

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования



**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники
Отделение информационных технологий
Направление подготовки 09.04.04 Программная инженерия

Отчёт по лабораторной работе №1

МЕТОДЫ ФИЛЬТРАЦИИ СИГНАЛОВ

по дисциплине Основы теории управления автономными системами

Выполнил студент гр. 8ПМ4Л

Подпись

Дата

Сокуров Р.Е.
Фамилия И.О.

Проверил доцент ОАР

Подпись

Дата

Хожаев И.В.
Фамилия И.О.

Томск 2024 г.

Цель

Изучить основные методы фильтрации измерительных сигналов.

Задачи

- 1) Изучить принцип работы следующих фильтров:
 - фильтра экспоненциального сглаживания;
 - фильтра скользящего среднего;
 - медианного фильтра;
 - фильтра с ограничением скорости нарастания сигнала;
- 2) Сгенерировать полезный сигнал и добавить к нему равномерно распределенный шум и всплески большой амплитуды;
- 3) Реализовать каждый из ранее перечисленных фильтров любым известным способом, проверить работоспособность фильтров на сгенерированном ранее зашумленном сигнале и изучить влияние настроечного параметра фильтра на качество обработки сигнала;
- 4) Подобрать комбинацию фильтров, отделяющих полезный сигнал от шума обоих типов; сравнить исходный полезный сигнал и отфильтрованный;
- 5) Оформить отчет.

Ход работы

1. Генерация полезного сигнала и добавление к нему равномерно распределённого шума и всплесков большой амплитуды.

В качестве генерируемого сигнала была выбрана функция $y = \sin(x)$. Данная функция была рассчитана на массиве $x \in [-4\pi; 4\pi]$, состоящем из 800 элементов. Затем, с помощью библиотеки «NumPy» языка «Python» был добавлен равномерно распределённый шум, а также высокоамплитудные случайные всплески (Рисунок 1). Код отображён в файле «code.py» в приложении А.

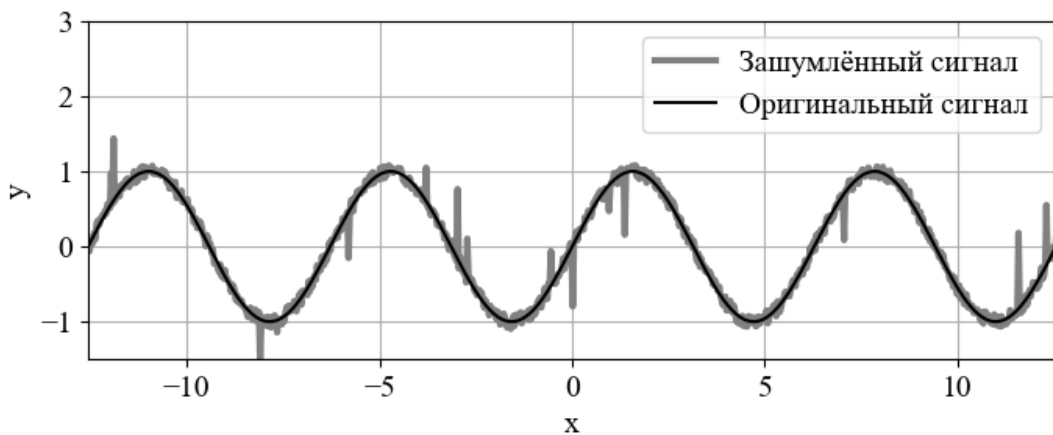


Рисунок 1 – Генерация зашумлённого сигнала

2. Реализация фильтра экспоненциального сглаживания.

Фильтр экспоненциального сглаживания – простейший фильтр низких частот, выходной сигнал которого равен взвешенной сумме последнего измеренного значения сигнала и значения отфильтрованного сигнала на предыдущем шаге:

$$y_n = \alpha \cdot x_n + (1 - \alpha) \cdot y_{n-1},$$

где x – измеренные значения (зашумлённый сигнал), y – значения отфильтрованного сигнала, α – настроечный параметр фильтра.

Данный фильтр был реализован в функции «Exponential_smoothing_filter» в файле «code.py» приложения А.

Данная функция была использована с тремя разными значениями коэффициента $\alpha \in \{0.1, 0.25, 0.5\}$:

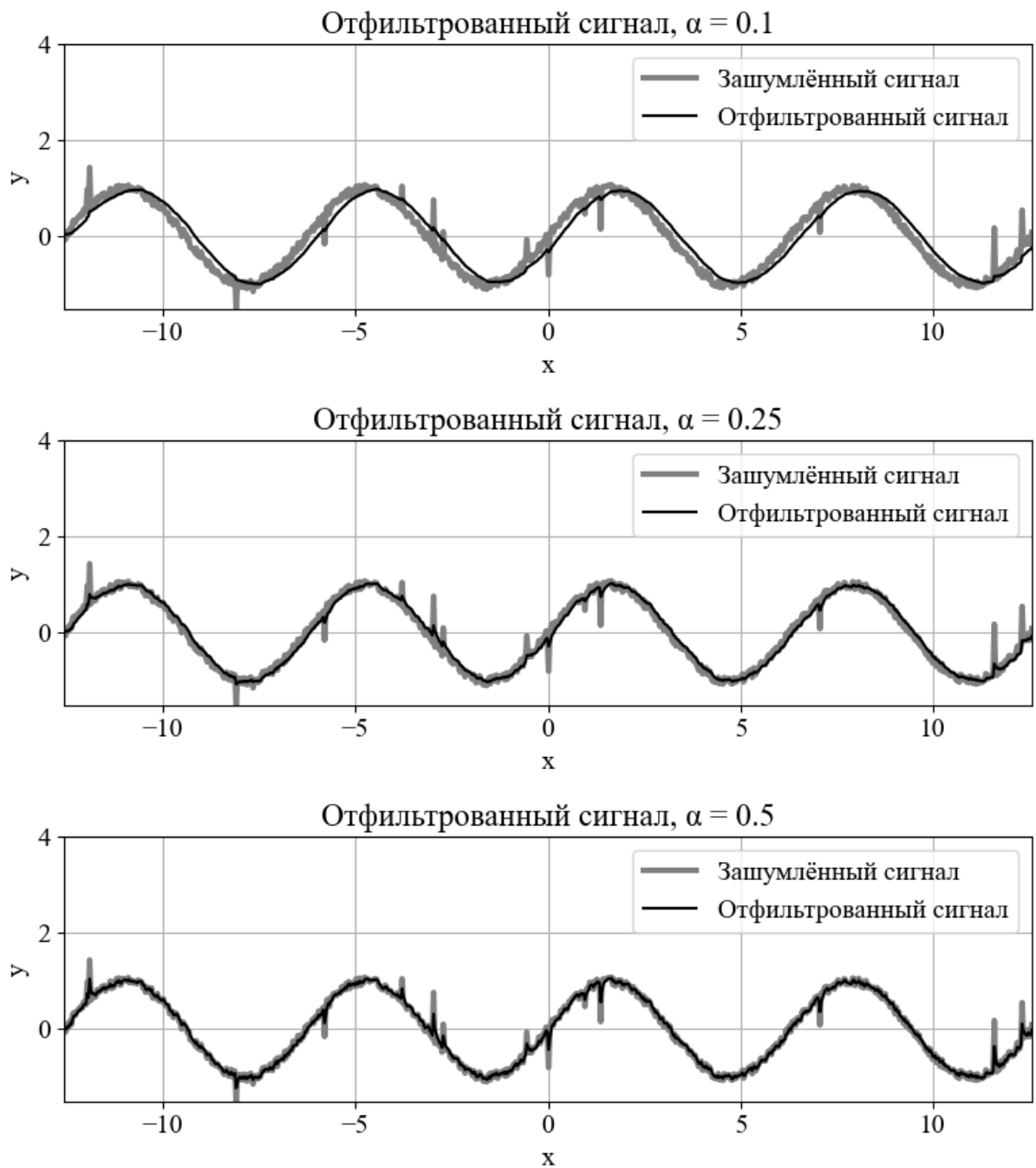


Рисунок 2 – Демонстрация работы фильтра экспоненциального сглаживания

По итогам анализа полученных графиков, оптимальным было принято значение $\alpha = 0.25$, поскольку данное значение значительно сглаживает высокочастотные помехи в сигнале, но ещё не создаёт столь значимое запаздывание, как, например, значение $\alpha = 0.1$.

3. Реализация фильтра скользящего среднего

Фильтр скользящего среднего является фильтром низких частот. Он хранит последние n измерений зашумленного сигнала и выдает на выход их среднее арифметическое.

Данный фильтр был реализован в функции «moving_average_filter» в файле «code.py» приложения А, значение $n \in \{8, 16, 32\}$:

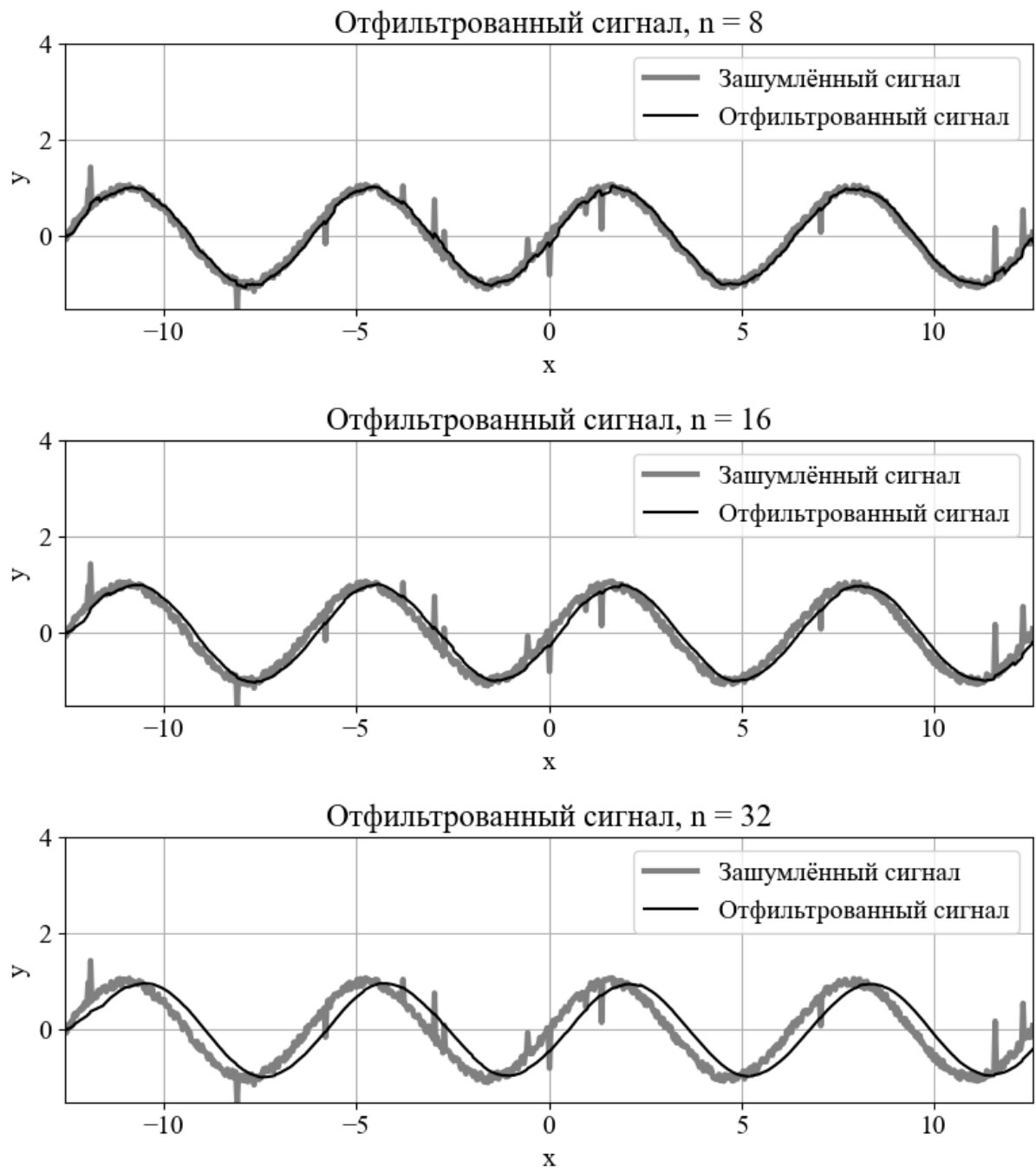


Рисунок 3 – Демонстрация работы фильтра скользящего среднего

Здесь наилучшим значением показал себя размер окна, равный 16, поскольку отставание отфильтрованного сигнала по фазе становится уже слишком большим при $n = 32$, а при $n = 8$ видны участки с резкими переходами.

4. Реализация медианного фильтра

Медианный фильтр является фильтром высоких частот. Его выходной сигнал равен медиане последних n измерений фильтруемого сигнала. Степень сглаживания сигнала также определяется шириной окна n .

Данный фильтр был реализован в функции «median_filter» в файле «code.py» приложения А, значение $n \in \{8, 16, 32\}$:

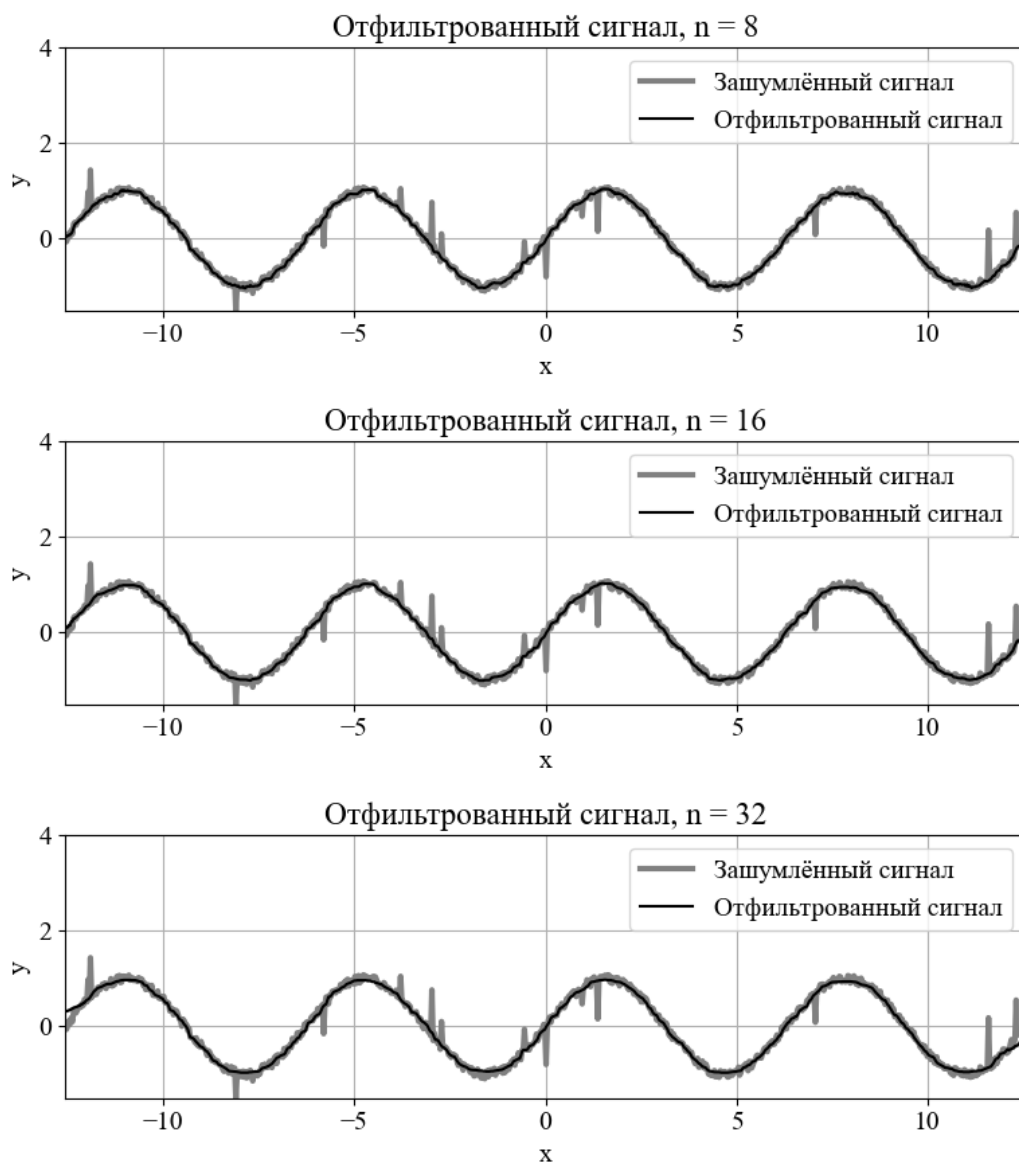


Рисунок 4 – Демонстрация работы медианного фильтра

Здесь, уже при размере окна $n=8$ фильтр уже справился с низкими частотами (высокоамплитудных всплесков в итоговом сигнале нет), однако лучшим всё же оказался размер окна $n=32$, хотя стоит учесть, что фильтр может выдавать некорректные значения в начале (до того, как заполнится размер окна).

5. Реализация фильтра с ограничением скорости нарастания сигнала

Фильтр с ограничением скорости нарастания сигнала выводит на выход значение входного сигнала без изменений, если приращение между текущим и предыдущим его значением меньше заданного порогового значения. В противном случае значение выходного сигнала фильтра не изменяется.

Данный фильтр был реализован в функции «rate_limit_filter» в файле «code.py» приложения А, значение $threshold \in \{0.05, 0.1, 0.2\}$:

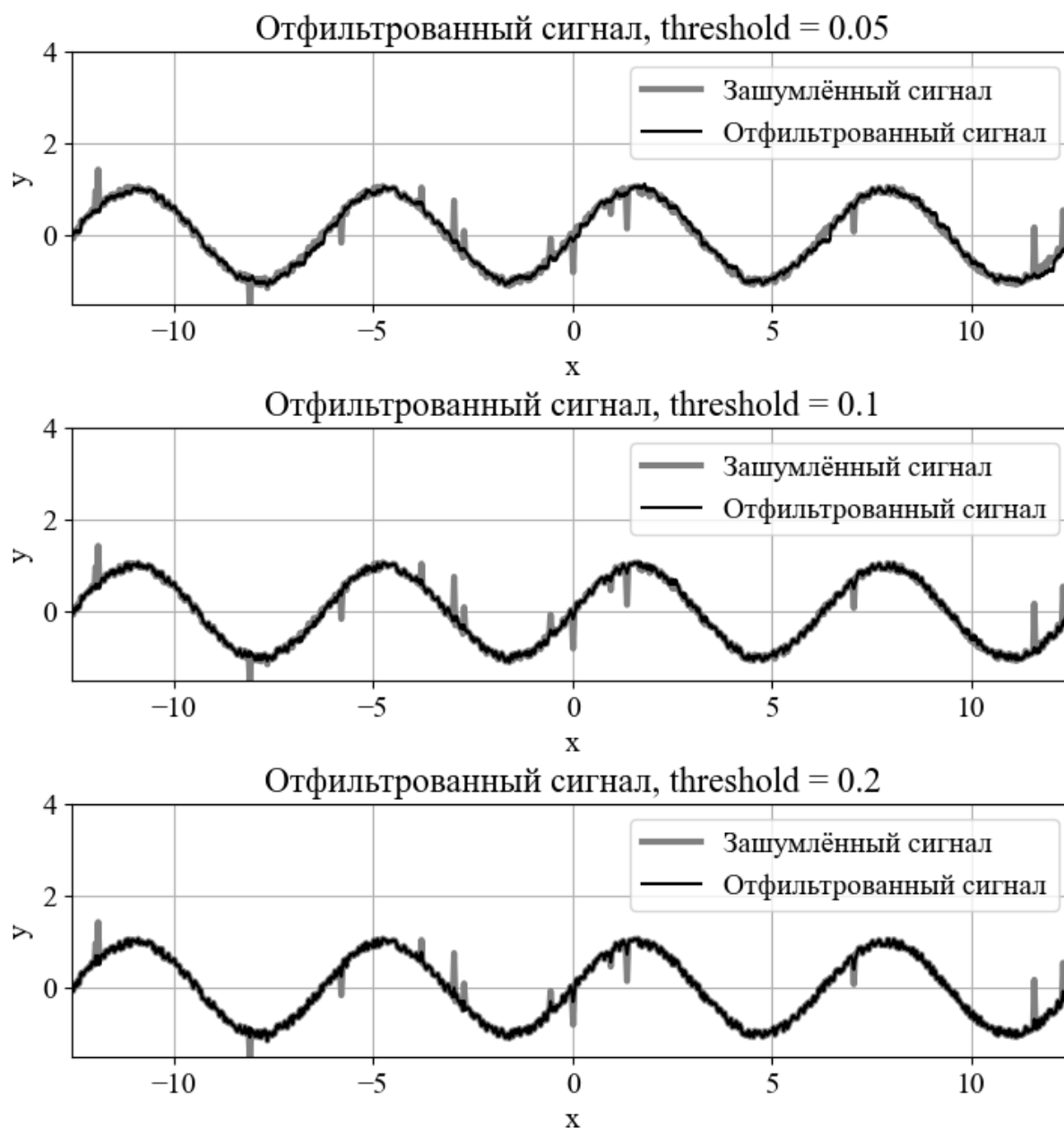


Рисунок 5 – Демонстрация работы фильтра с ограничением скорости нарастания

В данном фильтре, в отличие от всех предыдущих, именно увеличение значения *threshold* ведёт к большей схожести с оригинальным сигналом. Наилучшим значением было принято $threshold = 0.1$, поскольку при $threshold = 0.05$ уже появляются искажение сигнала.

6. Реализация ансамбля (комбинации) фильтров

Для реализации комбинации фильтров, которые лучше всех воспроизводят исходный сигнал из зашумленного, были выбраны медианный фильтр с $n = 32$ и фильтр экспоненциального сглаживания с $\alpha = 0.25$:

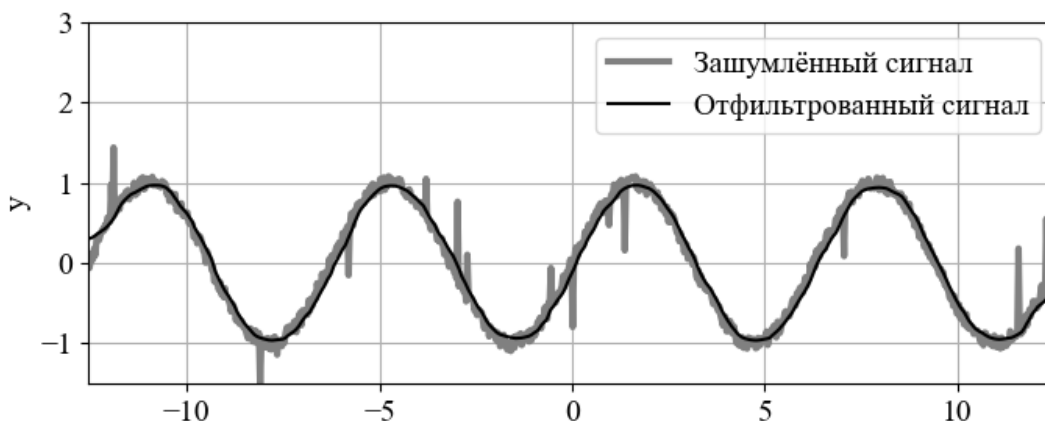


Рисунок 6 – Отфильтрованный двумя фильтрами сигнал

Несмотря на отклонение значений в начале и в конце (из-за особенностей работы медианного фильтра), данный ансамбль отлично справляется со своей задачей фильтрации, при условии, что интересующая нас информация находится не на краях графика.

Заключение

В ходе данной лабораторной работы был составлен полезный сигнал $y = \sin(x)$, из которого, путём искусственного зашумления, получили материал для дальнейшей обработки фильтрами. Были исследованы 4 вида фильтров: фильтр экспоненциального сглаживания, фильтр скользящего среднего, медианный фильтр, фильтр с ограничением скорости нарастания сигнала. Каждый из них рассматривался при различных значениях настраиваемых параметров.

По итогам данных исследований было выбрано два фильтра, комбинация которых использовалась в п.6 работы для максимального приближения зашумлённого сигнала к исходному.

Приложение А
Реализация фильтров

```

1. # Подключение необходимых библиотек
2. import numpy as np
3. import matplotlib.pyplot as plt
4.
5. # Установление зерна случайно генерации
6. # Для возобновляемости экспериментов
7. np.random.seed(52)
8.
9. # Постоянные переменные
10. ALPHAS = [0.1, 0.25, 0.5] # Степень влияния прошлых значений для Exponential_smoothing_filter
11. N = [8, 16, 32] # Используется в скользящем среднем и медианном фильтрах
12. THRESHOLDS = [0.05, 0.1, 0.2] # Порог скорости нарастания сигнала
13.
14. # Настройки шрифта для графика
15. plt.rcParams['font.family'] = 'Times New Roman'
16. plt.rcParams['font.size'] = 14
17.
18. # -----# Создание функции каждого фильтра -----#
19. def exponential_smoothing_filter(data, alpha):
20.     """
21.     Функция реализации фильтра экспоненциального сглаживания
22.
23.     :param a: массив данных, которые нужно отфильтровать
24.     :param alpha: регулировочный коэффициент
25.     :return: отфильтрованный массив
26.     """
27.     filtered_data = data.copy()
28.     for i in range(1, filtered_data.size):
29.         filtered_data[i] = alpha * filtered_data[i] + (1-alpha) * filtered_data[i-1]
30.     return filtered_data
31.
32. def moving_average_filter(data, n):
33.     """
34.     Функция реализации фильтра скользящего среднего
35.
36.     :param data: массив данных, которые нужно отфильтровать
37.     :param n: ширина окна усреднения
38.     :return: отфильтрованный массив
39.     """
40.     # Массив для хранения отфильтрованных значений
41.     filtered_data = []
42.
43.     # Массив для хранения текущего окна измерений
44.     current_values = []
45.
46.     # Пройдем по всем элементам исходного массива
47.     for new_value in data:
48.         # Добавляем текущее новое значение в массив измерений
49.         current_values.append(new_value)
50.
51.         # Если количество элементов меньше n, считаем среднее по всем доступным значениям
52.         if len(current_values) < n:
53.             avg = np.mean(current_values)
54.         else:
55.             # Как только элементов стало n или больше, считаем среднее по последним n значениям
56.             avg = np.mean(current_values[-n:])
57.
58.         # Добавляем отфильтрованное значение в результат
59.         filtered_data.append(avg)
60.
61.     return filtered_data
62.
63. def median_filter(data, window_size):
64.     """
65.     Функция реализации медианного фильтра
66.
67.     :param data: массив данных, которые нужно отфильтровать
68.     :param n: ширина окна расчёта медианы
69.     :return: отфильтрованный массив
70.     """
71.     # Если размер окна меньше 1 или больше длины сигнала, возвращаем исходный сигнал

```

```

72.     if window_size < 1 or window_size > len(data):
73.         return data
74.
75.     # Создаем список для хранения отфильтрованных значений
76.     filtered_data = []
77.
78.     half_window = window_size // 2
79.
80.     # Проходим по каждому элементу сигнала
81.     for i in range(len(data)):
82.         # Выбираем окно вокруг текущего элемента
83.         start_idx = max(0, i - half_window)
84.         end_idx = min(len(data), i + half_window + 1)
85.
86.         # Получаем подмассив, сортируем его и находим медиану
87.         window = sorted(data[start_idx:end_idx])
88.
89.         # Если окно нечетное, выбираем центральный элемент
90.         if window_size % 2 == 1:
91.             median = window[len(window) // 2]
92.         else:
93.             # Если окно четное, берем среднее арифметическое двух центральных элементов
94.             mid1 = window[len(window) // 2 - 1]
95.             mid2 = window[len(window) // 2]
96.             median = (mid1 + mid2) / 2.0
97.
98.         # Добавляем медиану в отфильтрованный сигнал
99.         filtered_data.append(median)
100.
101.     return np.array(filtered_data)
102.
103. def rate_limit_filter(data, threshold):
104.     """
105.     Функция реализации функции ограниченной скорости нарастания сигнала
106.
107.     :param data: массив данных, которые нужно отфильтровать
108.     :param threshold: пороговое значение скорости нарастания
109.     :return: отфильтрованный массив
110.     """
111.
112.     # Инициализация массива выходного сигнала
113.     filtered_data = [data[0]]
114.
115.     # Применение фильтра
116.     for i in range(1, len(data)):
117.         delta = data[i] - data[i-1]
118.         if abs(delta) <= threshold:
119.             filtered_data.append(data[i])
120.         else:
121.             # Ограничение прироста
122.             if delta > 0:
123.                 filtered_data.append(filtered_data[-1] + threshold)
124.             else:
125.                 filtered_data.append(filtered_data[-1] - threshold)
126.
127.     return filtered_data
128.
129. # ----- Создание чистого и зашумленного сигналов для дальнейших экспериментов -----
130. # -----#
131. x = np.linspace(-4*np.pi, 4*np.pi, 800) # Создание массива иксов
132. y = np.sin(x) # Функция
133. y_clean = np.sin(x)
134. noise = np.random.uniform(-0.1, 0.1, 800) # Добавляем равномерно распределенный шум
135. spikes = np.random.choice([0, 1], size=x.shape, p=[0.98, 0.02]) # Случайные всплески (2%
136. # -----#
137. # -----#
138. # -----#
139. # -----#
140. # -----#
141. # -----#
142. # -----#
143. # -----#
144. # -----#
145. # -----#
146. # -----#
147. # -----#
148. # -----#
149. # -----#
150. # -----#
151. # -----#
152. # -----#
153. # -----#
154. # -----#
155. # -----#
156. # -----#
157. # -----#
158. # -----#
159. # -----#
160. # -----#
161. # -----#
162. # -----#
163. # -----#
164. # -----#
165. # -----#
166. # -----#
167. # -----#
168. # -----#
169. # -----#
170. # -----#
171. # -----#
172. # -----#
173. # -----#
174. # -----#
175. # -----#
176. # -----#
177. # -----#
178. # -----#
179. # -----#
180. # -----#
181. # -----#
182. # -----#
183. # -----#
184. # -----#
185. # -----#
186. # -----#
187. # -----#
188. # -----#
189. # -----#
190. # -----#
191. # -----#
192. # -----#
193. # -----#
194. # -----#
195. # -----#
196. # -----#
197. # -----#
198. # -----#
199. # -----#
200. # -----#
201. # -----#
202. # -----#
203. # -----#
204. # -----#
205. # -----#
206. # -----#
207. # -----#
208. # -----#
209. # -----#
210. # -----#
211. # -----#
212. # -----#
213. # -----#
214. # -----#
215. # -----#
216. # -----#
217. # -----#
218. # -----#
219. # -----#
220. # -----#
221. # -----#
222. # -----#
223. # -----#
224. # -----#
225. # -----#
226. # -----#
227. # -----#
228. # -----#
229. # -----#
230. # -----#
231. # -----#
232. # -----#
233. # -----#
234. # -----#
235. # -----#
236. # -----#
237. # -----#
238. # -----#
239. # -----#
240. # -----#
241. # -----#
242. # -----#
243. # -----#
244. # -----#
245. # -----#
246. # -----#
247. # -----#
248. # -----#
249. # -----#
250. # -----#
251. # -----#
252. # -----#
253. # -----#
254. # -----#
255. # -----#
256. # -----#
257. # -----#
258. # -----#
259. # -----#
260. # -----#
261. # -----#
262. # -----#
263. # -----#
264. # -----#
265. # -----#
266. # -----#
267. # -----#
268. # -----#
269. # -----#
270. # -----#
271. # -----#
272. # -----#
273. # -----#
274. # -----#
275. # -----#
276. # -----#
277. # -----#
278. # -----#
279. # -----#
280. # -----#
281. # -----#
282. # -----#
283. # -----#
284. # -----#
285. # -----#
286. # -----#
287. # -----#
288. # -----#
289. # -----#
290. # -----#
291. # -----#
292. # -----#
293. # -----#
294. # -----#
295. # -----#
296. # -----#
297. # -----#
298. # -----#
299. # -----#
300. # -----#
301. # -----#
302. # -----#
303. # -----#
304. # -----#
305. # -----#
306. # -----#
307. # -----#
308. # -----#
309. # -----#
310. # -----#
311. # -----#
312. # -----#
313. # -----#
314. # -----#
315. # -----#
316. # -----#
317. # -----#
318. # -----#
319. # -----#
320. # -----#
321. # -----#
322. # -----#
323. # -----#
324. # -----#
325. # -----#
326. # -----#
327. # -----#
328. # -----#
329. # -----#
330. # -----#
331. # -----#
332. # -----#
333. # -----#
334. # -----#
335. # -----#
336. # -----#
337. # -----#
338. # -----#
339. # -----#
340. # -----#
341. # -----#
342. # -----#
343. # -----#
344. # -----#
345. # -----#
346. # -----#
347. # -----#
348. # -----#
349. # -----#
350. # -----#
351. # -----#
352. # -----#
353. # -----#
354. # -----#
355. # -----#
356. # -----#
357. # -----#
358. # -----#
359. # -----#
360. # -----#
361. # -----#
362. # -----#
363. # -----#
364. # -----#
365. # -----#
366. # -----#
367. # -----#
368. # -----#
369. # -----#
370. # -----#
371. # -----#
372. # -----#
373. # -----#
374. # -----#
375. # -----#
376. # -----#
377. # -----#
378. # -----#
379. # -----#
380. # -----#
381. # -----#
382. # -----#
383. # -----#
384. # -----#
385. # -----#
386. # -----#
387. # -----#
388. # -----#
389. # -----#
390. # -----#
391. # -----#
392. # -----#
393. # -----#
394. # -----#
395. # -----#
396. # -----#
397. # -----#
398. # -----#
399. # -----#
400. # -----#
401. # -----#
402. # -----#
403. # -----#
404. # -----#
405. # -----#
406. # -----#
407. # -----#
408. # -----#
409. # -----#
410. # -----#
411. # -----#
412. # -----#
413. # -----#
414. # -----#
415. # -----#
416. # -----#
417. # -----#
418. # -----#
419. # -----#
420. # -----#
421. # -----#
422. # -----#
423. # -----#
424. # -----#
425. # -----#
426. # -----#
427. # -----#
428. # -----#
429. # -----#
430. # -----#
431. # -----#
432. # -----#
433. # -----#
434. # -----#
435. # -----#
436. # -----#
437. # -----#
438. # -----#
439. # -----#
440. # -----#
441. # -----#
442. # -----#
443. # -----#
444. # -----#
445. # -----#
446. # -----#
447. # -----#
448. # -----#
449. # -----#
450. # -----#
451. # -----#
452. # -----#
453. # -----#
454. # -----#
455. # -----#
456. # -----#
457. # -----#
458. # -----#
459. # -----#
460. # -----#
461. # -----#
462. # -----#
463. # -----#
464. # -----#
465. # -----#
466. # -----#
467. # -----#
468. # -----#
469. # -----#
470. # -----#
471. # -----#
472. # -----#
473. # -----#
474. # -----#
475. # -----#
476. # -----#
477. # -----#
478. # -----#
479. # -----#
480. # -----#
481. # -----#
482. # -----#
483. # -----#
484. # -----#
485. # -----#
486. # -----#
487. # -----#
488. # -----#
489. # -----#
490. # -----#
491. # -----#
492. # -----#
493. # -----#
494. # -----#
495. # -----#
496. # -----#
497. # -----#
498. # -----#
499. # -----#
500. # -----#
501. # -----#
502. # -----#
503. # -----#
504. # -----#
505. # -----#
506. # -----#
507. # -----#
508. # -----#
509. # -----#
510. # -----#
511. # -----#
512. # -----#
513. # -----#
514. # -----#
515. # -----#
516. # -----#
517. # -----#
518. # -----#
519. # -----#
520. # -----#
521. # -----#
522. # -----#
523. # -----#
524. # -----#
525. # -----#
526. # -----#
527. # -----#
528. # -----#
529. # -----#
530. # -----#
531. # -----#
532. # -----#
533. # -----#
534. # -----#
535. # -----#
536. # -----#
537. # -----#
538. # -----#
539. # -----#
540. # -----#
541. # -----#
542. # -----#
543. # -----#
544. # -----#
545. # -----#
546. # -----#
547. # -----#
548. # -----#
549. # -----#
550. # -----#
551. # -----#
552. # -----#
553. # -----#
554. # -----#
555. # -----#
556. # -----#
557. # -----#
558. # -----#
559. # -----#
560. # -----#
561. # -----#
562. # -----#
563. # -----#
564. # -----#
565. # -----#
566. # -----#
567. # -----#
568. # -----#
569. # -----#
570. # -----#
571. # -----#
572. # -----#
573. # -----#
574. # -----#
575. # -----#
576. # -----#
577. # -----#
578. # -----#
579. # -----#
580. # -----#
581. # -----#
582. # -----#
583. # -----#
584. # -----#
585. # -----#
586. # -----#
587. # -----#
588. # -----#
589. # -----#
590. # -----#
591. # -----#
592. # -----#
593. # -----#
594. # -----#
595. # -----#
596. # -----#
597. # -----#
598. # -----#
599. # -----#
600. # -----#
601. # -----#
602. # -----#
603. # -----#
604. # -----#
605. # -----#
606. # -----#
607. # -----#
608. # -----#
609. # -----#
610. # -----#
611. # -----#
612. # -----#
613. # -----#
614. # -----#
615. # -----#
616. # -----#
617. # -----#
618. # -----#
619. # -----#
620. # -----#
621. # -----#
622. # -----#
623. # -----#
624. # -----#
625. # -----#
626. # -----#
627. # -----#
628. # -----#
629. # -----#
630. # -----#
631. # -----#
632. # -----#
633. # -----#
634. # -----#
635. # -----#
636. # -----#
637. # -----#
638. # -----#
639. # -----#
640. # -----#
641. # -----#
642. # -----#
643. # -----#
644. # -----#
645. # -----#
646. # -----#
647. # -----#
648. # -----#
649. # -----#
650. # -----#
651. # -----#
652. # -----#
653. # -----#
654. # -----#
655. # -----#
656. # -----#
657. # -----#
658. # -----#
659. # -----#
660. # -----#
661. # -----#
662. # -----#
663. # -----#
664. # -----#
665. # -----#
666. # -----#
667. # -----#
668. # -----#
669. # -----#
670. # -----#
671. # -----#
672. # -----#
673. # -----#
674. # -----#
675. # -----#
676. # -----#
677. # -----#
678. # -----#
679. # -----#
680. # -----#
681. # -----#
682. # -----#
683. # -----#
684. # -----#
685. # -----#
686. # -----#
687. # -----#
688. # -----#
689. # -----#
690. # -----#
691. # -----#
692. # -----#
693. # -----#
694. # -----#
695. # -----#
696. # -----#
697. # -----#
698. # -----#
699. # -----#
700. # -----#
701. # -----#
702. # -----#
703. # -----#
704. # -----#
705. # -----#
706. # -----#
707. # -----#
708. # -----#
709. # -----#
710. # -----#
711. # -----#
712. # -----#
713. # -----#
714. # -----#
715. # -----#
716. # -----#
717. # -----#
718. # -----#
719. # -----#
720. # -----#
721. # -----#
722. # -----#
723. # -----#
724. # -----#
725. # -----#
726. # -----#
727. # -----#
728. # -----#
729. # -----#
730. # -----#
731. # -----#
732. # -----#
733. # -----#
734. # -----#
735. # -----#
736. # -----#
737. # -----#
738. # -----#
739. # -----#
740. # -----#
741. # -----#
742. # -----#
743. # -----#
744. # -----#
745. # -----#
746. # -----#
747. # -----#
748. # -----#
749. # -----#
750. # -----#
751. # -----#
752. # -----#
753. # -----#
754. # -----#
755. # -----#
756. # -----#
757. # -----#
758. # -----#
759. # -----#
760. # -----#
761. # -----#
762. # -----#
763. # -----#
764. # -----#
765. # -----#
766. # -----#
767. # -----#
768. # -----#
769. # -----#
770. # -----#
771. # -----#
772. # -----#
773. # -----#
774. # -----#
775. # -----#
776. # -----#
777. # -----#
778. # -----#
779. # -----#
780. # -----#
781. # -----#
782. # -----#
783. # -----#
784. # -----#
785. # -----#
786. # -----#
787. # -----#
788. # -----#
789. # -----#
790. # -----#
791. # -----#
792. # -----#
793. # -----#
794. # -----#
795. # -----#
796. # -----#
797. # -----#
798. # -----#
799. # -----#
800. # -----#
801. # -----#
802. # -----#
803. # -----#
804. # -----#
805. # -----#
806. # -----#
807. # -----#
808. # -----#
809. # -----#
810. # -----#
811. # -----#
812. # -----#
813. # -----#
814. # -----#
815. # -----#
816. # -----#
817. # -----#
818. # -----#
819. # -----#
820. # -----#
821. # -----#
822. # -----#
823. # -----#
824. # -----#
825. # -----#
826. # -----#
827. # -----#
828. # -----#
829. # -----#
830. # -----#
831. # -----#
832. # -----#
833. # -----#
834. # -----#
835. # -----#
836. # -----#
837. # -----#
838. # -----#
839. # -----#
840. # -----#
841. # -----#
842. # -----#
843. # -----#
844. # -----#
845. # -----#
846. # -----#
847. # -----#
848. # -----#
849. # -----#
850. # -----#
851. # -----#
852. # -----#
853. # -----#
854. # -----#
855. # -----#
856. # -----#
857. # -----#
858. # -----#
859. # -----#
860. # -----#
861. # -----#
862. # -----#
863. # -----#
864. # -----#
865. # -----#
866. # -----#
867. # -----#
868. # -----#
869. # -----#
870. # -----#
871. # -----#
872. # -----#
873. # -----#
874. # -----#
875. # -----#
876. # -----#
877. # -----#
878. # -----#
879. # -----#
880. # -----#
881. # -----#
882. # -----#
883. # -----#
884. # -----#
885. # -----#
886. # -----#
887. # -----#
888. # -----#
889. # -----#
890. # -----#
891. # -----#
892. # -----#
893. # -----#
894. # -----#
895. # -----#
896. # -----#
897. # -----#
898. # -----#
899. # -----#
900. # -----#
901. # -----#
902. # -----#
903. # -----#
904. # -----#
905. # -----#
906. # -----#
907. # -----#
908. # -----#
909. # -----#
910. # -----#
911. # -----#
912. # -----#
913. # -----#
914. # -----#
915. # -----#
916. # -----#
917. # -----#
918. # -----#
919. # -----#
920. # -----#
921. # -----#
922. # -----#
923. # -----#
924. # -----#
925. # -----#
926. # -----#
927. # -----#
928. # -----#
929. # -----#
930. # -----#
931. # -----#
932. # -----#
933. # -----#
934. # -----#
935. # -----#
936. # -----#
937. # -----#
938. # -----#
939. # -----#
940. # -----#
941. # -----#
942. # -----#
943. # -----#
944. # -----#
945. # -----#
946. # -----#
947. # -----#
948. # -----#
949. # -----#
950. # -----#
951. # -----#
952. # -----#
953. # -----#
954. # -----#
955. # -----#
956. # -----#
957. # -----#
958. # -----#
959. # -----#
960. # -----#
961. # -----#
962. # -----#
963. # -----#
964. # -----#
965. # -----#
966. # -----#
967. # -----#
968. # -----#
969. # -----#
970. # -----#
971. # -----#
972. # -----#
973. # -----#
974. # -----#
975. # -----#
976. # -----#
977. # -----#
978. # -----#
979. # -----#
980. # -----#
981. # -----#
982. # -----#
983. # -----#
984. # -----#
985. # -----#
986. # -----#
987. # -----#
988. # -----#
989. # -----#
990. # -----#
991. # -----#
992. # -----#
993. # -----#
994. # -----#
995. # -----#
996. # -----#
997. # -----#
998. # -----#
999. # -----#
1000. # -----#

```

```

141. # plt.figure(figsize=[8,3])
142. # plt.plot(x,y, color='gray', linewidth=3, label='Зашумлённый сигнал')
143. # plt.plot(x,y_clean, color='black', linewidth=1.5, label='Оригинальный сигнал')
144. # plt.grid()
145. # ax = plt.gca()
146. # ax.set_xlim([-4*np.pi,4*np.pi])
147. # ax.set_ylim([-1.5,3])
148. # plt.xlabel('x')
149. # plt.ylabel('y')
150. # plt.legend()
151.
152. # ----- Вывод на график результата фильтрации (нужный раскоментить) -----
153. # -----#
154. # plt.figure(figsize=[8, 9]) # Размер всей фигуры: 8x9 (3 графика по 8x3)
155.
156. # for idx, alpha in enumerate(ALPHAS):
157. #     y_exp = Exponential_smoothing_filter(y, alpha)
158. #     plt.subplot(len(ALPHAS), 1, idx + 1)
159. #     plt.plot(x, y, color='gray', linewidth=3, label='Зашумлённый сигнал')
160. #     plt.plot(x, y_exp, color='black', linewidth=1.5, label=f'Отфильтрованный сигнал')
161. #     plt.grid()
162. #     plt.xlim([-4 * np.pi, 4 * np.pi])
163. #     plt.ylim([-1.5, 4])
164. #     plt.xlabel('x')
165. #     plt.ylabel('y')
166. #     plt.legend()
167. #     plt.title(f'Отфильтрованный сигнал,  $\alpha = \{\alpha\}$ ')
168.
169. # plt.tight_layout()
170.
171. # ----- Вывод на график результата фильтрации (нужный раскоментить) -----
172. # -----#
173. # plt.figure(figsize=[8, 9]) # Размер всей фигуры: 8x9 (3 графика по 8x3)
174.
175. # for idx, n in enumerate(N):
176. #     y_exp = moving_average_filter(y, n)
177. #     plt.subplot(len(N), 1, idx + 1)
178. #     plt.plot(x, y, color='gray', linewidth=3, label='Зашумлённый сигнал')
179. #     plt.plot(x, y_exp, color='black', linewidth=1.5, label=f'Отфильтрованный сигнал')
180. #     plt.grid()
181. #     plt.xlim([-4 * np.pi, 4 * np.pi])
182. #     plt.ylim([-1.5, 4])
183. #     plt.xlabel('x')
184. #     plt.ylabel('y')
185. #     plt.legend()
186. #     plt.title(f'Отфильтрованный сигнал,  $n = \{n\}$ ')
187.
188. # plt.tight_layout()
189.
190. # ----- Вывод на график результата фильтрации (нужный раскоментить) -----
191. # -----#
192. # plt.figure(figsize=[8, 9]) # Размер всей фигуры: 8x9 (3 графика по 8x3)
193.
194. # for idx, n in enumerate(N):
195. #     y_exp = median_filter(y, n)
196. #     plt.subplot(len(N), 1, idx + 1)
197. #     plt.plot(x, y, color='gray', linewidth=3, label='Зашумлённый сигнал')
198. #     plt.plot(x, y_exp, color='black', linewidth=1.5, label=f'Отфильтрованный сигнал')
199. #     plt.grid()
200. #     plt.xlim([-4 * np.pi, 4 * np.pi])
201. #     plt.ylim([-1.5, 4])
202. #     plt.xlabel('x')
203. #     plt.ylabel('y')
204. #     plt.legend()
205. #     plt.title(f'Отфильтрованный сигнал,  $n = \{n\}$ ')
206.
207. # plt.tight_layout()
208.

```

```

209. # ----- Вывод на график результата фильтрации (нужный раскоментить) -----
-----#
210.
211. # plt.figure(figsize=[8, 9]) # Размер всей фигуры: 8x9 (3 графика по 8x3)
212.
213. # for idx, threshold in enumerate(THRESHOLDS):
214. #     y_exp = rate_limit_filter(y, threshold)
215. #     plt.subplot(len(THRESHOLDS), 1, idx + 1)
216. #     plt.plot(x, y, color='gray', linewidth=3, label='Зашумлённый сигнал')
217. #     plt.plot(x, y_exp, color='black', linewidth=1.5, label=f'Отфильтрованный сигнал')
218. #     plt.grid()
219. #     plt.xlim([-4 * np.pi, 4 * np.pi])
220. #     plt.ylim([-1.5, 4])
221. #     plt.xlabel('x')
222. #     plt.ylabel('y')
223. #     plt.legend()
224. #     plt.title(f'Отфильтрованный сигнал, threshold = {threshold}')
225.
226. # plt.tight_layout()
227.
228. # ----- Вывод на график результата фильтрации (нужный раскоментить) -----
-----#
229.
230. y_med = median_filter(y, 32)
231. y_exp = Exponential_smoothing_filter(y_med, 0.25)
232. plt.figure(figsize=[8,3])
233. plt.plot(x,y, color='gray', linewidth=3, label='Зашумлённый сигнал')
234. plt.plot(x,y_exp, color='black', linewidth=1.5, label='Отфильтрованный сигнал')
235. plt.grid()
236. ax = plt.gca()
237. ax.set_xlim([-4*np.pi,4*np.pi])
238. ax.set_ylim([-1.5,3])
239. plt.xlabel('x')
240. plt.ylabel('y')
241. plt.legend()
242.
243. plt.show()
244.

```