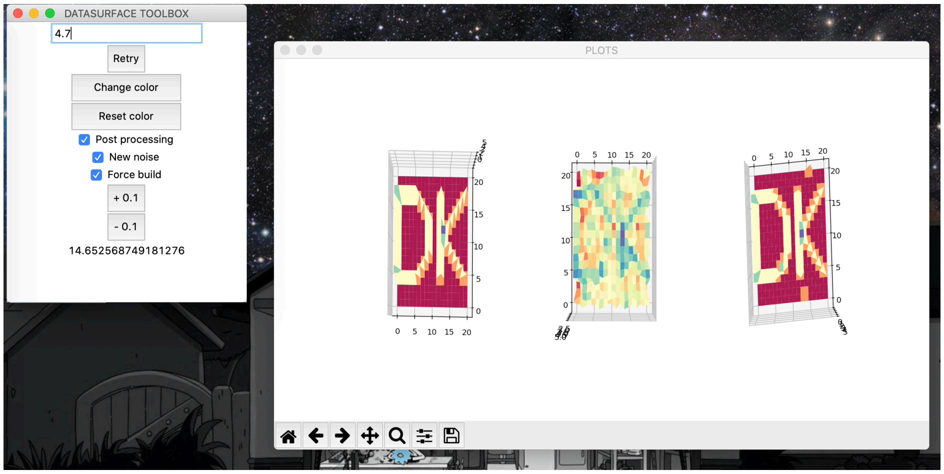
Изменения в приложении

Для начала я решил превратить свою программу с изображениями в тестовую среду, где можно удобно проверять результаты работы или менять параметры задачи.

Для этого я создал 2 окна: 1 с самими графиками, а другое с элементами управления.



Пока что для удобства здесь очень ограниченный функционал, и сделано все средствами питона 3.8 и библиотеки tkinter. Изначально я все сделал в одном окне, но оказалось, что в таком случае ломается разрешение графиков и их очень плохо видно, как будто изображение очень размыто. Я долго искал в гугле решение, но все пишут, что это баг Макинтоша и его экрана Ретина. Возможно есть смысл переписать на C#, тогда визуально будет более красиво.

Перечислю что означают элементы из меню по порядку сверху вниз:

1. Это блок текста, в котором отображается значение уровня ошибки, его можно туда вписать руками или нажимать на кнопки и уровнь будет меняться. Соответственно с изменением этого значения графики будут перестраиваться.

2. Кнопка Retry запускает процесс попытки восстановления изображения

3. Change color меняет цвет графиков, т.к. иногда на некоторый тестах плохо различимы отличия в определенном цвете.

4. Reset color возвращает цвет по-умолчанию

5. Post processing нужен для восстановления "интенсивности" изображения. Дело в том, что после процесса минимизации функционала восстановленный результат теряет в своих значениях значение самой

ошибки. То есть если в определенной точке изначально было значение 5, то в восстановленном изображении с уровнем ошибки 4.7 значение будет 0.3. Post processing добавляет значение ошибки к результату, чтобы он получился близким к идеальному. Но на результатах работы это не отражается и даже без этой функции видно то изображение, которое надо было восстановить.

6. New noise кнопка которая регулирует нужно ли по-новому зашумить изображение при новой попытке или попробовать снова восстановить текущее.

7. Force build нужен для форсирования построения. Иногда процесс минимизации функционала не удается выполнить из-за локальных минимумов или прочих ошибок алгоритма. При включенной кнопке Force build программа будет пробовать восстановить пока ей этого не удастся. При выключенной будет показывать ошибку, из-за чего все пошло не так. Эта функция нужна была мне для отладки.

8. Далее идут кнопки увеличения/уменьшения уровня ошибки, можно добавить больше вариантов, но легче самому вписывать нужные значения.

9. И последнее это коэффициент достоверности восстановления. Я сравниваю исходное и восстановленное изображение по точками и вычисляю модуль разности в каждой точке, далее суммирую эти разности и вывожу на экран. Чем меньше этот коэффициент, тем достовернее получилось восстановление.

Во втором окне как и раньше 3 графика слево направо: Исходное изображение, изображение с добавлением шумов, восстановленное изображение.

Всю программу можно доработать, добавить больше функционала и возможностей. Например, возможность загрузить изображение из файла или может отрисовывать пошагово процесс восстановления для наглядности. Но лучше делать это не средствами питона, а через java script или C#.

Что делал.

1.

Для начала я переделал способ вычисления интеграла площади

$$S = \min_{\forall u(x,y) \in U} \iint\limits_{\Omega} \sqrt{1 + (\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2} dxdy$$

Вычисляю я его по кубатурной формуле Симпсона.

$$\iint\limits_{\Omega} f(x,y)dxdy = \frac{hk}{9} \sum_{i=0}^{2n} \sum_{j=0}^{2m} \lambda_{ij} f_{ij}$$

Где λ_{ij} являются соответствующими элементами матрицы

$$\Lambda = \begin{bmatrix} 1 & 4 & 2 & 4 & 2 & ... & 4 & 2 & 4 & 1 \\ 4 & 16 & 8 & 16 & 8 & ... & 16 & 8 & 16 & 4 \\ 2 & 8 & 4 & 8 & 4 & ... & 8 & 4 & 8 & 2 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 2 & 8 & 4 & 8 & 4 & ... & 8 & 4 & 8 & 2 \\ 4 & 16 & 8 & 16 & 8 & ... & 16 & 8 & 16 & 4 \\ 1 & 4 & 2 & 4 & 2 & ... & 4 & 2 & 4 & 1 \end{bmatrix}$$

а f_{ij} - значения функции в точках (i, j).
Реализацию написал самостоятельно. Просто на каждом шаге стою марицу и вычисляю интеграл.

2.

Далее моя программа все равно не заработала, и я начал тестировать различные методы численного интегрирования. У меня было **3** варианта формул, по которым считаются интегралы из

$$u = \frac{1}{2}(\int_{x_0}^{x} q(\xi, y)d\xi + \int_{y_0}^{y} w(x,\xi)d\xi + u(x, y_0) + u(x_0, y)).$$

1. **Формула прямоугольников**(правых или левых одинаково) по итогу оказалась самой рабочей.

$$\int_a^b f(x)dx = \sum_{i=0}^{n} f(x_i)(x_{i+1} - x_i).$$

2. **Формула трапеций**

$$\int_a^b f(x)dx = h * (\frac{f(x_0) + f(x_n)}{2}) \sum_{i=1}^{n-1} f(x_i).$$

3. **Формула Симпсона**

$$\int_a^b f(x)dx = \frac{h}{3} * (f(x_0) + f(x_{2N}) + 2\sum_{j=2,2}^{2N-2} f(x_j) + 4\sum_{j=1,2}^{2N-1} f(x_j)).$$

Для каждого варианта я написал свой вариант программы, что по итогу оказалось трудоемким и немного бесполезным процессом из-за разрывности функции рисунка. Все эти формулы получаются из замены подынтегральной функции интерполяционными многочленами разных степеней, но так как табличные данные – это поточечно заданная функция, то интерполировать ее у меня не получилось. В картинке нельзя исследовать промежуточное значение функции по известным значениям, так как все значения взяты произвольно.

Ход программы

В этом разделе описаны шаги восстановления изображения, описанные в приложении.

Заполнение тестовых данных

Для начала я заполняю исходную матрицу так, чтобы получался рисунок. В матрицу [20x20] в нужные точки кладу значение lVal, а в пустых узлах оставляю 0.

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	1
0		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4		0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00
5		0.00	5.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00
6		0.00	5.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00
7		0.00	5.00	0.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00
8		0.00	5.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00
9		0.00	5.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00
10		0.00	5.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00
11		0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00
12		0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00	0.00	5.00	0.00	0.00
13		0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	0.00	5.00	0.00	0.00
14		0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	0.00	5.00	0.00	0.00
15		0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00	5.00	0.00	0.00
16		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
17		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
18		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

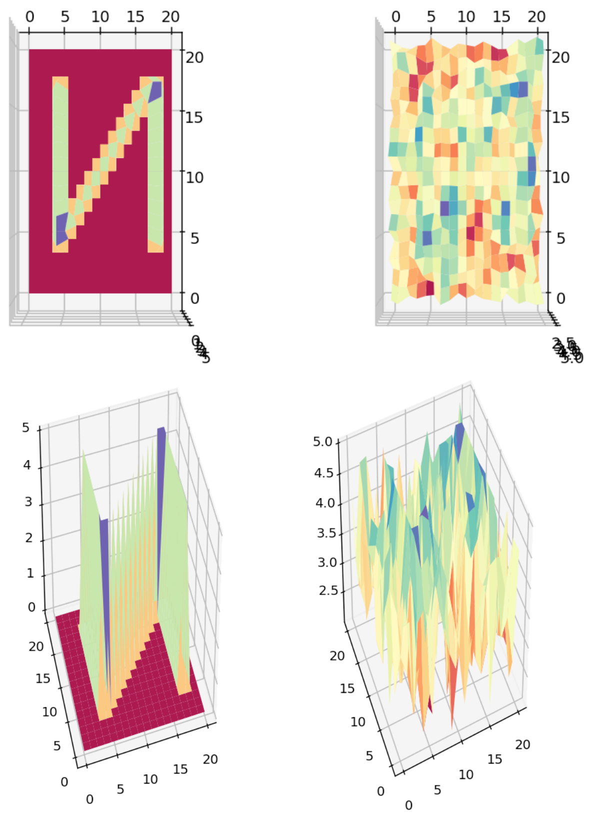
Пример для буквы N и lVal = 5

Далее идет добавление шумов к картинке. Используя генератор рандомных чисел, я к каждой точке добавляю случайное вещественное число $\leq lVal$. Если шумы больше, чем "интенсивность" самого рисунка, то возникает неопределенность при минимизации, и такой рисунок не удается однозначно восстановить.

	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	966	4.05949	4.96576	5.25470	2.13643	3.21608	2.93383	2.25939	2.06596	5.03503	4.33150	5.00199	2.68332	4.92624	2.67823	2.32855
1	333	3.11662	2.68770	4.78467	3.77294	2.86465	3.16405	4.30241	3.70693	5.87120	3.43579	2.89690	4.84195	3.90359	4.77736	5.73716
2	316	3.86011	4.00364	5.66585	5.97329	5.75252	2.00741	5.13065	5.93302	5.15795	2.32251	4.17235	2.02694	4.95339	2.57068	2.61124
3	708	2.84828	3.30496	4.56811	2.74681	5.90348	4.70525	3.61566	2.50105	4.20197	4.21605	5.69348	2.61760	2.35022	5.62250	2.26532
4	427	5.00000	2.43524	2.01900	4.76539	5.51764	2.97630	2.22324	5.07316	5.74437	2.68605	3.08195	5.35131	5.00000	3.64914	2.98217
5	375	5.00000	5.00000	2.59817	3.63914	2.52693	4.54915	2.10092	4.23030	3.05776	2.83623	3.49574	4.40033	5.00000	2.58233	5.02061
6	072	5.00000	2.04914	5.00000	4.47254	5.28185	5.57961	2.53919	5.03393	4.74542	2.07113	2.21115	2.82969	5.00000	3.06095	2.21775
7	233	5.00000	5.55037	4.18986	5.00000	4.97413	2.25722	5.57084	2.98096	5.07083	4.75492	5.32640	5.19618	5.00000	2.08729	5.60117
8	33	5.00000	4.25458	3.82724	3.78172	5.00000	3.83950	5.48190	2.11154	4.95342	4.56132	3.18634	3.38470	5.00000	2.89002	4.17421
9	556	5.00000	4.23231	4.19828	2.69053	3.74672	5.00000	4.03937	4.59528	2.84294	3.26961	2.66338	5.44775	5.00000	3.23787	3.14753
10	348	5.00000	5.80015	5.26268	5.75744	3.86916	5.93180	5.00000	4.57101	4.17309	5.75196	2.71737	5.76003	5.00000	5.88165	5.05278
11	097	5.00000	2.20798	3.11637	2.07799	2.14197	4.23637	2.30508	5.00000	5.74033	4.55650	5.20194	5.00196	5.00000	5.86303	3.26657
12	794	5.00000	5.06517	4.23552	2.07243	5.44448	2.09497	4.85530	4.12568	5.00000	4.11284	5.69588	4.55721	5.00000	3.68551	3.83965
13	746	5.00000	4.10107	4.24553	4.86634	4.80854	2.75390	2.59833	2.52591	2.34091	5.00000	5.60085	3.41078	5.00000	5.32575	4.84836
14	364	5.00000	2.04868	5.54679	2.38995	2.07049	2.55390	5.22444	4.69021	5.64811	4.95868	5.00000	2.90695	5.00000	2.99281	2.40583
15	214	5.00000	2.04003	4.75429	4.64093	2.85450	3.77501	4.85143	2.61414	5.94256	4.30650	5.34780	5.00000	5.00000	4.52407	5.77316
16	952	4.57149	2.76608	5.33506	4.78742	3.04225	5.65216	5.81135	2.57306	3.29156	4.21160	3.07960	3.46204	5.36450	2.55355	2.85043
17	575	2.73906	2.72606	3.09537	4.10243	4.41255	4.79981	3.65734	3.98430	5.51539	2.23103	4.64240	2.45090	2.02664	3.07948	2.21950
18	914	2.37101	5.91737	2.44522	4.49601	3.77028	3.41061	4.44377	2.69067	5.09215	4.90557	2.73396	4.55565	3.55037	2.35300	2.31636

Зашумленная матрица

Вот так выглядят исходная и зашумленная матрицы с буквой



Восстановление изображения

Далее начинается процесс восстановления. Пусть $u_{ij} = u(x_i, y_j)$;
В формуле для $u_{ij} = \frac{1}{2}(\int_{x_0}^{x_i} q(\xi, y)d\xi + \int_{y_0}^{y_j} w(x, \xi)d\xi + u_{i0} + u_{oj})$ используются значения u_{i0} и u_{oj}. Поэтому сначала вычисляются граничные значения u_{i0} и u_{oj} $\forall i \in 1, 2, ..., N$ $\forall j \in 1, 2, ..., N$ по формулам:

$$u_{i0} = \frac{1}{2}(\int_{x_0}^{x_i} q(\xi, y)d\xi + u_{00} + u_{i0}) = \int_{x_0}^{x_i} q(\xi, y)d\xi + u_{00}$$

$$u_{0j} = \frac{1}{2}(\int_{y_0}^{y_i} w(x, \xi)d\xi + u_{00} + u_{0j}) = \int_{y_0}^{y_i} w(x, \xi)d\xi + u_{00}$$

Соответственно для каждого из 3 вариантов я запрограммировал следующие формулы:

1. **Прямоугольники**

$$u_{i0} = \sum_{k=0}^{i} q_{k0} + u_{00}; \quad u_{0j} = \sum_{k=0}^{j} w_{0k} + u_{00};$$

2. **Трапеции**

$$u_{i0} = \frac{q_{00} + q_{i0}}{2} + \sum_{k=1}^{i-1} q_{k0} + u_{00}; \quad u_{0j} = \frac{w_{00} + w_{0j}}{2} + \sum_{k=1}^{j-1} w_{0k} + u_{00};$$

3. **Симпсон**

$$u_{i0} = \frac{1}{3}(q_{00}+q_{i0}+2\sum_{k=2,2}^{i-1} q_{k0}+4\sum_{k=1,2}^{i-1} q_{k0})+u_{00}; \quad u_{0j} = \frac{1}{3}(w_{00}+w_{0j}+2\sum_{k=2,2}^{j-1} w_{0k}+4\sum_{k=1,2}^{j-1} w_{0k})+u_{00};$$

Минимизация функционала площади выполняется по переменным $q = \frac{\partial u}{\partial x}$ и $w = \frac{\partial u}{\partial y}$. В начале работы алгоритма заполняются ограничивающие неравенства $|u_{ij} - a_{ij}| \leq \sigma$ относительно искомых q_{i0} и w_{0j}. Далее минимизируются функционал $\int \sqrt{1 + r^2(x)}dx$. Получаем искомые минимальные q_{i0} и w_{0j} и вычисляются u_{i0} и u_{0j}

После вычисления всех граничных значений происходит вычисление во внутренних узлах сетки.

1. **Прямоугольники**

$$u_{ij} = \frac{1}{2}(\sum_{k=0}^{i} q_{kj} + \sum_{k=0}^{j} w_{ik} + u_{i0} + u_{0j});$$

2. **Трапеции**

$$u_{ij} = \frac{1}{2}(\frac{q_{0j} + q_{ij}}{2} + \sum_{k=1}^{i-1} q_{kj} + \frac{w_{0i} + w_{ij}}{2} + \sum_{k=1}^{j-1} w_{ik} + u_{i0} + u_{0j});$$

3. **Симпсон**

$$u_{ij} = \frac{1}{2}(\frac{1}{3}(q_{0j}+q_{ij}+2\sum_{k=2,2}^{i-1} q_{kj}+4\sum_{k=1,2}^{i-1} q_{kj})+\frac{1}{3}(w_{i0}+w_{ij}+2\sum_{k=2,2}^{j-1} w_{ik}+4\sum_{k=1,2}^{j-1} w_{ik})+u_{i0}+u_{0j});$$

Аналогично граничному случаю, сначала заполняются ограничивающие неравенства $|u_{ij} - a_{ij}| \leq \sigma$, затем минимизация исходного функционала площади $\iint\limits_{\Omega} \sqrt{1 + (q)^2 + (w)^2}dxdy$. Получаем искомые минимальные q_{ij} и w_{ij}. Подставив их обратно в формулу для u_{ij}, вычисляется сам u_{ij} и т.д.

Таким образом получается пошагово восстановленная матрица и она выводится на экран 3 графиком. В данный момент я использую 1 уровень ошибки на всех узлах сетки.

Также я ввел погрешность минимизации. Иногда в результате минимизации вместо 0 получаются невероятно маленькие, но положительные значения. Погрешность минимизации нужна, чтобы в таких случаях считать значения 0, т.к. иначе эти маленькие значения накапливаются и входят в сумму при численном интегрировании, что уменьшает точность результата.