**Министерство образования Российской Федерации**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

Факультет: Информатика и системы управления Кафедра: Информационная безопасность (ИУ8)

**Методы оптимизации Лабораторная работа №5 на тему:**

«Исследование стохастической фильтрации сигналов как

задачи двухкритериальной оптимизации с использованием

методов прямого пассивного поиска»

Вариант 18

**Преподаватель:**

Коннова Н.С.

**Студент**:  
Ожогин М.А.

**Группа:**

ИУ8-34

Москва 2024

# Цель работы

Изучить основные принципы многокритериальной оптимизации в комбинации с методами случайного и прямого пассивного поиска применительно к задаче фильтрации дискретного сигнала методом взвешенного скользящего среднего.

# Постановка задачи

На интервале [xmin, xmax] задан сигнал fk = f(xk), где дискретная последовательность отсчетов xk = xmin + k(xmax-xmin)/ K, k = 0, …, K, K – количество отсчетов. На сигнал наложен дискретный равномерный шум σ = (σ0, …, σK) с нулевым средним значением и амплитудой, равномерно распределенной на интервале [-a, a]: fk~ = fk + σk, σk = rnd(-a, a). В зависимости от варианта работы необходимо осуществить фильтрацию сигнала fk~одним из методов взвешенного скользящего среднего.

Используется модельная задача со следующими исходными данными:

* Амплитуда равномерного шума 2a = 0,5;
* Дискретизация веса свертки λl = l/L, l = 0, ..., L, L = 10;
* Вероятность попадания в окрестность экстремума P = 0,95;
* Интервал неопределенности ε = 0,01;
* Размер скользящего окна: r = 3, r = 5\*.

Метод расчета и метрика для варианта 18:

Среднее геометрическое

Используется метрика Чебышева, поэтому критерий зашумленности ω:

Критерий отличия δ:

Расчет расстояния dist до идеальной точки:

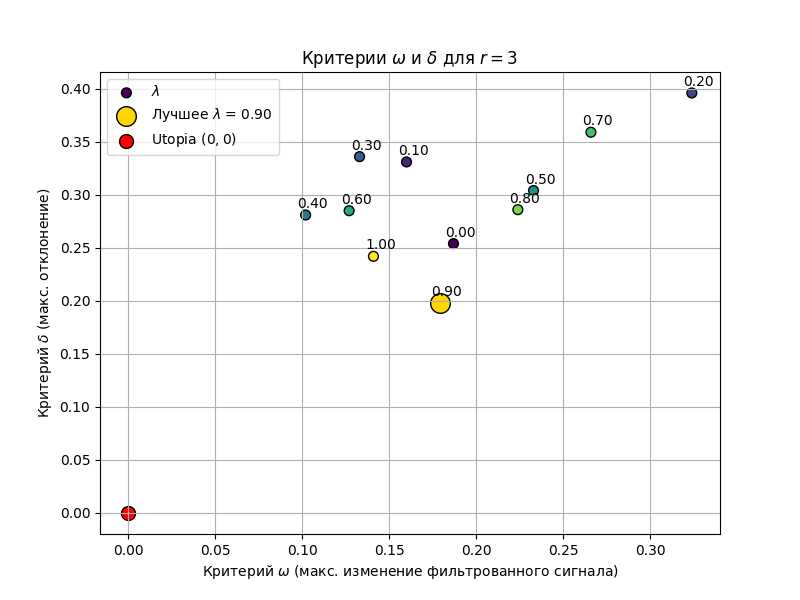
# Ход работы

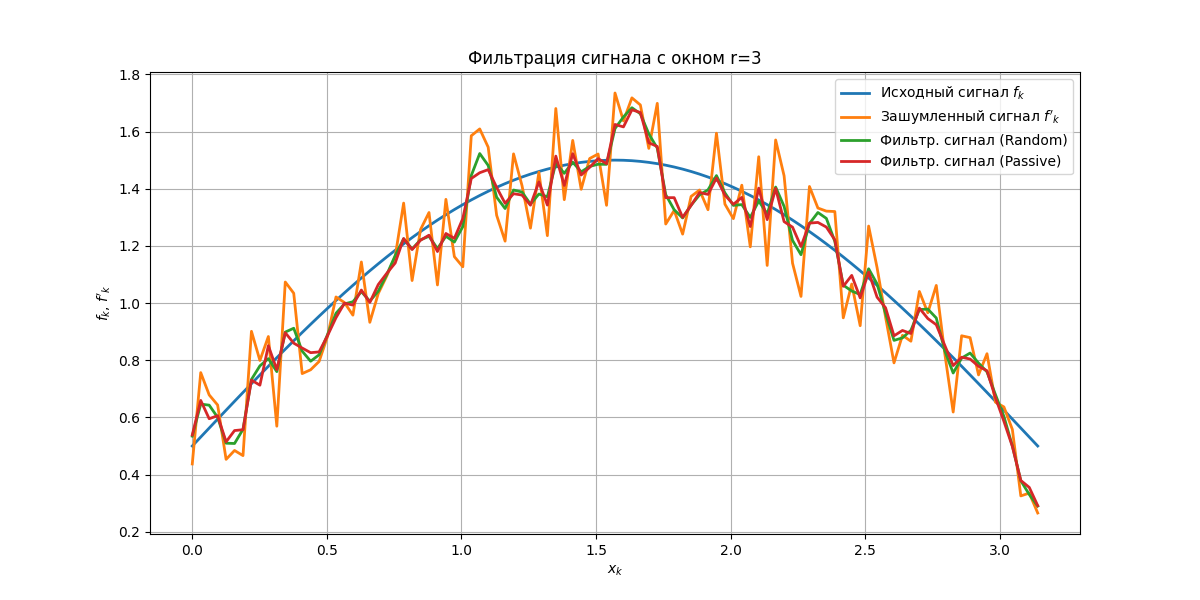
Используем исходные данные, приведенные в разделе “Постановка задачи”.

Результаты численного эксперимента для r = 3 и оптимальное значение веса λ\*, функционала J и критериев ω, δ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| λ | dist | alpha | ω | δ |
| 0.0 | 0.254 | [0.404 0.226 0.37 ] | 0.187 | 0.254 |
| 0.1 | 0.331 | [0.199 0.558 0.243] | 0.160 | 0.331 |
| 0.2 | 0.396 | [0.011 0.899 0.091] | 0.324 | 0.396 |
| 0.3 | 0.336 | [0.144 0.428 0.428] | 0.133 | 0.336 |
| 0.4 | 0.281 | [0.347 0.382 0.271] | 0.102 | 0.281 |
| 0.5 | 0.304 | [0.227 0.254 0.52 ] | 0.233 | 0.304 |
| 0.6 | 0.285 | [0.322 0.339 0.338] | 0.127 | 0.285 |
| 0.7 | 0.359 | [0.142 0.769 0.089] | 0.266 | 0.359 |
| 0.8 | 0.286 | [0.292 0.235 0.472] | 0.224 | 0.286 |
| 0.9 | 0.198 | [0.574 0.192 0.234] | 0.179 | 0.198 |
| 1.0 | 0.242 | [0.514 0.417 0.068] | 0.141 | 0.242 |

Лучшая λ\* = 0.90, dist = 0.198, ω = 0.179, δ = 0.198, J = 0,1809

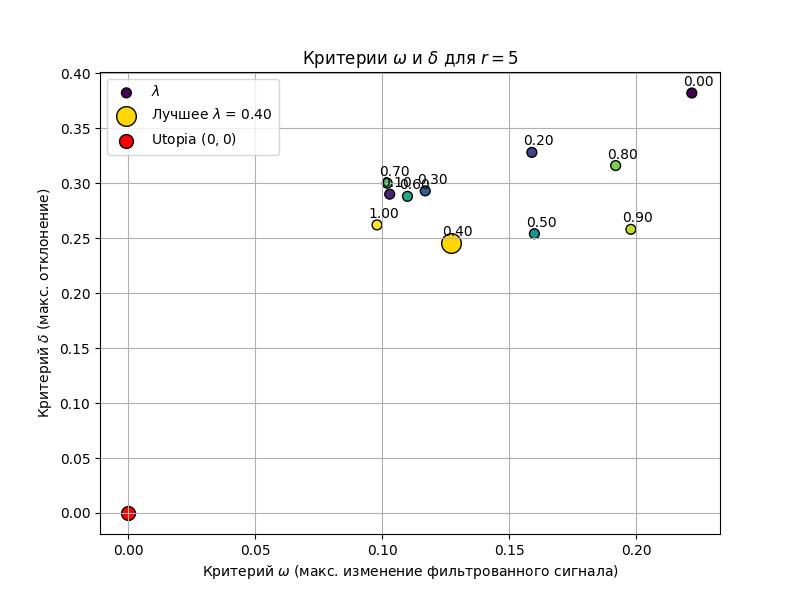


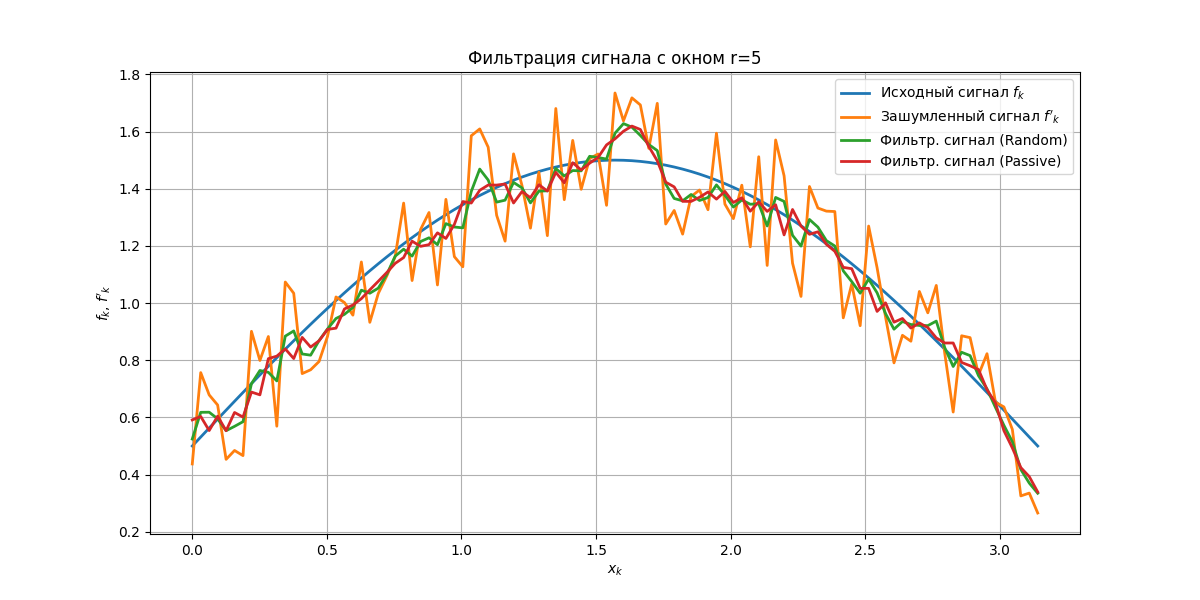


Результаты численного эксперимента для r = 5 и оптимальное значение веса λ\*, функционала J и критериев ω, δ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| λ | dist | alpha | ω | δ |
| 0.0 | 0.382 | [0.115, 0.182, 0.012, 0.612, 0.079] | 0.222 | 0.382 |
| 0.1 | 0.290 | [0.200, 0.095, 0.192, 0.226, 0.288] | 0.103 | 0.290 |
| 0.2 | 0.328 | [0.097, 0.005, 0.064, 0.423, 0.411] | 0.159 | 0.328 |
| 0.3 | 0.293 | [0.202, 0.203, 0.225, 0.059, 0.310] | 0.117 | 0.293 |
| 0.4 | 0.246 | [0.273, 0.250, 0.338, 0.031, 0.108] | 0.127 | 0.246 |
| 0.5 | 0.254 | [0.359, 0.040, 0.293, 0.119, 0.189] | 0.160 | 0.254 |
| 0.6 | 0.288 | [0.130, 0.160, 0.323, 0.133, 0.254] | 0.110 | 0.288 |
| 0.7 | 0.300 | [0.157, 0.219, 0.310, 0.242, 0.071] | 0.102 | 0.300 |
| 0.8 | 0.316 | [0.022, 0.183, 0.414, 0.036, 0.346] | 0.192 | 0.316 |
| 0.9 | 0.258 | [0.317, 0.177, 0.409, 0.057, 0.040] | 0.198 | 0.258 |
| 1.0 | 0.262 | [0.281, 0.191, 0.265, 0.168, 0.095] | 0.098 | 0.262 |

Лучшая λ\* = 0.40, dist = 0.246, ω = 0.127, δ = 0.246, J = 0,1984





Применяя случайный поиск для нахождения вектора α, прямой пассивный поиск — для поиска точки, максимально приближенной к точке утопии, усреднили значения функции и тем самым приблизили зашумленный график к графику изначального сигнала, что отражено на графиках.

# Вывод

В ходе выполнения лабораторной работы были изучены основные принципы многокритериальной оптимизации в комбинации с методами случайного и прямого пассивного поиска.

# Приложение.

Файл “main.py”:

import numpy as np  
import matplotlib.pyplot as plt  
from colorama import Fore, Style  
  
Xmin, Xmax = 0, np.pi  
K = 100  
a = 0.25  
L = 10  
r\_values = [3, 5]  
  
x\_k = np.linspace(Xmin, Xmax, K + 1)  
f\_k = np.sin(x\_k) + 0.5  
  
np.random.seed(42)  
noise = np.random.uniform(-a, a, len(f\_k))  
f\_noisy = f\_k + noise  
  
def geometric\_mean\_filter(signal, r, weights):  
 filtered\_signal = []  
 n = len(signal)  
 for k in range(n):  
 num, den = 1, 0 # Инициализация для геометрического среднего  
 for i in range(-r+1, r):  
 idx = k + i  
 if 0 <= idx < n:  
 w = weights[abs(i)]  
 if signal[idx] > 0: # Избегаем логарифмов от отрицательных значений или нуля  
 num \*= signal[idx] \*\* w  
 den += w  
 filtered\_signal.append(num \*\* (1 / den) if den != 0 else 0) # Геометрическое среднее  
 return np.array(filtered\_signal)  
  
def compute\_chebyshev\_metrics(original, noisy, filtered):  
 omega = np.max(np.abs(filtered[:-1] - filtered[1:])) # max(|f\_k - f\_{k-1}|)  
 delta = np.max([np.abs(filtered[k] - noisy[k]) for k in range(len(original))]) # max(|f\_noisy\_k - f\_original\_k|)  
 dist = max(omega, delta)  
 return omega, delta, dist  
  
  
def random\_search(signal, noisy\_signal, r, n\_iter=100):  
 best\_weights = None  
 best\_dist = float("inf")  
 for \_ in range(n\_iter):  
 weights = np.random.rand(r)  
 weights = weights / np.sum(weights) # Нормализация  
 filtered\_signal = geometric\_mean\_filter(noisy\_signal, r, weights)  
 omega, delta, dist = compute\_chebyshev\_metrics(signal, noisy\_signal, filtered\_signal)  
 if dist < best\_dist:  
 best\_dist = dist  
 best\_weights = weights  
 return best\_weights, best\_dist  
  
def passive\_search\_criteria(signal, noisy\_signal, r, L=10):  
 lambdas = [l / L for l in range(L + 1)]  
 omega\_values = []  
 delta\_values = []  
 dist\_values = []  
 weights\_values = []  
  
 for lamb in lambdas:  
 weights = np.random.rand(r)  
 weights = weights / np.sum(weights) # Нормализация  
 filtered\_signal = geometric\_mean\_filter(noisy\_signal, r, weights)  
 omega, delta, dist = compute\_chebyshev\_metrics(signal, noisy\_signal, filtered\_signal)  
 omega\_values.append(round(omega, 3))  
 delta\_values.append(round(delta, 3))  
 dist\_values.append(round(dist, 3))  
 weights\_values.append(np.round(weights, 3))  
  
 return lambdas, omega\_values, delta\_values, dist\_values, weights\_values  
  
  
for r in r\_values:  
 best\_weights, best\_dist\_random = random\_search(f\_k, f\_noisy, r)  
 filtered\_signal\_random = geometric\_mean\_filter(f\_noisy, r, best\_weights)  
  
 lambdas, omega\_values, delta\_values, dist\_values, weights\_values = passive\_search\_criteria(f\_k, f\_noisy, r)  
  
 best\_dist = min(dist\_values)  
 best\_lambda\_index = dist\_values.index(best\_dist)  
 best\_lambda = lambdas[best\_lambda\_index]  
 weights\_passive = weights\_values[best\_lambda\_index]  
 filtered\_signal\_passive = geometric\_mean\_filter(f\_noisy, r, weights\_passive)  
  
 plt.figure(figsize=(12, 6))  
 plt.plot(x\_k, f\_k, label="Исходный сигнал $f\_k$", linewidth=2)  
 plt.plot(x\_k, f\_noisy, label="Зашумленный сигнал $f'\_k$", linewidth=2)  
 plt.plot(x\_k, filtered\_signal\_random, label=f"Фильтр. сигнал (Random)", linewidth=2)  
 plt.plot(x\_k, filtered\_signal\_passive, label=f"Фильтр. сигнал (Passive)", linewidth=2)  
 plt.title(f"Фильтрация сигнала с окном r={r}")  
 plt.xlabel("$x\_k$")  
 plt.ylabel("$f\_k$, $f'\_k$")  
 plt.legend()  
 plt.grid(True)  
 plt.show()  
  
 plt.figure(figsize=(8, 6))  
 plt.scatter(omega\_values, delta\_values, c=lambdas, cmap='viridis', s=50, edgecolor='k', label="$\\lambda$")  
 for i, lamb in enumerate(lambdas):  
 plt.annotate(f"{lamb:.2f}", (omega\_values[i], delta\_values[i]), textcoords="offset points", xytext=(5, 5),  
 ha='center')  
  
 plt.scatter(omega\_values[best\_lambda\_index], delta\_values[best\_lambda\_index],  
 c='gold', s=200, edgecolor='black', label=f"Лучшее $\\lambda$ = {best\_lambda:.2f}")  
 plt.scatter(0, 0, c='red', s=100, edgecolor='k', label="Utopia $(0, 0)$")  
 plt.title(f"Критерии $\\omega$ и $\\delta$ для $r={r}$")  
 plt.xlabel("Критерий $\\omega$ (макс. изменение фильтрованного сигнала)")  
 plt.ylabel("Критерий $\\delta$ (макс. отклонение)")  
 plt.legend()  
 plt.grid(True)  
 plt.show()  
  
 print(f"Окно r={r}:")  
 max\_weights\_width = max(len(str(weight)) for weight in weights\_values)  
 header = f" λ | dist | {'weights'.ljust(max\_weights\_width)} | ω | δ "  
 print(header)  
 print("-" \* len(header))  
 for i in range(11):  
 if i == best\_lambda\_index:  
 print(  
 Fore.GREEN +  
 f"{lambdas[i]:.2f} | {dist\_values[i]:.3f} | {str(weights\_values[i]).ljust(max\_weights\_width)} | {omega\_values[i]:.3f} | {delta\_values[i]:.3f}"  
 + Style.RESET\_ALL  
 )  
 else:  
 print(  
 f"{lambdas[i]:.2f} | {dist\_values[i]:.3f} | {str(weights\_values[i]).ljust(max\_weights\_width)} | {omega\_values[i]:.3f} | {delta\_values[i]:.3f}"  
 )  
 print(  
 f"Лучший λ\* = {best\_lambda:.2f}, dist = {best\_dist:.3f}, ω = {omega\_values[best\_lambda\_index]:.3f}, δ = {delta\_values[best\_lambda\_index]:.3f}"  
 )  
 print()