Алгоритм для обработки и построения графиков

Исходные данные представляют собой матрицу размера 1000*14.

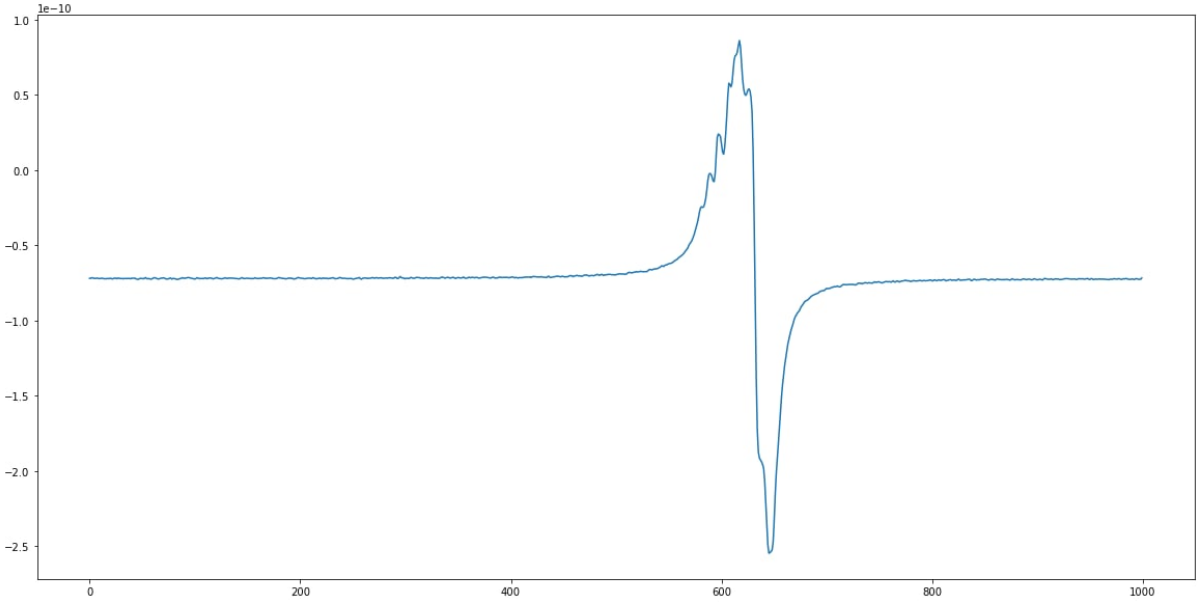
	point_1	point_2	point_3	point_4	point_5	point_6	point_7	point_8	point_9	point_10	point_11	point_12	point_13	point_14
0	3306	3033	2761	2663	2639	2578	2703	2868	3075	3254	3510	3536	3550	3425
1	3324	3044	2868	2734	2650	2632	2679	2892	3071	3215	3421	3555	3504	3436
2	3302	3077	2865	2727	2661	2574	2651	2789	3018	3213	3477	3552	3548	3545
3	3296	3082	2830	2731	2633	2488	2681	2846	3084	3217	3448	3602	3527	3497
4	3301	3078	2848	2770	2600	2511	2664	2884	3062	3252	3466	3572	3510	3475
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	3243	2992	2779	2698	2566	2607	2676	2904	3103	3312	3482	3619	3533	3450
996	3303	3032	2813	2712	2603	2603	2708	2864	3134	3210	3492	3619	3488	3448
997	3244	3012	2869	2690	2610	2466	2693	2929	3068	3237	3582	3603	3500	3460
998	3297	3061	2851	2689	2566	2573	2679	2850	3002	3236	3495	3634	3567	3459
999	3305	3045	2804	2725	2602	2635	2611	2881	3105	3303	3487	3554	3476	3512

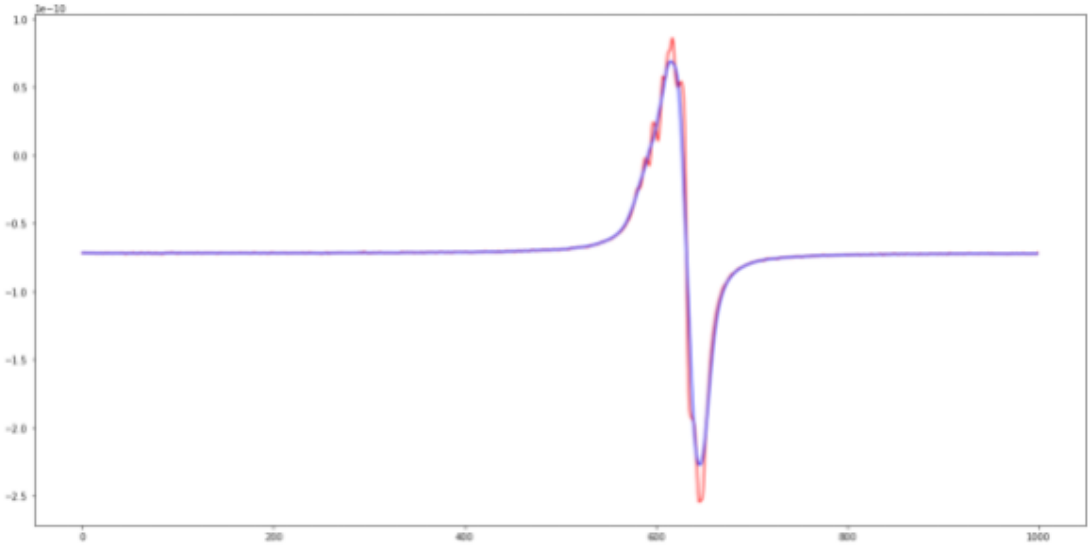
1000 rows × 14 columns

Веса синуса

Так выглядит график обработанный синусом(слева) и с применением фильтра(справа)

```
weights = [np.sin(2* np.pi * i) for i in range(14)]
arr_sin = list(range(0,13))
str_sin = list(range(0,14))
full = list()
for i in range(data.shape[0]):
    for j in range(data.shape[1]):
        str_sin[j] = np.mean((weights[j] * data.iloc[i][j]))
    full.append(sum(str_sin))
```





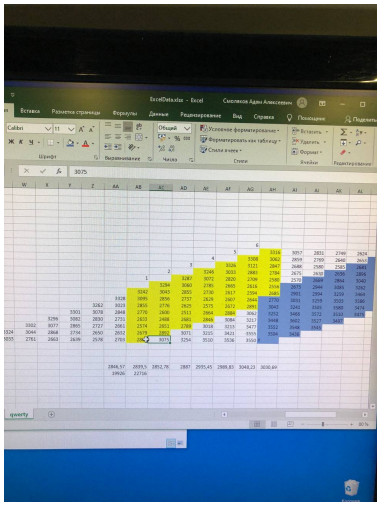
LOWESS (Locally Weighted Scatterplot Smoothing)

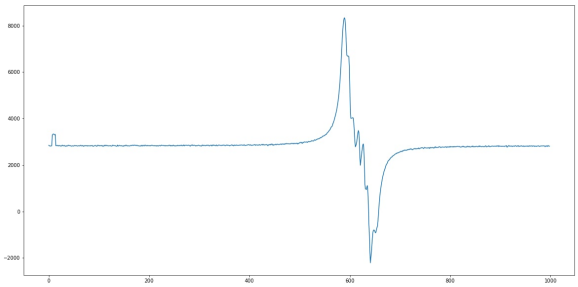
Наложение точек (все точки)

Алгоритм, в котором точки накладываются друг на друга по такой схеме(рисунок слева) выдаёт следующий результат(рисунок слева)

```
arr = data.to_numpy()
dots = [0] * 2000
dots_first = []
dots_last = []

for i in range(993):
    kernel_first = arr[i:7+i,:7]
    b_f = np.asarray(kernel_first)
    b_f = np.fliplr(b_f)
    #print('Antidiagonal first (sum): ', np.trace(b_f))
    #print('Antidiagonal (elements): ', np.diagonal(b_f))

    kernel_last = arr[i:7+i,7:]
    b_l = np.asarray(kernel_last)
    b_l = np.fliplr(b_l)
```

```
    #print('Antidiagonal last (sum): ', np.trace(b_l))
    #print('Antidiagonal (elements): ', np.diagonal(b_l))

    dots_first.append((np.trace(b_f)) / 7)
    dots_last.append((np.trace(b_l)) / 7)

dots[0:7] = dots_first[0:7]
dots[7:14] = dots_last[0:7]
for i in range(2,1986):
    dots[7*i:7*(i+1)] = dots_first[7*(i-1):7*i]
    dots[7*(i+1):7*(i+2)] = dots_last[7*(i-1):7*i]
```

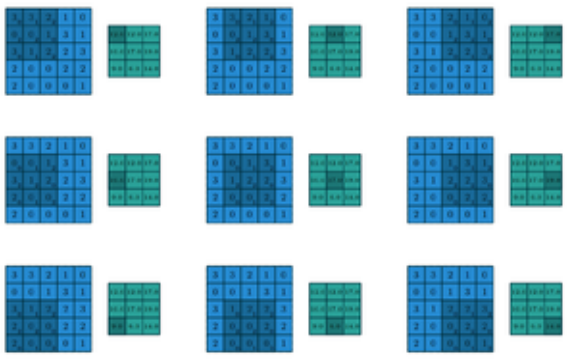




Свертка

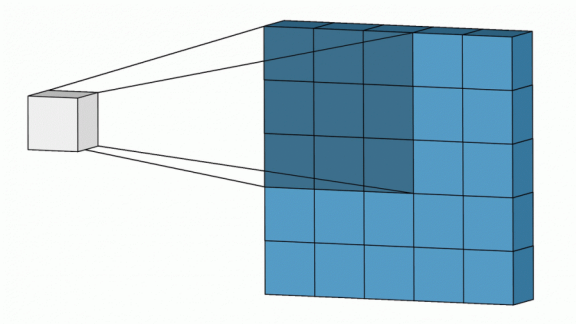
При обработке входных точек с помощью свертки(рисунок слева) и домнажении на веса синуса, получается такой грфик(рисунок срава). В этом методе важным является выбор сверточного ядра.

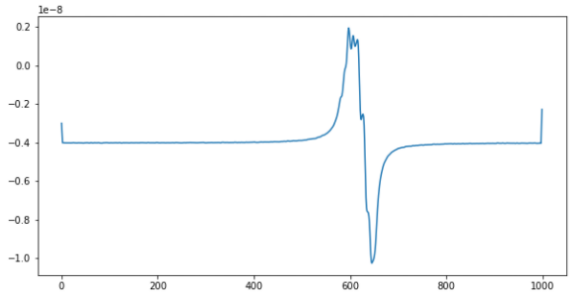
```
a = data
a = a.to_numpy()

k = np.array([[5,3,5],[2,5,3],[8,4,4],[2,3,9]])
from scipy import ndimage
res = ndimage.convolve(a, k, mode='constant', cval=1.0)
```



Операция свертки

Все 14000 точек без обработки

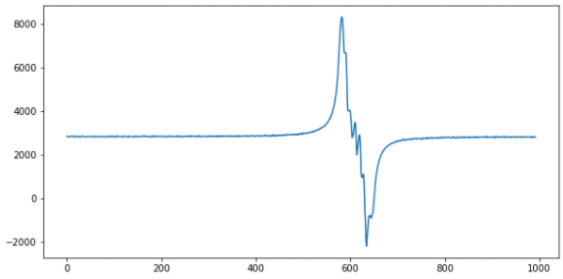




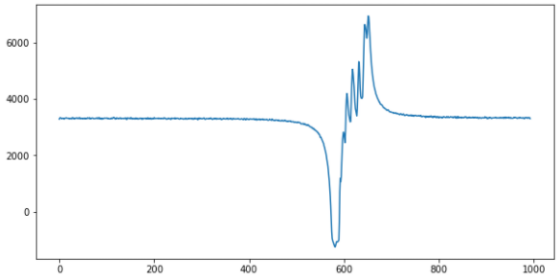
1000 точек, полученные домножением на синус

Наложение точек (Первые 7 и последние 7 точек)

Так как точки в строке - соответсвуют развертке амлитуды модуляции. То первые 7 - соответсуют росту модуляции, а последние 7 - падению.



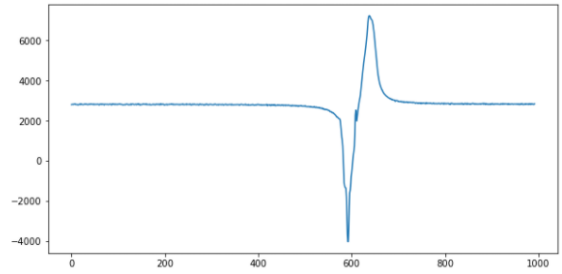
Первые 7 точек



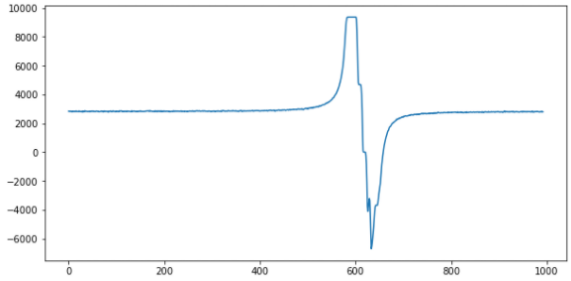
Последние 7 точек

Наложение точек(Произвольное начало среза и количество точек для построения)

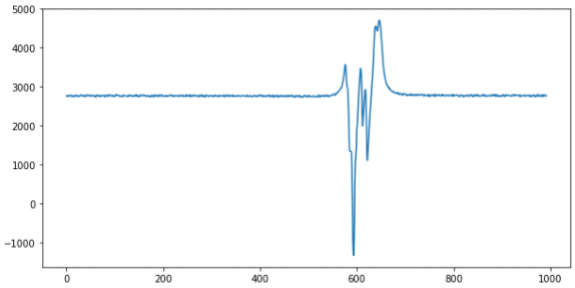
```
a = int(input("Введи стартовую точку среза матрицы"))
b = int(input("Введите размерность ядра. Ядро должно быть квадратной матрицей, поэтому укажите только одно число"))
arr = data.to_numpy()
dots = []

for i in range(993):
    kernel = arr[i:b+i,a-1:(a-1)+b]
    b_f = np.asarray(kernel)
    b_f = np.fliplr(b_f)
    #print('Antidiagonal (sum): ', np.trace(b_f))
    #print('Antidiagonal (elements): ', np.diagonal(b_f))

    dots.append((np.trace(b_f)) / b)
```



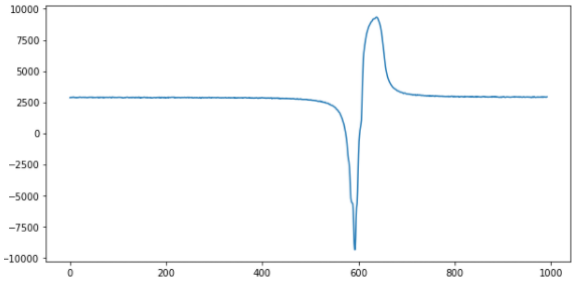
С 4 по 11 точку. Ядро размером 7 на 7



С 2 по 6 точку. Ядро размером 4 на 4



С 3-10 точку. Ядро размера 7 на 7

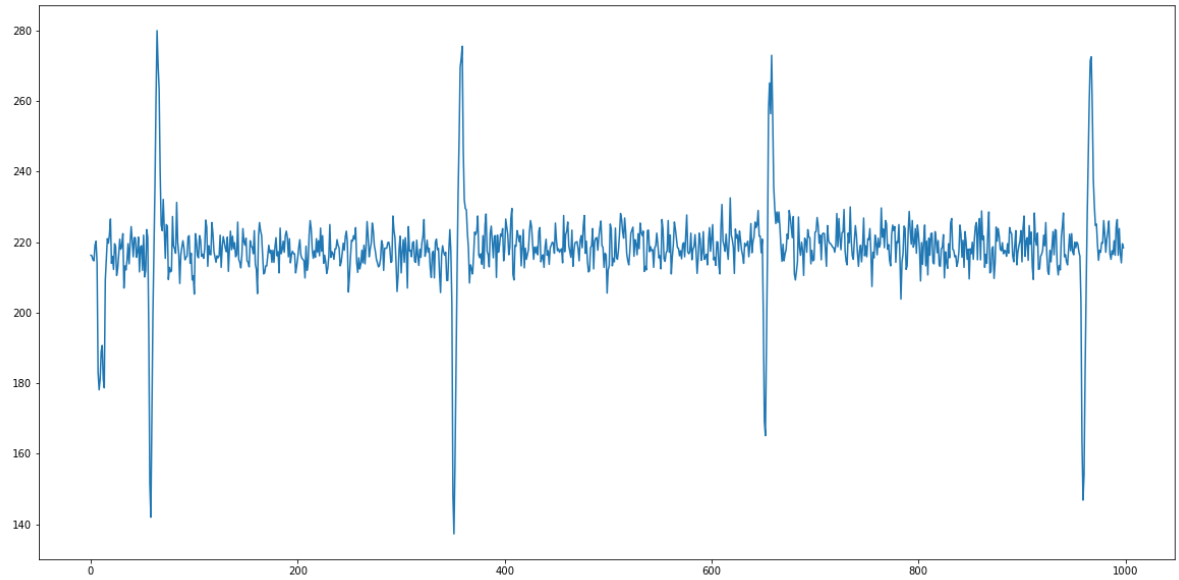


С 6 - 11 точку. Ядро размера 5 на 5

Данные, полученные с помощью EPR_Terminal(Файлы ниже)

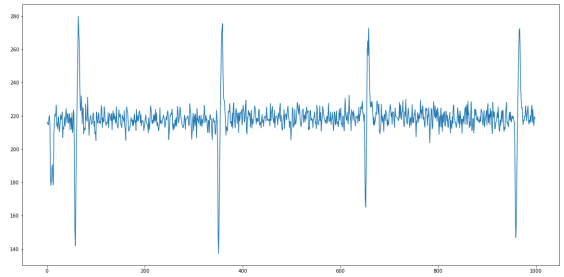
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2288bb46-64d2-4ca8-9d71-d6a10ec8c070/Mn_sw-15_ampl-210.xlsx

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/68f4e21c-b13d-4522-8e44-26c17045a58a/Mn_sw-30_ampl-100.xlsx

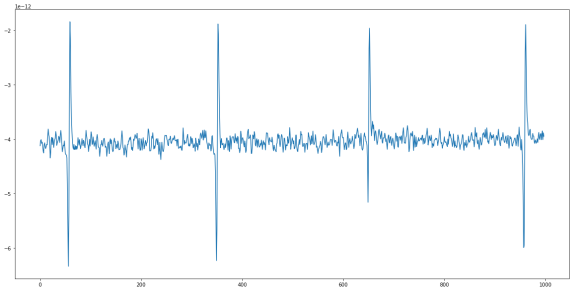


Ширина развертки - 15mT. Амлитуда модуляуции - 210uT, Шаг одинаковый - 30 uT

Ниже приведены 2 графика, построенные по одним и тем же данным. Левый график построен с помощью алгортима наложения весов. Правый - с помощью алгортима умножения на синус



Ширина развертки - 30mT. Амлитуда модуляуции - 100uT. Алгоритм - Наложение точек



Ширина развертки - 30mT. Амлитуда модуляуции - 100uT. Алгоритм - умножение на веса синуса