Итоговое Задание 1

В данной работе нет слишком строгих условий, нет единственно верных решений, есть некоторый простор для творчества. Для решения задач потребуется почитать документацию и подумать.

Эпизод 1

У вас есть картинки поверхности Луны. Они чёрно-белые, и это нормально - такая была камера. К сожалению, качество их не очень - контраст страдает, вместо честных градаций от чёрного до белого камера зафиксировала 42 оттенка серого. Давайте улучшим картинки, чтобы они более внятно воспринимались глазом.

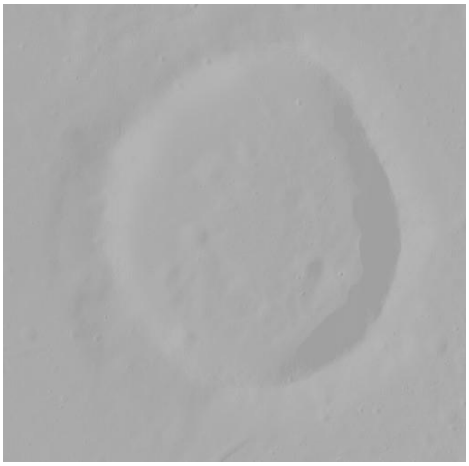
Картинки здесь.

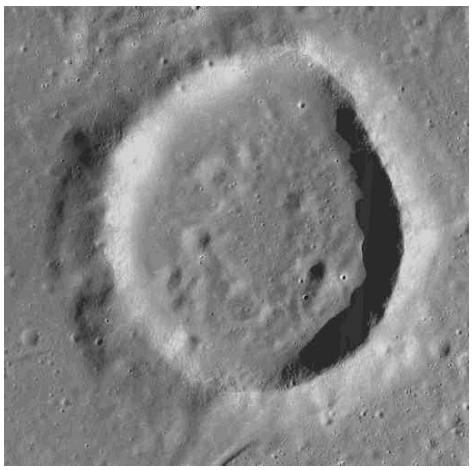
Задача - переинтерполировать картинки, использовав весь доступный диапазон для улучшения контраста. Логика преобразования такая:

1. чёрно-белая картинка - это по сути простой 2D-массив, каждый элемент которого представляет собой 1 пиксель картинки (с цветными всё несколько сложнее),
2. каждый пиксель может иметь значения от 0 до 255,
3. реально у нас на этих нерезких картинках все пиксели находятся где-то внутри небольшого диапазона значений (условно, от 10 до 50 вместо полного спектра),
4. поэтому нужно считать картинку и аккуратно линейно "растянуть" диапазон, минимальное значение на исходной картинки должно получить значение 0 в выходном изображении, а максимальное на исходной - 255 на выходе.

Внимание! Если когда-нибудь у вас в руках окажутся настоящие картинки, только что полученные с Луны, ни в коем случае не делайте так! Вернее, всегда сохраняйте оригиналы с точностью до последнего бита. И чётко сообщайте человечеству, где здесь исходные сырые данные из космоса, а где результат творческих преобразований и улучшений. Ну и не храните, конечно, столь ценные данные в форматах со сжатием и потерями, таких как jpg.

Пример того, как может выглядеть результат (это один из файлов до и после, их там несколько):





Для чтения и записи файлов можно поступить примерно так:

```
from PIL import Image

# считаем картинку в numpy array
img = Image.open(file_path)
data = np.array(img)

# ... логика обработки

# запись картинки после обработки
res_img = Image.fromarray(updated_data)
res_img.save(new_file_path)
```

Эпизод 2

У вас есть набор показаний, снятых с датчиков. Предполагается, что вообще-то в данных есть какие-то осмысленные сигналы, по которым можно проследить зависимости. К сожалению, данные очень сильно зашумлены.

К счастью, со случайными шумами можно неплохо бороться, усредняя сигнал по времени. Этим и предстоит заняться.

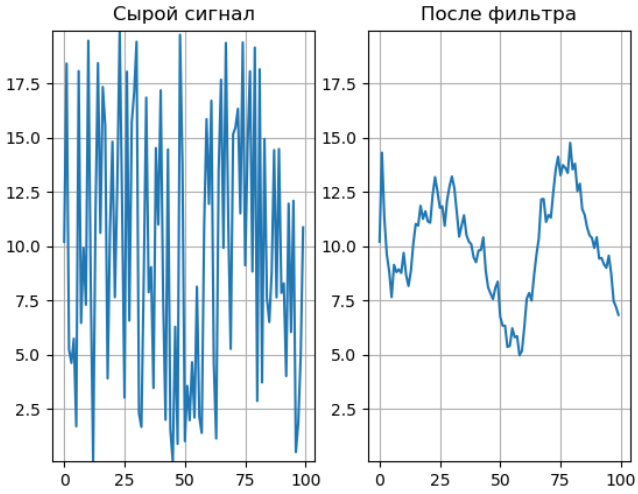
Данные здесь. В каждом файле просто набор значений, полученных в последовательные моменты времени.

Задача - применить к данным фильтр, отрисовать исходные и отфильтрованные данные. Логика фильтра следующая:

1. будем использовать простое скользящее среднее с шагом 10,
2. смотреть будем только назад по времени,
3. стартовые точки сгладим максимально доступным образом.

То есть 0-вое показание остаётся просто собой, 1-ое показание становится средним 0-го и 1-го, 2-ое показание становится средним показаний от 0-го до 2-го и т.д. Показание N становится средним показаний от N-9 до N включительно.

Пример того, как может выглядеть результат:



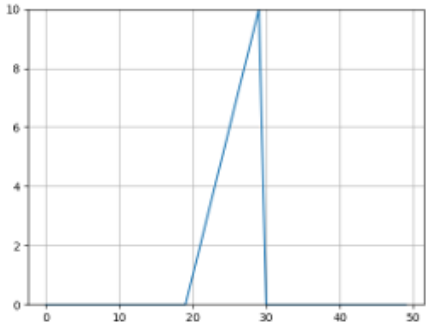
Эпизод 3

Некоторый процесс описывается следующим уравнением:

$$\vec{u}(x)^{n+1} = \vec{u}(x)^n - 0.5 \cdot \mathbf{A} \cdot \vec{u}(x)^n$$

В этом уравнении u(x)^n - функция u в момент времени n. Фактически задана формула, как от момента времени n перейти к моменту времени n+1.

Функция задана поточечно. Её начальный вид можно считать из этого файла. Примерный вид начального состояния:



Матрица A в этой записи имеет вот такой вид:

```
[[ 1.  0.  0. ...  0.  0. -1.]
 [-1.  1.  0. ...  0.  0.  0.]
 [ 0. -1.  1. ...  0.  0.  0.]
 ...
 [ 0.  0.  0. ...  1.  0.  0.]
 [ 0.  0.  0. ... -1.  1.  0.]
 [ 0.  0.  0. ...  0. -1.  1.]]
```

То есть на главной диагонали расположены элементы 1, с циклическим сдвигом влево на один элемент от главной диагонали расположены -1, все остальные элементы 0.

Посчитайте и визуализируйте 255 шагов по времени этого процесса.

Пример того, как может выглядеть результат: