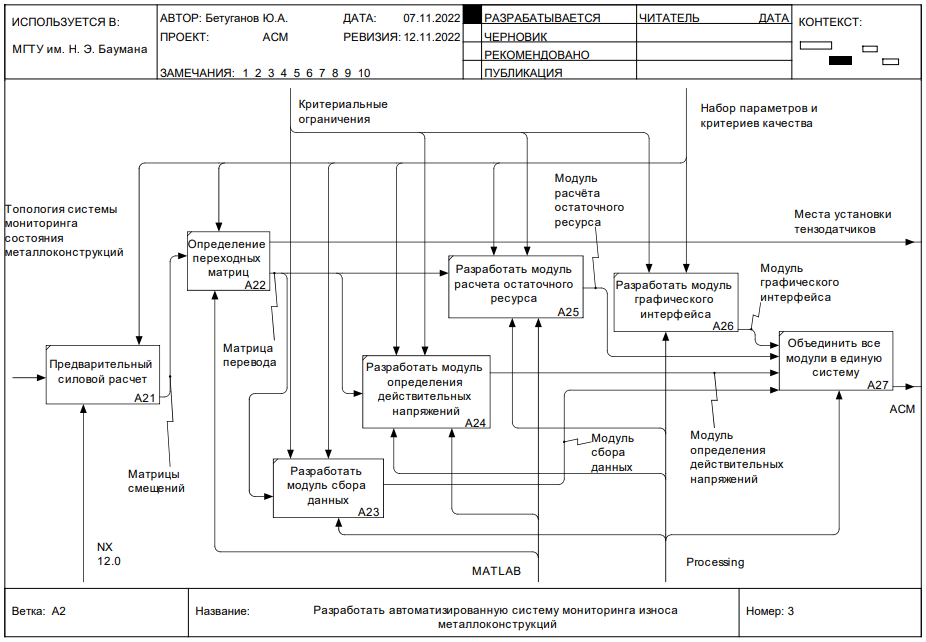


Рисунок 9 – Декомпозиция блока «Разработать автоматизированную систему поддержки принятия решений при управлении жизненным циклом платформенных автомобильных весов»

После создания АСМ, с помощью нее будет сгенерирован набор различных вариантов конструкции, для которых модуль расчета критериев качества рассчитает значения критериев оптимальности. После чего, на основе полученных решений, будет проведена проверка полученных конструкций. Будет сгенерирован больший набор вариантов конструкции, для которых будут определены значения критериев качества. Из полученного набора решений будут отфильтрованы те, что не подходят под критериальные ограничения, после чего будут найдены оптимальные решения. При необходимости требования к критериальным ограничениям будут ужесточаться до тех пор, пока не останется минимальный набор вариантов, из которых, на основе предпочтений пользователя, будет выбран один оптимальный (рисунок 8).

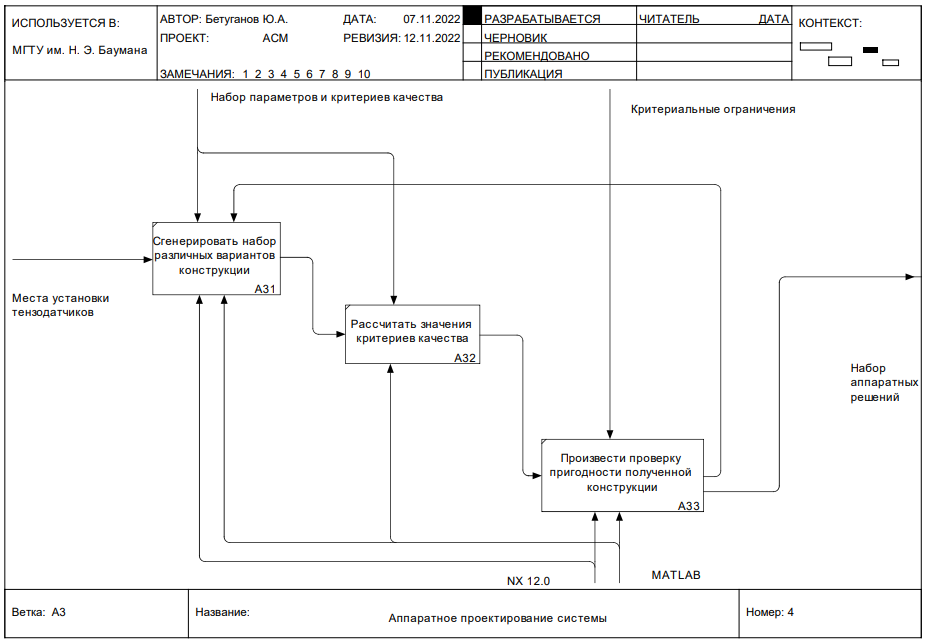


Рисунок 10 – Декомпозиция блока «Подобрать оптимальный вариант конструкции»

**2.4. Диаграмма целей**

Главной задачей АСМ, разрабатываемой в рамках данной работы, является автоматизация процесса мониторинга воспринимаемых металлоконструкцией нагрузок. Данную комплексную задачу легче решить, если разбить ее на некоторое конечное количество подзадач с помощью диаграммы целей нотации ARIS (рисунок 11). Главная задача «Разработать автоматизированную систему поддержки принятия решений при управлении жизненным циклом платформенных автомобильных весов» разбивается на три ползадачи: «Разработать программу для сбора данных о воспринимаемых нагрузках металлоконструкцией нагрузок и анализа полученных данных», «Обеспечить удобство процесса мониторинга состояния конструкций» и «Повысить эффективность процесса наблюдения за состоянием металлоконструкций».

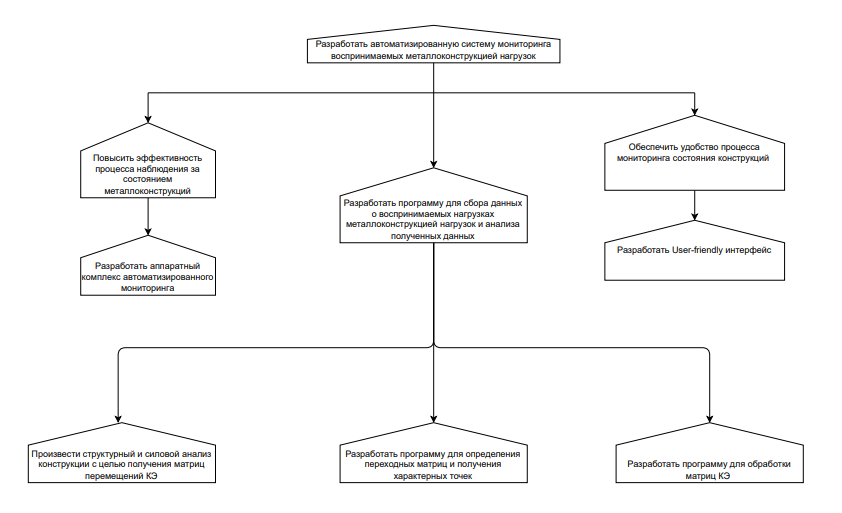


Рисунок 11 – Диаграмма целей

**3. Техническое задание**

Техническое задание на автоматизированную систему составлено в соответствии с ГОСТ 34.602-89 [11]. Оно состоит из 4 разделов и содержит в себе список параметров, по которым впоследствии будет проверяться разработанная система. Именно эти параметры дают возможность заказчику понять, соответствует ли система требованиям, которые были заложены на этапе формирования технического задания.

**3.1. Общие сведения**

**Основания для разработки** – задание на выполнение курсового проекта.

**Заказчик** – кафедра «Компьютерные системы автоматизации производства» МГТУ им. Н.Э. Баумана.

**Разработчик** **–** студент кафедры «Компьютерные системы автоматизации производства» Бетуганов Ю. А.

**Наименование темы разработки** **–** «Автоматизированная система мониторинга воспринимаемых металлоконструкцией нагрузок».

**3.2. Требования к программе**

Разрабатываемая система должна обеспечить мониторинг воспринимаемых конструкцией нагрузок и её остаточный ресурс, а также сохранение данных расчетов в отдельный текстовый файл для последующего считывания.

На этапе ***ввода исходных данных*** должен быть обеспечен следующий функционал:

1. CAD-модель металлоконструкции отправляются на предварительную обработку при проектировании системы.

На этапе ***проведения расчётов*** должен быть обеспечен следующий функционал:

1. Расчеты проводятся в фоновом режиме.
2. Все результаты должны быть записаны в базу данных для дальнейшей с ними работы.
3. Программа должна быть устойчива к появлению расчетов, вызывающих ошибки при их проведении.
4. Устойчивость к ошибкам расчётов, в случае критической ошибки – вывод пользователю и плановое завершение работы.

На этапе ***работы с результатами*** должен быть обеспечен следующий функционал:

1. Отображение результата в удобной форме: модель с цветовой дифференциацией.
2. Сохранение результата в базу данных.

Разрабатываемая система должна работать под управлением ОС Windows 10 и использовать современную систему инженерного расчета.

**4. Техническое проектирование**

**4.1. Определение хронологии системы мониторинга состояния**

Упрощенная схема мониторинга состояния конструкций представляет из себя отслеживание розы ветров и определение среднего износа исходя из данных. Поскольку отсутствуют точные данные по воздействию на конструкцию возможно производить лишь примерные предположения по необходимости проведения.

**4.2. Задание набора управляющих параметров**

Так как основной нагрузкой, воспринимаемой металлоконструкцией, является ветровая нагрузка, необходимо определять нагрузку на протяжении долгого промежутка времени и сохранять эту информацию. Основываясь на требованиях быстродействия и устойчивости устройства сбора данных, необходимо задать набор управляющих параметров таким образом, чтобы в процессе их оптимизации охватить как можно большее количество различных вариантов аппаратного и программного исполнения. В качестве управляющих параметров могут выступать любые величины, максимально полно характеризующие устройство, такие как: значения которых нельзя выразить через другие параметры, количество составляющих элементов, периодичность опроса, точность.

Опишем эти параметры ниже:

* Частота опроса ν измерительного преобразователя.
* Схема подключения тензорезистора.
* Чувствительность тензорезистора .
* Температурный коэффициент .
* Габаритные размеры балок крепления.
* Параметры сечения ребра жесткости B1 и H1, которые обозначают соответственно ширину и высоту прямоугольного сечения.
* Параметры сечения поперечной балки B2 и H2.

Помимо параметров измерительной системы, также важно указать параметры сетки, от которых зависит количество конечных элементов модели, напрямую влияющее на точность проводимого расчета:

* Минимальный линейный размер элемента сетки-4 мм
* Схема разбиения КЭ- QUAD4

**4.3. Расчетная схема металлоконструкции**

Для силового расчета конструкции и определения переходных матриц необходимо создавать КЭ-модель всей платформы и произвести силовой расчет при воздействии инерциальных сил, которые в данном приближении аналогичны ветровым нагрузкам

Создание модели будет производиться в декартовой системе координат. Для упрощения процесса будет произведен переход от трехмерной модели к упрощённой модели срединных плоскостей каждого из элементов.

Для закрепления конструкции на земле на основание будут наложены связи по трем степеням свободы (запрещены линейные перемещения вдоль всех осей).

К КЭ посередине платформы будет приложена распределенная нагрузка в размере 50 кг. Для приложения нагрузки к каждому конкретному КЭ общая нагрузка будет разделена количество элементов.

Расчетная схема платформы приведена на рисунке 12.

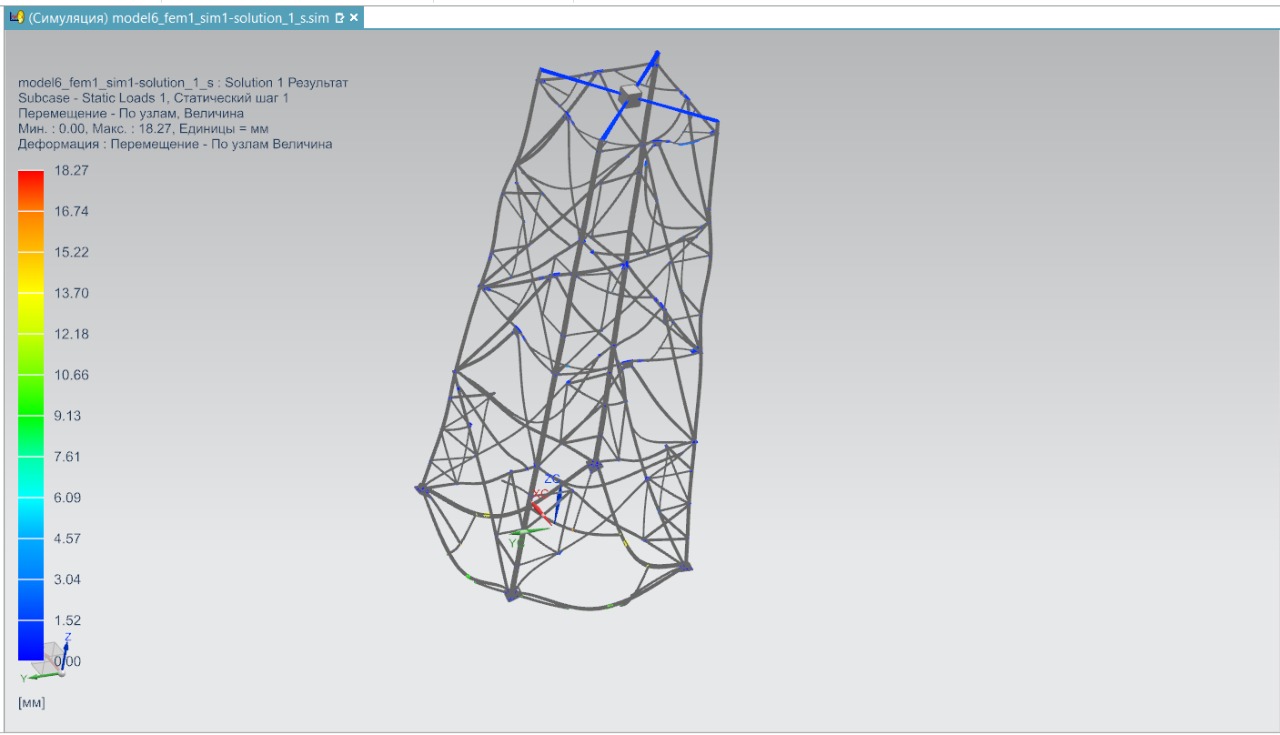


Рисунок 12 – Расчетная схема платформы

По результатам расчета должны быть получены следующие критерии оптимальности: **матрица координат КЭ**, **матрица перемещения узлов под действием нагрузок**.

**4.4. Модуль сбора данных**

Модуль сбора данных должен обеспечивать необходимые скорости записи данных, но учитывающие максимальный объем собранных данных. При условии записи состояний вышки на протяжении всего времени эксплуатации конструкции необходимо производить сбор данных не реже чем 1 раз в 1 секунду.

Для данного рода замеров необходимо использовать непосредственно тензорезистор с полной мостовой схемой подключения, а также измерительный преобразователь, с частотой АЦП более чем 1 Гц. Также АЦП должен иметь высокую разрядность, как , например, в АЦП с сигма-дельта-модуляцией. Для заданных требований подходит измерительный преобразователь КСК 5.1-2П.

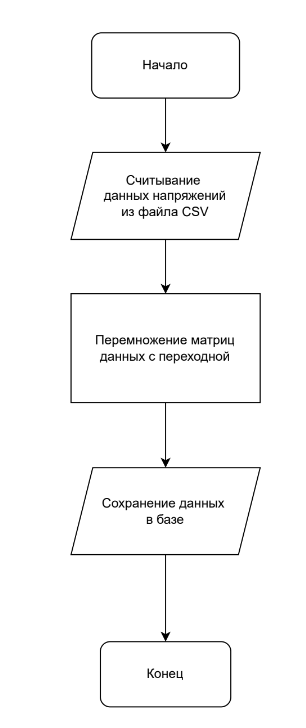
Так как падение напряжение из-за длины проводов значительно влияет на точность показания, из-за этого необходимо использовать измерительные преобразователи в непосредственной близости.

Исходя из заданных параметров, необходимо помещать преобразователи в непосредственной близости от датчиков. Преобразователи должны быть подключены к компьютерному устройству, которое будет отправлять данные на сервер.

**4.5. Модуль определения действительных напряжений и остаточного ресурса**

**4.5.1. Алгоритм работы модуля определения действительных напряжений**

Модуль определения действительных напряжений принимает на вход матрицу перехода для конкретной конструкции, данные о воспринимаемых нагрузках с тензоэлемента. На основе методов, используемых при определении критерия качества и в алгоритме выбора редуцированных моделей для мониторинга технических конструкций, формируем математический аппарат, который будет использован в программе. Блок-схема алгоритма работы модуля определения действительных напряжений на рисунке 13.



Рисунке 13 – Блок-схема алгоритма работы модуля определения действительных напряжений

**4.5.2. Алгоритм работы модуля определения остаточного ресурса**

Модуль определения остаточного ресурса принимает на вход данные о воспринимаемых нагрузках после обработки, данные об остаточном ресурсе. На основе теории прочности материалов при циклических напряжениях и понятий об усталости материалов, формируем математический аппарат, который будет использован в программе.Блок-схема алгоритма работы модуля определения действительных напряжений на рисунке 14.



Рисунок 14 – Блок-схема алгоритма работы модуля определения остаточного ресурса

**4.6 Модуль графического интерфейса**

Блок-схема алгоритма работы модуля графического интерфейса приведена на рисунке 15. В основе графического интерфейса лежит два вида отображения металлоконструкции: фактические деформации под действием нагрузок и отображение износа компонентов конструкции при помощи градиентного выделения.

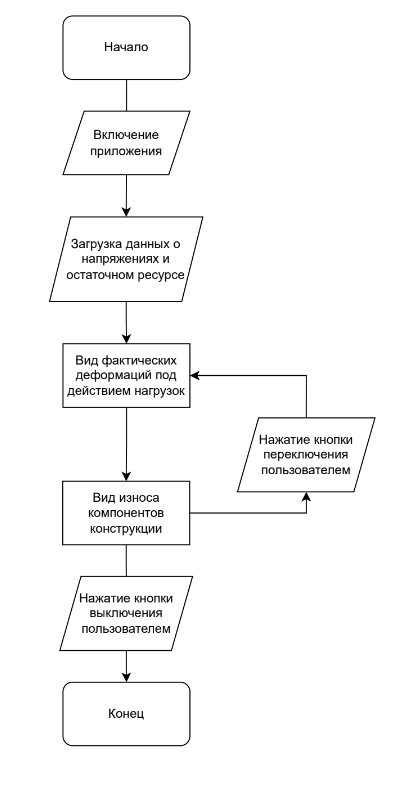
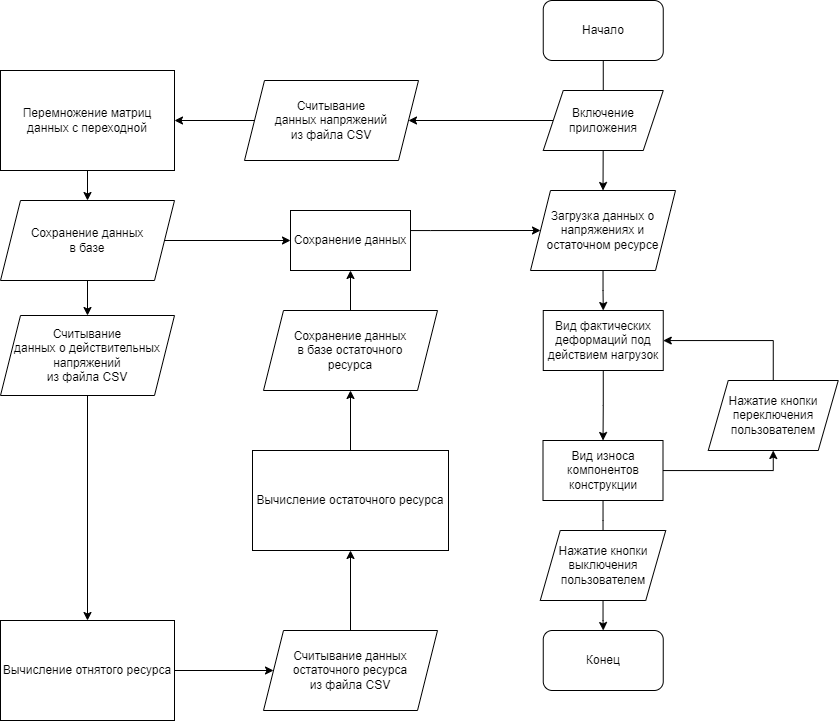


Рисунок 15 – Блок-схема работы главного модуля

**4.7 Объединение всех программных модулей**

Необходимо скомпоновать программу, соблюдая все логические и функциональные связи. Блок-схема компоновки всех программных модулей приведена на рисунке 16.

Рисунок 16 – Блок-схема компоновки всех программных модулей

**5. Рабочее проектирование**

**5.1. Выбор системы инженерного расчета и языка программирования**

В качестве CAE-системы для реализации модуля инженерного расчета была выбрана универсальная программная система конечно-элементного анализа NX2012. Встроенный в систему набор команд, необходимых для автоматического построения КЭ-модели, ее нагружения и сохранения результатов расчета в отдельный текстовый файл. Разработчиками системы была предусмотрена возможность выполнения всех вычислений в фоновом режиме. Данная программа недоступна для скачивания на официальном сайте компании в связи с геополитической обстановкой.

Разработка других модулей системы будет осуществляться с помощью языка программирования высокого уровня Processing, на базе которого реализовано большое количество пользовательских библиотек и расширений. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нем программ по сравнению с аналогичным кодом, написанным на компилируемых языках более низкого уровня. Сам язык, библиотеки и расширения на его основе распространяются по свободной лицензии и лежат в открытом доступе.

**5.2. Планирование эксперимента**

Формирование матриц и массивов и операции над ними будут осуществляться с помощью Matlab, предназначенной для работы с продвинутыми математическими алгоритмами. Генерация матрицы перехода будет произведена при помощи функций SVD сингулярного разложения.

Полученная матрица матрица V будет использована для определения хорошего решения – точек конструкции, к которым нужно крепить тензоэлементы для получения наиболее близких результатов составляющих действующих напряжений. Основным показателем малый градиент напряжений в пределах элемента.

clear emx emy A A1 M I

A(1,:)=x3(:,6);

A(2,:)=y1(:,6);

%import

%%

clear B I I2 I1 M1 M2 Ie

[U,S,V] = svd(A,"econ"); %сингулярное разложение

clear B I vm

[B,I] = sort(-abs(V(:,1)));%сортируем в поисках хорошего решения точки №1

n=100;

for i = 1:n

[M1(1),I1(1)] =min(abs(x3((I(i)+1):end,2)-x3(I(i),2)));

[M1(2),I1(2)] =min(abs(x3(1:I(i)-1,2)-x3(I(i),2)));

[M2,I2] =min(M1);

vm1(i)=abs(A(1,I1(I2))-A(1,I(i)))/abs(A(1,I1(I2))+A(1,i))+abs(A(2,I1(I2))-A(2,I(i)))/abs(A(2,I1(I2))+A(2,I(i)));

end

vm1 % ищем такие элементы где малый градиент напряжений (деформаций) в пределах элемента

%[B,Ie1] = sort(-abs(vm.\*A(1,I(1:n))));

[B,Ie1] = sort(abs(vm1));

%vm1(Ie1(1))

clear B I vm

[B,I] = sort(-abs(V(:,2)));%сортируем в поисках хорошего решения точки №2

n=100;

for i = 1:n

[M1(1),I1(1)] =min(abs(x3((I(i)+1):end,2)-x3(I(i),2)));

[M1(2),I1(2)] =min(abs(x3(1:I(i)-1,2)-x3(I(i),2)));

[M2,I2] =min(M1);

vm2(i)=abs(A(1,I1(I2))-A(1,I(i)))/abs(A(1,I1(I2))+A(1,i))+abs(A(2,I1(I2))-A(2,I(i)))/abs(A(2,I1(I2))+A(2,I(i)));

end

%[B,Ie2] = sort(-abs(vm.\*A(2,I(1:n))));

[B,Ie2] = sort(abs(vm2));

x3(I(Ie1(1)),2) %хорошая точка №1 (номер узла)

x3(I(Ie2(1)),2) %хорошая точка №2 (номер узла)

%% подготовка к визуализации

clear emx emy A A1 ed

ee1=0;k=0;k1=0;k2=0;

for i = 1:numel(xe(:,6))

ee=xe(i,1);

k=k+1;

if ee==ee1

else

k=1;k1=k1+1;

end

if ((k1>0)&&(k>0))

[M,I] = min(abs(xe(i,2)-xd(:,1)));

ed(k1,1)=xe(i,1);

ed(k1,k+1)=xe(i,3);% элементы

ed(k1,k+5)=xe(i,4);

ed(k1,k+9)=xe(i,5);

ed(k1,k+12)=xd(I,5);% перемещения

ed(k1,k+16)=xd(I,6);

ed(k1,k+20)=xd(I,7);

end

ee1=ee;

end

hold on

plot3(ed(:,2),ed(:,6),ed(:,10),'r.');

plot3(ed(:,3),ed(:,7),ed(:,11),'g.');

plot3(ed(:,4),ed(:,8),ed(:,12),'b.');

plot3(ed(:,5),ed(:,9),ed(:,13),'m.');

hold off

%x3(I,2)

Листинг 1 – программа MATLAB для определения характерных точек и составления матрицы перехода

**5.3. Создание расчета конструкции**

В общем виде программный модуль для исполнения инженерного расчета состоит из упрощенной модели металлоконструкции, определения связей и нагрузок, силового расчета при заданных параметрах.

Упрощенная модель создается путем создания модели серединных плоскостей элементов. Для создания такой модели необходимо использользовать функцию “Срединная поверхность по параметрам”. После создания этих элементов необходимо скрыть твердотельные элементы, провести создание “Универсальных соединений” между полученными плоскостями. Для проведения расчета необходимо задать точки фиксации в основании конструкции, приложить равномерную нагрузку по двум перпендикулярным направлениям.

Полученные результаты представляют из себя множество координат точек и их смещений представленных в файле формата CSV.

**5.4. Создание программы для отображения нагрузок и износа**

За демонстрацию точек и перемещений отвечает программа, описанная в листингах 2 и 3. Основной принцип работы данной программы заключается в циклическом отображении каждой точки. Для отображения также используется аргумент функции P3D для симуляции трехмерного пространства, а также смещение положения позиции и наклона за счет функции rotateX(),rotateZ() и translate().

Функция setup() - это обязательная функция, которая вызывается один раз при запуске программы в среде Processing. Она используется для инициализации настроек программы, таких как размер окна, фоновый цвет, загрузка изображений.

import processing.core.\*;

import processing.data.\*;

import processing.event.\*;

import processing.opengl.\*;

import processing.\*;

import java.\*;

import g4p\_controls.GWindow;

import g4p\_controls.\*;

GButton btnMOVE;

GButton btnTENS;

GButton btnHP;

GButton btnCOEF1;

GCustomSlider sldrX;

GCustomSlider sldrY;

GTextBase nmb\_X;

GTextBase nmb\_Y;

float X\_float, Y\_float;

float wheel;

PVector Pos\_of\_main;

Table table, tableE;

int n=3000000;

float [][][] xx = new float[n][5][4];

float [][][] xd = new float[n][5][4];

float [][][] xt = new float[n][5][4];

float [][][] hp = new float[n][5][4];

float max\_hp=3300000000.0;

float []tension\_x = new float[n];

float []tension\_y = new float[n];

//float [][][] xd = new float[n][5][4];

int ee=0;

int ee1;

int k, k1, k2, k3;

int x, y;

int n2=100;

int nF=100;

int [][] F = new int[nF][nF];

float [][] xg = new float[n2+10][6];

float e, coef;

float [][][] constr=new float[n][5][4];

float displayWidth\_rect\_1;

float displayWidth\_rect\_2;

float displayHeight\_rect\_1;

float displayHeight\_rect\_2;

float xc, yc, zc, dx, dy, dz, r1, r2, r3, xb1, xb2, xb3, yb1, yb2, yb3, rb, s;

PImage lin, cur, cir, back, b1, b2;

float Sc=0.8;

float max1=0;

float max2=0;

float min1=0;

float min2=0;

float Scd=200;

boolean state\_C=false;

boolean state\_b1=false;

boolean state\_b2=false;

boolean state\_b3=false;

int imax1=0;

int imax2=0;

int Fpr1, Fpr2;

public void setup() {

surface.setResizable(true);

surface.setLocation(100, 100);

coef=1;

nmb\_X=new GTextField (this, 40, 800, 200, 60);

nmb\_Y=new GTextField (this, 40, 1000, 200, 60);

sldrX = new GCustomSlider(this, 40, 300, 200, 60, null);

sldrY = new GCustomSlider(this, 40, 500, 200, 60, null);

btnMOVE = new GButton(this, 510, 20, 140, 40, "Режим отображения перемещений");

btnTENS = new GButton(this, 810, 20, 140, 40, "Режим отображения напряжений");

btnHP = new GButton(this, 1110, 20, 140, 40, "Режим отображения запаса хп");

btnCOEF1 = new GButton(this, 1410, 20, 140, 40, "Вернуть исходный размер конструкции");

lin = loadImage("lin2.png");

cur = loadImage("cur.png");

cir = loadImage("s2.png");

b1 = loadImage("b1.png");

b2 = loadImage("b2.png");

back = loadImage("back2.png");

back.resize(int(displayWidth\_rect\_2), int(displayHeight\_rect\_1));

xb1=displayWidth\*0.7;

yb1=displayHeight\*0.15;

xb2=displayWidth\*0.8;

yb2=displayHeight\*0.15;

xb3=displayWidth\*0.9;

yb3=displayHeight\*0.15;

rb=displayHeight\*0.15;

}

В данной программе эта функция задаёт параметры отображения окна:

1. Размер окна - ширина и высота.

2. Положение окна на экране - координаты X и Y.

3. Режим окна - обычный или развернутый на весь экран.

4. Область отображения - размер и положение окна на экране.

5. Фоновый цвет окна.

6. Доступность кнопок закрытия, свертывания и разворачивания окна.

7. Возможность изменения размера окна.

8. Возможность перемещения окна по экрану.

Помимо этого, в данной функции создаются элементы управления, такие как кнопки и слайдеры, а также назначаются зависимости кнопок и функций.

Функция settings() - это встроенная функция в языке программирования Processing, которая используется для определения настроек для обработки рендеринга графики. Она представляет собой первую функцию, которая вызывается при запуске программы.

void settings() {

size(displayWidth, displayHeight, P3D);

table = loadTable("data.csv", "header");

tableE = loadTable("tensionE.csv", "header");

k1=0;

k=0;

xc=0;

yc=0;

zc=0;

displayWidth\_rect\_1=displayWidth\*4/16;

displayWidth\_rect\_2=displayWidth;

displayHeight\_rect\_1=displayHeight;

displayHeight\_rect\_2=displayHeight\*2/16;

println(displayWidth, displayHeight);

for (TableRow row : table.rows()) { // загрузка координат в массив

ee = row.getInt(0);

k1=k1+1;

xx[k1][0][0] = row.getFloat(1)\*0.03f;// координата X

xx[k1][1][0] = row.getFloat(5)\*0.03f;// координата Y

xx[k1][2][0] = row.getFloat(9)\*0.03f;// координата Z

xx[k1][0][1] = row.getFloat(2)\*0.03f;// компоненты вектора перемещений при действии нагружения 1 X

xx[k1][1][1] = row.getFloat(6)\*0.03f;// компоненты вектора перемещений при действии нагружения 1 Y

xx[k1][2][1] = row.getFloat(10)\*0.03f;// компоненты вектора перемещений при действии нагружения 1 Z

xx[k1][0][2] = row.getFloat(3)\*0.03f;// компоненты вектора перемещений при действии нагружения 2 X

xx[k1][1][2] = row.getFloat(7)\*0.03f;// компоненты вектора перемещений при действии нагружения 2 Y

xx[k1][2][2] = row.getFloat(11)\*0.03f;// компоненты вектора перемещений при действии нагружения 2 Z

xx[k1][0][3] = row.getFloat(4)\*0.03f;//

xx[k1][1][3] = row.getFloat(8)\*0.03f;//

xx[k1][2][3] = row.getFloat(12)\*0.03f;//

xd[k1][0][0] = row.getFloat(13)\*0.002f;//

xd[k1][0][1] = row.getFloat(17)\*0.002f;//

xd[k1][0][2] = row.getFloat(21)\*0.002f;//

}

k3=0;

for (TableRow rowE : tableE.rows()) { // загрузка координат в массив

ee = rowE.getInt(0);

xt[k3][0][0] =rowE.getFloat(11);// Von-mises X

xt[k3][1][0] = xt[k3][0][0];// Von-mises Y

k3=k3+1;

}

}

Функция settings() не обязательна для использования, но позволяет изменять или установить следующие основные параметры:

1. Размер окна приложения - функция size() задает ширину и высоту окна приложения. Можно использовать только один вызов этой функции в функции settings().

2. Режим рендеринга - Processing поддерживает два основных режима рендеринга графики: P2D (2D) и P3D (3D). Функция size() используется для установки режима рендеринга в P2D или P3D.

3. Настройки количество кадров в секунду - частота кадров (FPS) отвечает за количество кадров, которые отображаются на экране за одну секунду. Функция frameRate() устанавливает количество FPS. В данном частном случае эта функция не использовалась с целью улучшения опыта использования програмного обеспечения.

4. Режим отображения на экране - Иногда приложение должно быть запущено в полноэкранном режиме или иметь определенный размер окна. Функция fullScreen() дает возможность запускать приложение в полноэкранном режиме, а функция size() используется для определения размера окна, когда приложение не находится в полноэкранном режиме.

После определения необходимых параметров, функция используется для определения массива конечных элементов и их напряжений.

public void draw() {

background(100, 100, 100);

rect(0, 0, displayWidth\_rect\_1, displayHeight\_rect\_1);

fill(255);

rect(0, 0, displayWidth\_rect\_2, displayHeight\_rect\_2);

fill(255);

textSize(20);

fill(0, 0, 0);

text("Режим отображения перемещений", 320, displayHeight\_rect\_2/2+20);

text("Режим отображения напряжений", 700, displayHeight\_rect\_2/2+20);

text("Режим отображения запаса хп", 1100, displayHeight\_rect\_2/2+20);

text("Вернуть исходный размер конструкции", 1450, displayHeight\_rect\_2/2+20);

text("Состовляющая по оси Х="+X\_float, 10, height\*3/8-20);

text("Состовляющая по оси Y="+Y\_float, 10, height\*5/8-20);

translate(width\*5/8, height\*0.85);

rotateX(PI/2);

rotateZ(PI/4);

if (wheel==1) {

coef=coef-0.03;

if (coef<0)

{

coef=0;

}

if (coef>1.1)

{

coef=1.15;

}

}

if (wheel==-1) {

coef=coef+0.03;

if (coef<0)

{

coef=0;

}

if (coef>1.1)

{

coef=1.1;

}

}

fill(255);

stroke(255);

thread("requestData1");

thread("requestData2");

thread("requestData3");

thread("requestData4");

thread("requestData5");

thread("requestData6");

thread("requestData7");

if (state\_b1) {//режим отображения перемещений

for (int i = 0; i < k1; i++) {

stroke(int(tension\_x[i])\*50, 0, int(tension\_y[i])\*50, 100);

point(constr[i][0][0], constr[i][0][1], constr[i][0][2]);

}

}

if (state\_b2) {//режим отображения напряжений

for (int i = 0; i < k1; i++) {

stroke(s, xt[i][0][0]\*40\*abs(mouseX-displayWidth\*5/8)/displayWidth, xt[i][0][0]\*40\*abs(mouseY-displayHeight\*1/2)/displayHeight);

point(xx[i][0][0]\*coef, xx[i][1][0]\*coef, xx[i][2][0]\*coef);

}

}

if (state\_b3) {//режим отображения запаса хп

for (int i = 0; i < k1; i++) {

stroke(255\*( (hp[i][0][0]+hp[i][1][0])/max\_hp), 0, 0);

point(xx[i][0][0]\*coef, xx[i][1][0]\*coef, xx[i][2][0]\*coef);

}

}

wheel=0;

}

public void mouseWheel(MouseEvent event) {

float e = event.getCount();

println(e);

wheel=e;

}

Функция draw() является одной из основных функций в библиотеке Processing. Важно отметить, что функция draw() является бесконечным циклом и будет выполняться, пока не будет закрыто окно. Внутри функции draw() можно обновлять графические элементы в зависимости от входных данных, например, изменять цвет, положение или размер объектов в зависимости от данных, полученных от пользователя или других источников. Функция draw() также позволяет работать с анимацией и создавать движущиеся графические объекты. Для этого можно использовать функции, такие как translate() и rotate(), чтобы изменять положение и ориентацию объектов на экране. В целом, функция draw() является главной функцией для создания графических приложений в библиотеке Processing и позволяет создавать интерактивные и живые графические элементы, которые реагируют на взаимодействие пользователя.

В нашем случае, отображается конструкция в 3 различных вариантах:

1. Отображение перемещений нагруженной башни с учетом X и Y составляющих с цветовой индикацией,
2. Отображение напряжений нагруженной башни с учетом X и Y составляющих с цветовой индикацией,
3. Отображение остаточного ресурса нагруженной башни с учетом X и Y составляющих с цветовой индикацией.

Важной особенностью данной программы в том, чтобы совершать все расчёты не в основном потоке и тем самым увеличить скорость отображения точек башни. Для этого используется функция thread(), которая запускает параллельно функции draw() потоки , которые описаны ниже.

Помимо запуска процессов расчёта в главном потоке происходит масштабирование всей конструкции с помощью вращения колеса мыши.

void requestData1() {

if (state\_b1) {

for (int i = 0; i < n/4; i++) {

// if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

constr[i][0][0]=(xx[i][0][0]+xd[i][0][0]\*(X\_float-0.5)\*500)\*coef;

constr[i][0][1]=(xx[i][1][0]+xd[i][0][1]\*(Y\_float-0.5)\*500)\*coef;

constr[i][0][2]=(xx[i][2][0]+xd[i][0][2]\*(Y\_float-0.5)\*500)\*coef;

// }

}

}

}

void requestData2() {

if (state\_b1) {

for (int i = n/4; i < n/2; i++) {

// if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

constr[i][0][0]=(xx[i][0][0]+xd[i][0][0]\*(X\_float-0.5)\*500)\*coef;

constr[i][0][1]=(xx[i][1][0]+xd[i][0][1]\*(Y\_float-0.5)\*500)\*coef;

constr[i][0][2]=(xx[i][2][0]+xd[i][0][2]\*(Y\_float-0.5)\*500)\*coef;

// }

}

}

}

void requestData3() {

if (state\_b1) {

for (int i = n/2; i < n\*3/4; i++) {

// if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

constr[i][0][0]=(xx[i][0][0]+xd[i][0][0]\*(X\_float-0.5)\*500)\*coef;

constr[i][0][1]=(xx[i][1][0]+xd[i][0][1]\*(Y\_float-0.5)\*500)\*coef;

constr[i][0][2]=(xx[i][2][0]+xd[i][0][2]\*(Y\_float-0.5)\*500)\*coef;

// }

}

}

}

void requestData4() {

if (state\_b1) {

for (int i = n\*3/4; i < n; i++) {

// if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

constr[i][0][0]=(xx[i][0][0]+xd[i][0][0]\*(X\_float-0.5)\*500)\*coef;

constr[i][0][1]=(xx[i][1][0]+xd[i][0][1]\*(Y\_float-0.5)\*500)\*coef;

constr[i][0][2]=(xx[i][2][0]+xd[i][0][2]\*(Y\_float-0.5)\*500)\*coef;

//}

}

}

}

void requestData5() {

if (state\_b1) {

for (int i = 0; i < n; i++) {

//if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

tension\_x[i]=xt[i][0][0]\*abs(X\_float-0.5)\*2;

tension\_y[i]=xt[i][1][0]\*abs(Y\_float-0.5)\*2;

//}

}

}

}

void requestData6() {

if (state\_b2) {

for (int i = 0; i < n; i++) {

//if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

s=xt[i][0][0]\*X\_float+xt[i][1][0]\*Y\_float;

// }

}

}

}

void requestData7() {

if (state\_b3) {

for (int i = 0; i < n; i++) {

//if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

hp[i][0][0]=hp[i][0][0]+xt[i][0][0]\*X\_float;

hp[i][1][0]=hp[i][1][0]+xt[i][1][0]\*Y\_float;

// }

}

}

}

public void handleButtonEvents(GButton button, GEvent event) {

if (button == btnMOVE && event == GEvent.CLICKED) {

state\_b1=true;

state\_b2=false;

state\_b3=false;

//println("Button 'btnMOVE' was clicked");

}

if (button == btnTENS && event == GEvent.CLICKED) {

state\_b1=false;

state\_b2=true;

state\_b3=false;

//println("Button 'btnTENS' was clicked");

}

if (button == btnHP && event == GEvent.CLICKED) {

state\_b1=false;

state\_b2=false;

state\_b3=true;

//println("Button 'btnHP' was clicked");

}

if (button == btnCOEF1 && event == GEvent.CLICKED) {

coef=1.0;

//println("Button 'btnHP' was clicked");

}

}

public void handleSliderEvents(GValueControl slider, GEvent event) {

// println("Slider Value " + sldrX.getValueI());

println( sldrX.getValueF());

X\_float = sldrX.getValueF();

// println("Slider Value " + sldrY.getValueI());

println( sldrY.getValueF());

Y\_float = sldrY.getValueF();

}

public void handleTextEvents(GEditableTextControl textcontrol, GEvent event)

{

println("Button 'btnHP' was clicked");

}

Данный участок кода предназначен для запуска пересчета напряжений и положения конструкции. Он был специально вынесен наружу и для отдельного запуска процесса.

**6. Апробирование**

Апробирование системы автоматизированного мониторинга износа конструкции - это процесс проверки функциональности системы, которая позволяет автоматически следить за состоянием конструкции и выявлять признаки ее износа.

Описывая этот процесс, следует учитывать, что программное обеспечение состоит из 3 основных блоков:

1. Блок считывания данных. На этом этапе система настраивается согласно требованиям заказчика. Настраиваются параметры мониторинга, устанавливаются датчики измерения и т.д.

2. Блок расчета параметров. Этот этап позволяет проверить работоспособность системы без воздействия на конструкцию.

3. Тестирование интерфейса системы. роверка функциональности, удобства использования и эргономики пользовательского интерфейса программного обеспечения.

В процессе апробирования необходимо проверить работу каждого блока.

**6.1. Тестирование блока считывания данных**

Цель тестирования: проверить, что блок записи данных работает корректно и сохраняет переданные данные в базе данных.

Шаги тестирования:

1. Создание тестовой базы данных.

2. Написание автоматических тестов для блока записи данных.

3. Запуск тестов и проверка результатов.

Ожидаемые результаты тестирования:

1. Запись данных должна быть произведена без ошибок.

2. Данные должны быть корректно сохранены в базу данных.

3. При повторной записи данных должно происходить обновление уже сохраненных данных.

Результаты тестирования: блок записи данных успешно прошел автоматическое тестирование. Все ожидаемые результаты были достигнуты. Всего было проведено 10 тестов, из которых 10 были успешно выполнены.

Выводы: блок записи данных работает корректно и сохраняет переданные данные в базе данных. Тестирование показало, что блок функционирует без ошибок и готов для использования в дальнейшем .

**6.2. Тестирование блока расчета параметров**

Для этого необходимо выполнить следующие шаги:

1. Протестировать программу на известных данных.

Описание теста: подобрать несколько специальные данные, под которые проектировалась программа. В данном случае выборкой является массив значений параметров конечных элементов.

Результат: расчётный модуль произвёл корректные результаты при многократном тестировании, вследствие чего можно сделать вывод о его пригодности использования в данном проекте.

1. Проверить правильность ввода данных.

Описание теста: необходимо проверить , что все введенные данные правильные и во всех полях ввода присутствуют значения.

Результат: введенные массивы информации не претерпевают корректировок при вводе и многократной проверке.

3. Обратиться к специалистам. Обратитесь за помощью к специалистам в области расчета силовых параметров нагрузки, чтобы они проверили работу программы и высказали свое мнение о ее корректности и точности.

Результаты тестирования: блок записи данных успешно прошел автоматическое тестирование. Все ожидаемые результаты были достигнуты.

Выводы: блок расчета параметров работает корректно и сохраняет переданные данные в базе данных. Тестирование показало, что блок функционирует без ошибок и готов для использования в дальнейшем.

**6.3. Тестирование интерфейса системы**

Цель тестирования: проверить работоспособность интерфейса системы и убедиться, что пользователь может использовать все функции и возможности системы без проблем.

Метод тестирования: ручное тестирование

1. Изменение значения составляющих силы ветровой нагрузки

Описание теста: проверка работоспособности слайдеров для управления значения X и Y составляющих ветровой нагрузки

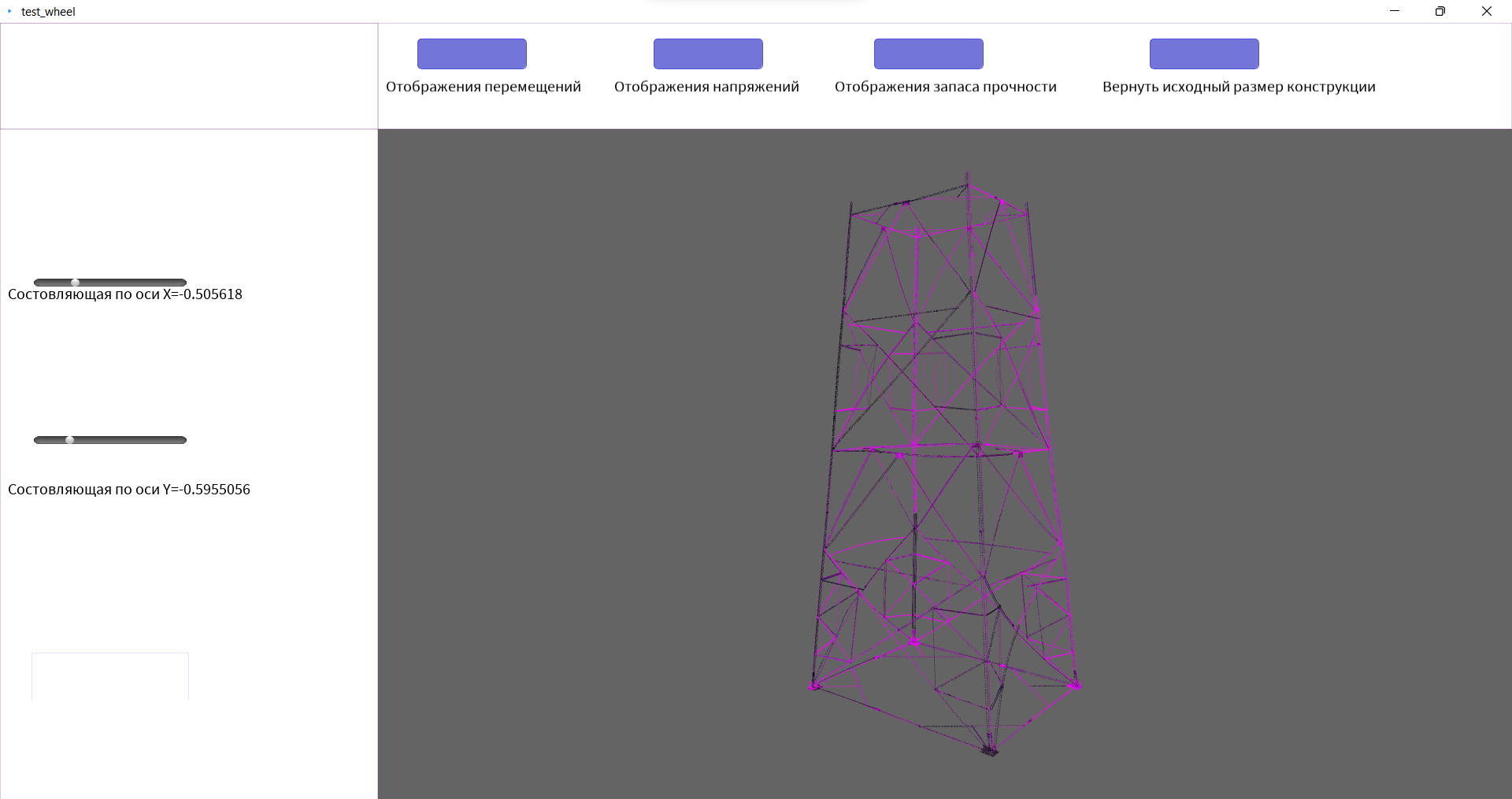
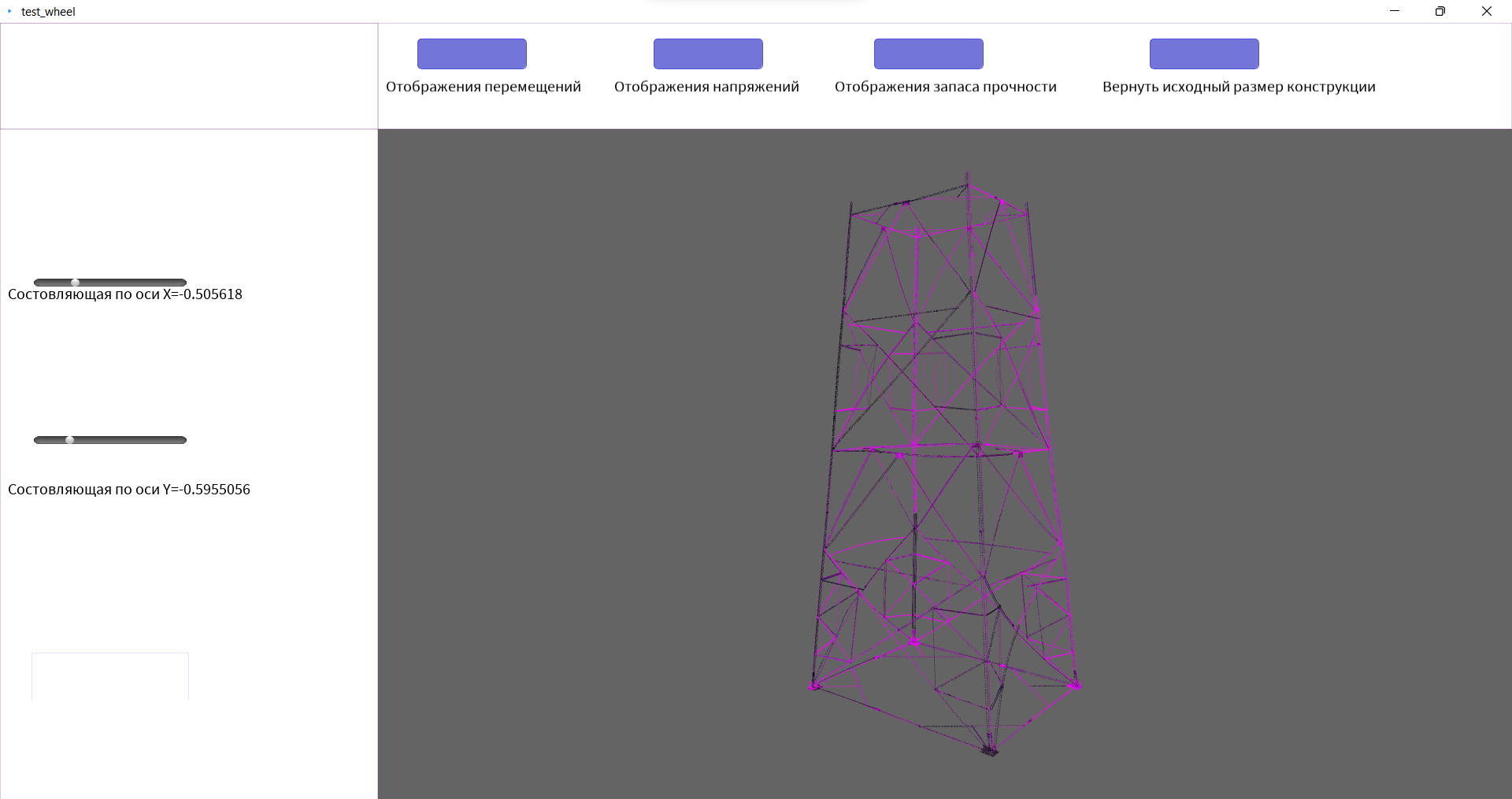


Рисунок 17 – Результат использования слайдеров

Результат: составляющие принимают значения, отображённые в программе(Рис. 17)

2. Переключение между режимами отображения

Описание теста: проверка работоспособности кнопок переключения режима отображения конструкции

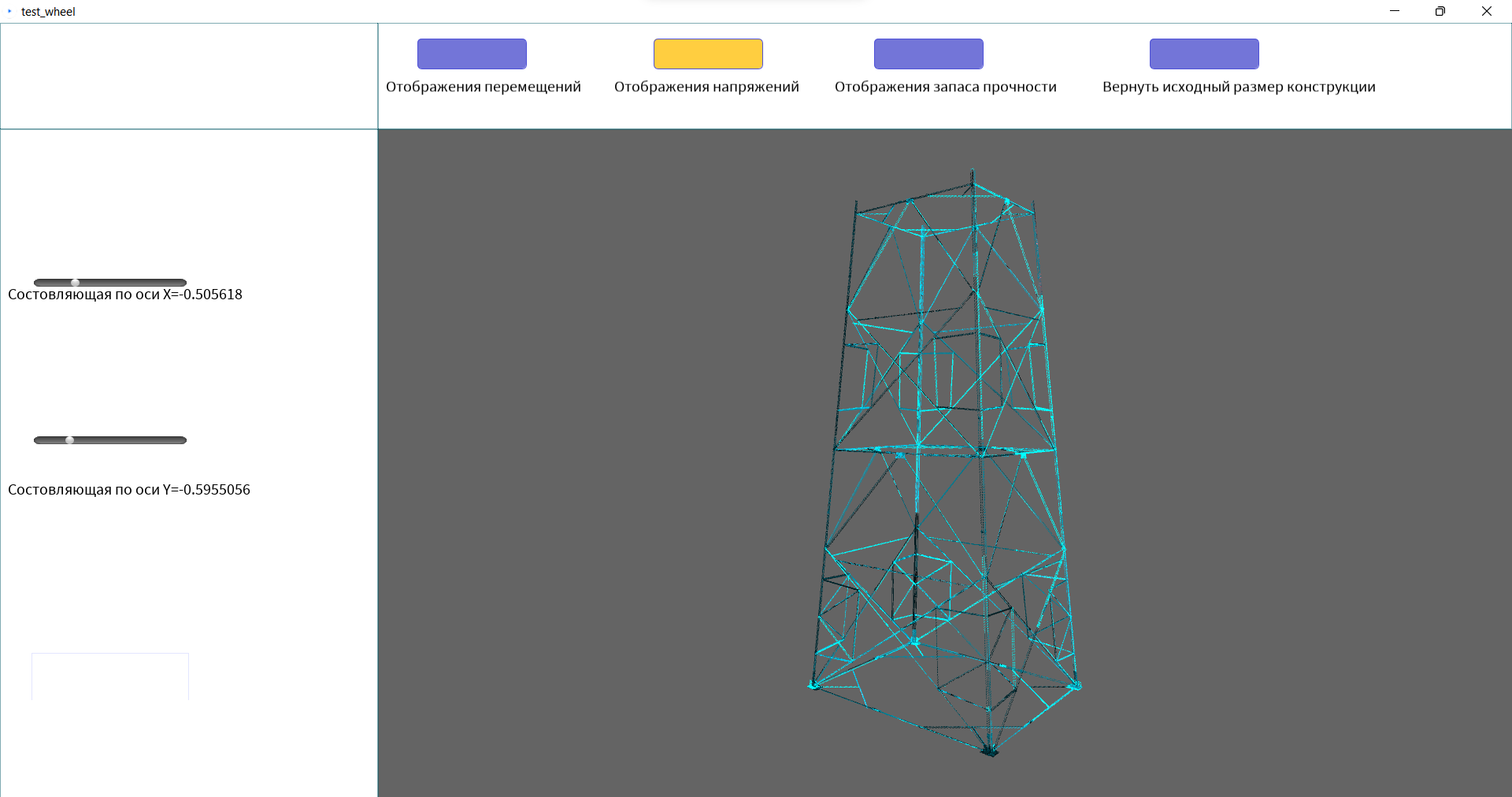
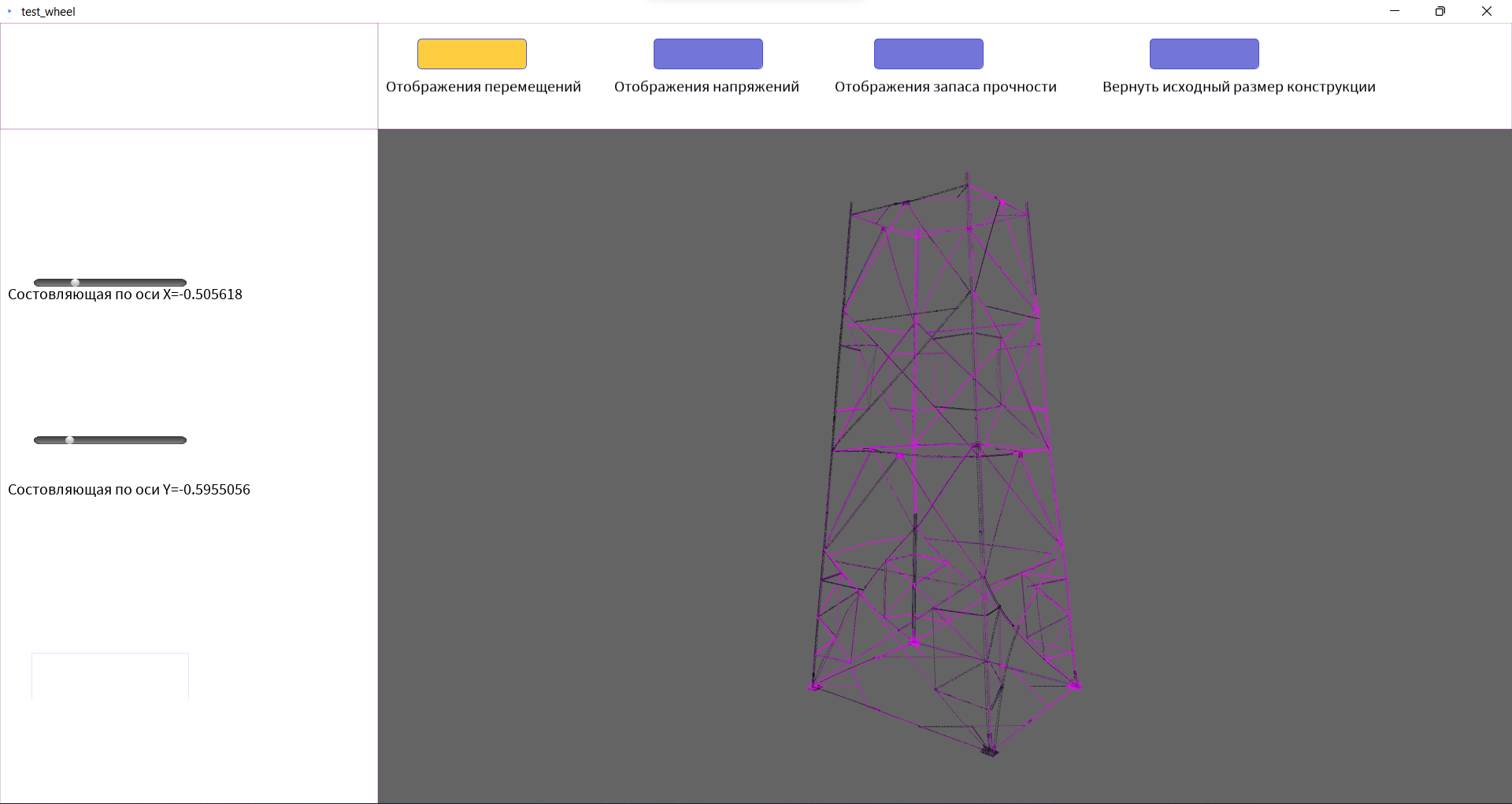


Рисунок 18 – Результат кнопок переключения режимов отображения

Результат: успешное отображение режима нагружения, перемещения и режима остаточного ресурса(Рис. 18)

3. Проверка возможности масштабирования конструкции

Описание теста: проверка возможности изменения размера отображенной конструкции, путем измпользования колеса мыши

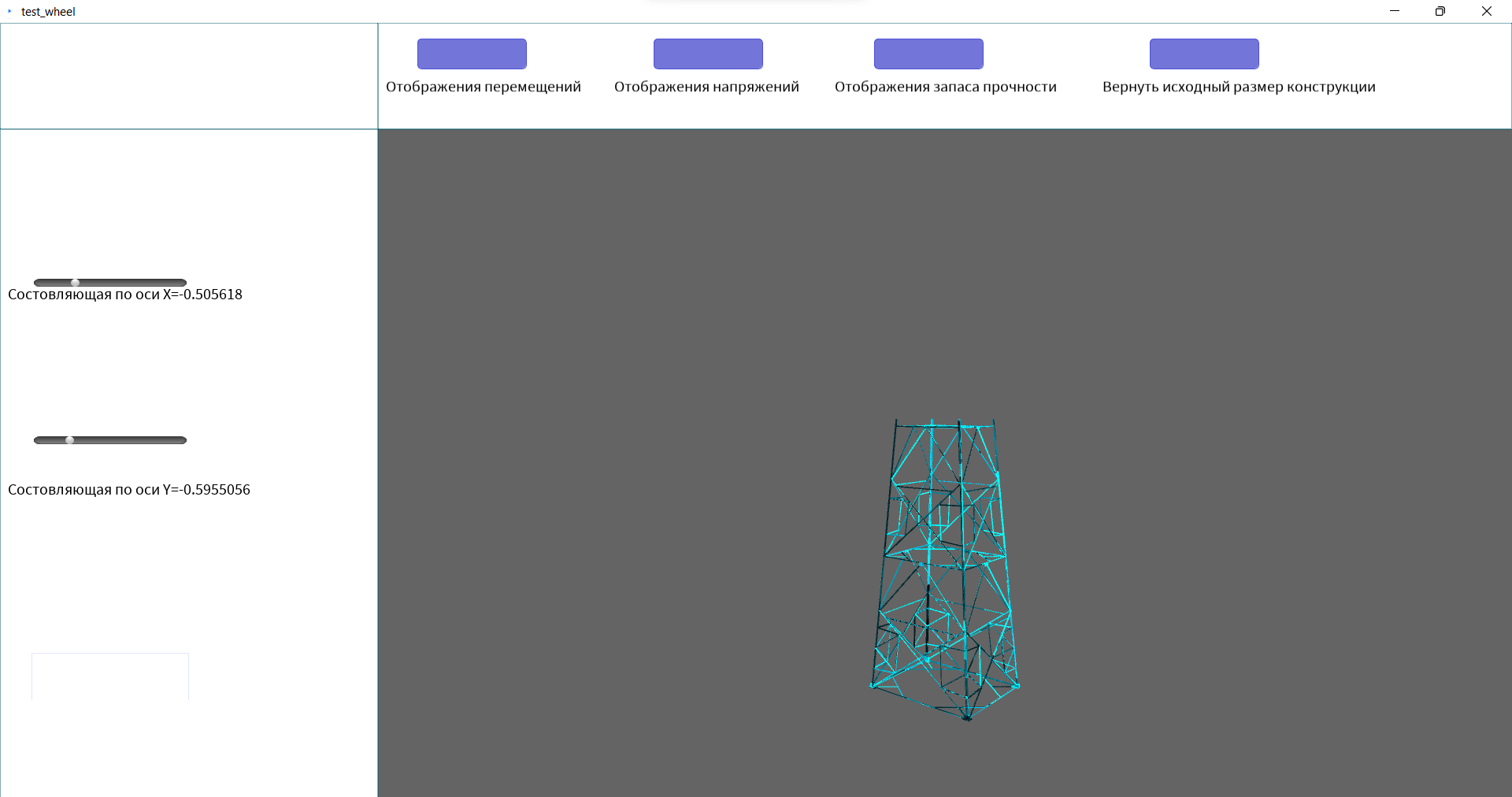


Рисунок 19 – Уменьшенный вид конструкции

Результат: конструкция изменяет свой размер в соответствии с направлением вращения колеса мыши и не переходит ограничительные рамки(Рис. 19).

Результаты тестирования: тестирование интерфейса системы было успешным. Все функции и возможности системы доступны пользователю без проблем.

**ЗАКЛЮЧЕНИЕ**

В ходе выполненной работы были разработаны элементы автоматизированной системы мониторинга воспринимаемых металлоконструкцией нагрузок.

Разработан модуль сбора данных, модуль определения действительных напряжений.

Разработка системы началась с построения КЭ-модели конструкции в NX2012, который в автоматическом режиме рассчитывал перемещения КЭ конструкции и сохранял результаты вычисления в виде таблицы Microsoft Excel.

Разработана программа в среде Processing IDE помощью для языка Java и Processing .

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Критерии качества и алгоритм выбора редуцированных моделей для мониторинга технических конструкций [Электронный ресурс] // Математическое моделирование и численные методы, 2016, выпуск 12.URL:https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=mmcm&paperid=88&option\_lang=rus (дата обращения: 18.12.22)
2. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – М.: Стандартинформ, 2009.
3. Справка Nastran NX 2012. [Электронный ресурс] // Siemens AG. URL:https://docs.plm.automation.siemens.com/tdoc/nxnastran/12/help/(дата обращения: 18.12.22)
4. Феодосьев В.И. Сопротивление материалов: Учеб. для вузов. - 10-е изд., перераб. и доп. М.: Изд-во МГТУ им. Н.Э. Баумана,. 1999.
5. ГОСТ 25.502 – 79. Расчеты и испытания на прочность в машиностроении. Методы механических испытаний материалов. Методы испытаний на усталость– М.: Стандартинформ, 2009.

**[ПРИЛОЖЕНИЕ А](#_heading=h.19c6y18)**

import processing.core.\*;

import processing.data.\*;

import processing.event.\*;

import processing.opengl.\*;

import processing.\*;

import java.\*;

import g4p\_controls.GWindow;

import g4p\_controls.\*;

GButton btnMOVE;

GButton btnTENS;

GButton btnHP;

GButton btnCOEF1;

GCustomSlider sldrX;

GCustomSlider sldrY;

GTextBase nmb\_X;

GTextBase nmb\_Y;

float X\_float, Y\_float;

float wheel;

PVector Pos\_of\_main;

Table table, tableE;

int n=3000000;

float [][][] xx = new float[n][5][4];

float [][][] xd = new float[n][5][4];

float [][][] xt = new float[n][5][4];

float [][][] hp = new float[n][5][4];

float max\_hp=3300000000.0;

float []tension\_x = new float[n];

float []tension\_y = new float[n];

//float [][][] xd = new float[n][5][4];

int ee=0;

int ee1;

int k, k1, k2, k3;

int x, y;

int n2=100;

int nF=100;

int [][] F = new int[nF][nF];

float [][] xg = new float[n2+10][6];

float e, coef;

float [][][] constr=new float[n][5][4];

float displayWidth\_rect\_1;

float displayWidth\_rect\_2;

float displayHeight\_rect\_1;

float displayHeight\_rect\_2;

float xc, yc, zc, dx, dy, dz, r1, r2, r3, xb1, xb2, xb3, yb1, yb2, yb3, rb, s;

PImage lin, cur, cir, back, b1, b2;

float Sc=0.8;

float max1=0;

float max2=0;

float min1=0;

float min2=0;

float Scd=200;

boolean state\_C=false;

boolean state\_b1=false;

boolean state\_b2=false;

boolean state\_b3=false;

int imax1=0;

int imax2=0;

int Fpr1, Fpr2;

void settings() {

size(displayWidth, displayHeight, P3D);

table = loadTable("data.csv", "header");

tableE = loadTable("tensionE.csv", "header");

k1=0;

k=0;

xc=0;

yc=0;

zc=0;

displayWidth\_rect\_1=displayWidth\*4/16;

displayWidth\_rect\_2=displayWidth;

displayHeight\_rect\_1=displayHeight;

displayHeight\_rect\_2=displayHeight\*2/16;

println(displayWidth, displayHeight);

for (TableRow row : table.rows()) { // загрузка координат в массив

ee = row.getInt(0);

k1=k1+1;

xx[k1][0][0] = row.getFloat(1)\*0.03f;// координата X

xx[k1][1][0] = row.getFloat(5)\*0.03f;// координата Y

xx[k1][2][0] = row.getFloat(9)\*0.03f;// координата Z

xx[k1][0][1] = row.getFloat(2)\*0.03f;// компоненты вектора перемещений при действии нагружения 1 X

xx[k1][1][1] = row.getFloat(6)\*0.03f;// компоненты вектора перемещений при действии нагружения 1 Y

xx[k1][2][1] = row.getFloat(10)\*0.03f;// компоненты вектора перемещений при действии нагружения 1 Z

xx[k1][0][2] = row.getFloat(3)\*0.03f;// компоненты вектора перемещений при действии нагружения 2 X

xx[k1][1][2] = row.getFloat(7)\*0.03f;// компоненты вектора перемещений при действии нагружения 2 Y

xx[k1][2][2] = row.getFloat(11)\*0.03f;// компоненты вектора перемещений при действии нагружения 2 Z

xx[k1][0][3] = row.getFloat(4)\*0.03f;//

xx[k1][1][3] = row.getFloat(8)\*0.03f;//

xx[k1][2][3] = row.getFloat(12)\*0.03f;//

xd[k1][0][0] = row.getFloat(13)\*0.002f;//

xd[k1][0][1] = row.getFloat(17)\*0.002f;//

xd[k1][0][2] = row.getFloat(21)\*0.002f;//

}

k3=0;

for (TableRow rowE : tableE.rows()) { // загрузка координат в массив

ee = rowE.getInt(0);

xt[k3][0][0] =rowE.getFloat(11);// Von-mises X

xt[k3][1][0] = xt[k3][0][0];// Von-mises Y

k3=k3+1;

}

}

public void setup() {

surface.setResizable(true);

surface.setLocation(100, 100);

coef=1;

nmb\_X=new GTextField (this, 40, 800, 200, 60);

nmb\_Y=new GTextField (this, 40, 1000, 200, 60);

sldrX = new GCustomSlider(this, 40, 300, 200, 60, null);

sldrY = new GCustomSlider(this, 40, 500, 200, 60, null);

btnMOVE = new GButton(this, 530, 20, 140, 40, " ");

btnTENS = new GButton(this, 830, 20, 140, 40, " ");

btnHP = new GButton(this, 1110, 20, 140, 40, " ");

btnCOEF1 = new GButton(this, 1460, 20, 140, 40, " ");

lin = loadImage("lin2.png");

cur = loadImage("cur.png");

cir = loadImage("s2.png");

b1 = loadImage("b1.png");

b2 = loadImage("b2.png");

back = loadImage("back2.png");

back.resize(int(displayWidth\_rect\_2), int(displayHeight\_rect\_1));

xb1=displayWidth\*0.7;

yb1=displayHeight\*0.15;

xb2=displayWidth\*0.8;

yb2=displayHeight\*0.15;

xb3=displayWidth\*0.9;

yb3=displayHeight\*0.15;

rb=displayHeight\*0.15;

}

public void draw() {

background(100, 100, 100);

rect(0, 0, displayWidth\_rect\_1, displayHeight\_rect\_1);

fill(255);

rect(0, 0, displayWidth\_rect\_2, displayHeight\_rect\_2);

fill(255);

textSize(20);

fill(0, 0, 0);

text("Отображения перемещений", 490, displayHeight\_rect\_2/2+20);

text("Отображения напряжений", 780, displayHeight\_rect\_2/2+20);

text("Отображения запаса прочности", 1060, displayHeight\_rect\_2/2+20);

text("Вернуть исходный размер конструкции", 1400, displayHeight\_rect\_2/2+20);

text("Состовляющая по оси Х="+(X\_float-0.5)\*2, 10, height\*3/8-20);

text("Состовляющая по оси Y="+(Y\_float-0.5)\*2, 10, height\*5/8-20);

translate(width\*5/8, height\*0.85);

rotateX(PI/2);

rotateZ(PI/4);

if (wheel==1) {

coef=coef-0.03;

if (coef<0)

{

coef=0;

}

if (coef>1.1)

{

coef=1.15;

}

}

if (wheel==-1) {

coef=coef+0.03;

if (coef<0)

{

coef=0;

}

if (coef>1.1)

{

coef=1.1;

}

}

fill(255);

stroke(255);

thread("requestData1");

thread("requestData2");

thread("requestData3");

thread("requestData4");

thread("requestData5");

thread("requestData6");

thread("requestData7");

if (state\_b1) {//режим отображения перемещений

for (int i = 0; i < k1; i++) {

stroke(int(tension\_x[i])\*50, 0, int(tension\_y[i])\*50, 100);

point(constr[i][0][0], constr[i][0][1], constr[i][0][2]);

}

}

if (state\_b2) {//режим отображения напряжений

for (int i = 0; i < k1; i++) {

stroke(s, xt[i][0][0]\*40\*abs(X\_float-0.5)\*2, xt[i][0][0]\*40\*abs(Y\_float-0.5)\*2);

point(xx[i][0][0]\*coef, xx[i][1][0]\*coef, xx[i][2][0]\*coef);

}

}

if (state\_b3) {//режим отображения запаса хп

for (int i = 0; i < k1; i++) {

stroke(255\*( (hp[i][0][0]+hp[i][1][0])/max\_hp), 0, 0);

point(xx[i][0][0]\*coef, xx[i][1][0]\*coef, xx[i][2][0]\*coef);

}

}

wheel=0;

}

public void mouseWheel(MouseEvent event) {

float e = event.getCount();

println(e);

wheel=e;

}

void requestData1() {

if (state\_b1) {

for (int i = 0; i < n/4; i++) {

// if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

constr[i][0][0]=(xx[i][0][0]+xd[i][0][0]\*(X\_float-0.5)\*500)\*coef;

constr[i][0][1]=(xx[i][1][0]+xd[i][0][1]\*(Y\_float-0.5)\*500)\*coef;

constr[i][0][2]=(xx[i][2][0]+xd[i][0][2]\*(Y\_float-0.5)\*500)\*coef;

// }

}

}

}

void requestData2() {

if (state\_b1) {

for (int i = n/4; i < n/2; i++) {

// if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

constr[i][0][0]=(xx[i][0][0]+xd[i][0][0]\*(X\_float-0.5)\*500)\*coef;

constr[i][0][1]=(xx[i][1][0]+xd[i][0][1]\*(Y\_float-0.5)\*500)\*coef;

constr[i][0][2]=(xx[i][2][0]+xd[i][0][2]\*(Y\_float-0.5)\*500)\*coef;

// }

}

}

}

void requestData3() {

if (state\_b1) {

for (int i = n/2; i < n\*3/4; i++) {

// if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

constr[i][0][0]=(xx[i][0][0]+xd[i][0][0]\*(X\_float-0.5)\*500)\*coef;

constr[i][0][1]=(xx[i][1][0]+xd[i][0][1]\*(Y\_float-0.5)\*500)\*coef;

constr[i][0][2]=(xx[i][2][0]+xd[i][0][2]\*(Y\_float-0.5)\*500)\*coef;

// }

}

}

}

void requestData4() {

if (state\_b1) {

for (int i = n\*3/4; i < n; i++) {

// if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

constr[i][0][0]=(xx[i][0][0]+xd[i][0][0]\*(X\_float-0.5)\*500)\*coef;

constr[i][0][1]=(xx[i][1][0]+xd[i][0][1]\*(Y\_float-0.5)\*500)\*coef;

constr[i][0][2]=(xx[i][2][0]+xd[i][0][2]\*(Y\_float-0.5)\*500)\*coef;

//}

}

}

}

void requestData5() {

if (state\_b1) {

for (int i = 0; i < n; i++) {

//if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

tension\_x[i]=xt[i][0][0]\*abs(X\_float-0.5)\*2;

tension\_y[i]=xt[i][1][0]\*abs(Y\_float-0.5)\*2;

//}

}

}

}

void requestData6() {

if (state\_b2) {

for (int i = 0; i < n; i++) {

//if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

s=xt[i][0][0]\*abs(X\_float-0.5)\*2+xt[i][1][0]\*abs(Y\_float-0.5)\*2;

// }

}

}

}

void requestData7() {

if (state\_b3) {

for (int i = 0; i < n; i++) {

//if (mouseX>displayWidth\_rect\_1 && mouseY>displayHeight\_rect\_2) {

hp[i][0][0]=hp[i][0][0]+xt[i][0][0]\*abs(X\_float-0.5)\*2;

hp[i][1][0]=hp[i][1][0]+xt[i][1][0]\*abs(Y\_float-0.5)\*2;

// }

}

}

}

public void handleButtonEvents(GButton button, GEvent event) {

if (button == btnMOVE && event == GEvent.CLICKED) {

state\_b1=true;

state\_b2=false;

state\_b3=false;

//println("Button 'btnMOVE' was clicked");

}

if (button == btnTENS && event == GEvent.CLICKED) {

state\_b1=false;

state\_b2=true;

state\_b3=false;

//println("Button 'btnTENS' was clicked");

}

if (button == btnHP && event == GEvent.CLICKED) {

state\_b1=false;

state\_b2=false;

state\_b3=true;

//println("Button 'btnHP' was clicked");

}

if (button == btnCOEF1 && event == GEvent.CLICKED) {

coef=1.0;

//println("Button 'btnHP' was clicked");

}

}

public void handleSliderEvents(GValueControl slider, GEvent event) {

// println("Slider Value " + sldrX.getValueI());

println( sldrX.getValueF());

X\_float = sldrX.getValueF();

// println("Slider Value " + sldrY.getValueI());

println( sldrY.getValueF());

Y\_float = sldrY.getValueF();

}

public void handleTextEvents(GEditableTextControl textcontrol, GEvent event)

{

println("Button 'btnHP' was clicked");

}