В данной работе рассматривается классическая задача из сопромата: балка(двутавр), закреплена на двух шарнирных опорах с двух концов, на балку могут быть приложены сосредоточенные силы(перпендикулярно балке) и моменты.

Есть три режима работы:

1)Ручной режим - пользователь сам задает куда приложить и какие по значению силы и моменты, на основе которых строятся эпюры поперечных сил, моментов и прогибов, также балка визуализируется(цвета визуализации зависят от напряжений, бордовый - критическая нагрузка, фиолетовый - нет нагрузки). Черные линии на визуализации показывают приложенные силы, красные - моменты. Режим сглаженной визуализации добавил просто для красоты, он сглаживает визуализацию балки при малом количестве точек расчета(так как Python по дефолту использует только одно ядро для расчетов). Также есть возможность выбирать профиль балки.

2)Автоматический режим - здесь пользователь выбирает только количество приложенных к балке сил, моментов и профиль двутавра, далее каждый тик значение каждой силы, момента и точки их приложения немного изменяются, эпюры и визуализация балки аналогична ручному режиму.

3)Обратный режим - здесь по рандомному набору значений перемещений строиться функция(график исходных перемещений), по которой восстанавливаются значения приложенных сил, по полученным силам строиться новый график перемещений(график восстановленных перемещений) для сравнения с исходным.

Файлы в проекте:

Core:

beam\_load\_simulator.py - класс, генерирующий значения сил и моментов и изменяющий их со временем для автоматического режима

beam\_loader.py - преобразует значения о нагрузках из .json формата в питоновский массив

beam\_solver.py - класс в котором прописаны методы расчетов значений для построения эпюр

reverce\_solver.py - генерирует рандомные значения перемещений и рассчитывает силы для обратного режима

Data:

profiles.json - база данных прифилей двутавров и их характеристик

остальные файлы не используются, на всякий случай оставил

GUI:

тут много файлов, каждый из которых содержит виджеты, все для создания интерфейса

Main.py - cлужит для запуска всей программы

Вычисления:

Для построения эпюр моментов(beam\_solver.py):

Рассчитываем реакции опор:

Затем идем слева направо и складываем все моменты с учетом знака

Для построения эпюр напряжений(beam\_solver.py):

Для построения эпюр сил(beam\_solver.py):

Аналогично моментам, находим реакции опор и слева направо добавляем силы

Для построения эпюр удлинений(beam\_solver.py):

Берем массив моментов, преобразуем массив значений в интерполирующую функцию, два раза интегрируем полученную функцию, в подставляем значения z(0) = z(l) = 0 для определения констант интегрирования и делим на EI

Расчеты для обратной задачи(reverce\_loader.py):

Получение случайных значений:

Рандом реализовал через суперпозицию синусоидальных мод со случайными коэффициентами, так было проще учесть что z(0) = z(l) = 0 и получаемая визуально похожа на реальные графики перемещений. Затем представляем функцию в виде массива.

Расчет сил:

Массив точек, полученный из предыдущей функции интерполируем в полином, дифференцируем его 4 раза, для получения q(z), делим полученную кривую на n отрезков, для каждого находим точку приложения() и значение эквивалентной силы(F):