МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра *Радиоприемных и радиопередающих устройств*
(полное название кафедры)

Утверждаю

Зав. кафедрой *РПиРПУ*

Киселев А.В.
(подпись, инициалы, фамилия)

«___» _____ 2017 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Чемасова Дмитрия Алексеевича
(фамилия, имя, отчество студента – автора работы)

Исследование модифицированного медианного фильтра
(тема работы)

Факультет радиотехники и электроники
(полное название факультета)

Направление подготовки *11.03.02 Инфокоммуникационные технологии и системы связи*
(код и наименование направления подготовки бакалавра)

Руководитель от НГТУ

Савиных И.С
(фамилия, имя, отчество)

к.т.н
(ученая степень, ученое звание)

_____
(подпись, дата)

Автор выпускной квалификационной работы

Чемасов Д.А
(фамилия, имя, отчество)

РЭФ РТВ14-31
(факультет, группа)

_____
(подпись, дата)

Новосибирск 2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра _____*Радиоприемных и радиопередающих устройств*_____
(полное название кафедры)

УТВЕРЖДАЮ

Зав. кафедрой _____
(фамилия, имя, отчество)

_____
(подпись, дата)

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

студенту _____*Чемасову Дмитрию Алексеевичу*_____
(фамилия, имя, отчество)

Направление подготовки _*11.03.02 Инфокоммуникационные технологии и системы связи*_____
(код и наименование направления подготовки бакалавра)

_____*Факультет радиотехники и электроники*_____
(полное название факультета)

Тема _____*Исследование модифицированного медианного фильтра*_____
(полное название темы выпускной квалификационной работы бакалавра)

Исходные данные (или цель работы) _____
1) Рассмотрение работы медианного фильтра
2) Исследование модифицированного медианного фильтра с прямоугольной взвешивающей функцией при формировании отсчетов

Структурные части работы *Введение. Обзорный раздел. Теоретический раздел. Исследовательский раздел. Экспериментальный раздел. Организационно-экономический раздел. Раздел охраны труда. Заключение. Список литературы*

Задание согласовано и принято к исполнению.

Руководитель от НГТУ	**Студент**
Савиных И.С	*Чемасов Д.А*
(фамилия, имя, отчество)	(фамилия, имя, отчество)
к.т.н	*РЭФ РТВ14-31*
(ученая степень, ученое звание)	(факультет, группа)
(подпись, дата)	(подпись, дата)

Тема утверждена приказом по НГТУ № _____ от «____» _____ 2017 г.

ВКР сдана в ГЭК № _____, тема сверена с данными приказа

_____
(подпись секретаря государственной экзаменационной комиссии по защите ВКР, дата)

_____
(фамилия, имя, отчество секретаря государственной экзаменационной комиссии по защите ВКР)

Аннотация

Тема: "Исследование модифицированного медианного фильтра"

В данной выпускной квалификационной работе рассмотрена работа некоторых нелинейных фильтров, построенных на основе ранговой статистики: медианного фильтра и модифицированного медианного фильтра. Построены зависимости коэффициента нелинейных искажений и коэффициента передачи данных фильтров от частоты входного сигнала. Исследована эффективность подавления гауссовского и импульсного шумов, а также их смеси, для мощностей шума на входе, обеспечивающих отношение сигнал/шум равное 10 и 2. Проведено сравнение эффективности нелинейных фильтров и линейного согласованного фильтра.

Ключевые слова: медианный фильтр, фильтр усеченного среднего, L-фильтр, согласованный фильтр, апертура фильтра, взвешивающая функция, дискретное преобразование Фурье, свертка, коэффициент передачи, коэффициент нелинейных искажений, отношение сигнал/шум.

Annotation

Subject: "Study of a modified median filter"

In this graduation qualification work, the work of some nonlinear filters based on rank statistics is considered: a median filter and a modified median filter. Dependences of the coefficient of nonlinear distortion and the coefficient of data transfer on the frequency of the input signal are constructed. The efficiency of suppression of Gaussian and pulsed noise, as well as their mixtures, for input noise power providing a signal-to-noise ratio equal to 10 and 2 is investigated. The efficiency of nonlinear filters and a linear matched filter is compared.

Keywords: median filter, trimmed mean filter, L-filter, matched filter, filter aperture, window function, discrete Fourier transform, convolution, transmission coefficient, nonlinear distortion coefficient, signal-to-noise ratio.

Содержание

Введение

Фильтрация – это процесс преобразования сигналов, заключающийся в направленном изменении частотного спектра. В результате данного преобразования происходит усиление или ослабление частотных составляющих в определенном диапазоне. Фильтрация широко используется для улучшения качества сигнала (уменьшения уровня шумов и помех) и устранения его искажений, выделения сигналов при частотном разделении каналов, ограничения полосы частот, занимаемой сигналами и т.д. [1, с.355]

Фильтры используются практически во всех электронных устройствах. Они находят свое применение в различных областях, таких как телекоммуникации, радиолокация, обработка сигналов и изображений, дистанционное зондирование и многих других.

Теория фильтров начала разрабатываться еще в 19 веке. Одним из ранних примеров практического применения электромеханических фильтров является гармонический телеграф, изобретенный в 70-е годы 19 века, в котором фильтры использовались для частотного разделения каналов. Однако применение фильтров для разделения сигналов начало широко использоваться лишь с 20-х годов 20 века, когда ведущие ученые признали существование спектра у сигналов. Фильтрация сигналов для повышения чувствительности приемных устройств стала применяться с начала 40-х годов 20 века, когда активно начала развиваться теория, позволяющая синтезировать оптимальные фильтры и оценивать возможности увеличения чувствительности приемников.

Цифровые фильтры появились с началом применения цифровых вычислительных машин в конце 40-х годов 20 века. Цифровой фильтр – это математический алгоритм, реализованный на аппаратном и/или программном уровне, который с заданной целью действует на входной и генерирует выходной цифровой сигнал [1, с.355]. Все фильтры подразделяются на линейные и нелинейные. Линейные фильтры играют существенную роль в теории и практике электротехники, но, к сожалению, не все практические проблемы поддаются линейным решениям. Так, например, локализованные «пики» или «импульсный шум» не могут быть эффективно удалены из других сигналов с помощью линейных фильтров. [2, с.10]

Классическим примером нелинейного фильтра является стандартный медианный фильтр. Этот фильтр оказывается чрезвычайно эффективным при удалении импульсных «пиков» из сигналов, однако во многих случаях он может вносить серьезные искажения. Стремление сохранить эффективность подавления импульсного шума вкупе с желанием

уменьшить уровень вносимых искажений привели к разработке различных модификаций медианного фильтра.

Цифровая обработка сигналов является одной из наиболее быстро развивающихся областей техники. В настоящее время цифровые фильтры применяются практически везде, где требуется обработка сигналов, например, в спектральном анализе, обработке изображений, видео, речи и звука и многих других приложениях [3, с.15]. Было разработано множество видов и модификаций цифровых фильтров, реализующих различные алгоритмы, и в этой области продолжаются активные исследования. Дело в том, что каждый фильтр разрабатывается для определенных целей. Наибольшую эффективность он имеет только в конкретных ситуациях при обработке определенных сигналов. В иных случаях его использование может быть неэффективным и даже может приводить к высокому уровню искажений, потере полезной информации. Например, стандартный медианный фильтр, отлично справляющийся с импульсными шумами, с нормально распределенным шумом справляется намного хуже согласованного фильтра, а при обработке синусоидальных сигналов "уплощает" вершины.

В данной работе будет исследована одна из модификаций медианного фильтра, призванная уменьшить уровень вносимых искажений, а также увеличить эффективность подавления гауссовского шума, сохранив избирательность к импульсному шуму.

1. Обзорно-теоретический раздел

Медианные фильтры

Один из видов нелинейных фильтров – фильтры, основанные на локальных L-оценках, т.е. на ранговой статистике. Их основным преимуществом является простота и скорость вычислений. Фильтры, основанные на ранговой статистике, обычно показывают хорошие результаты при обработке сигналов с аддитивным белым гауссовским и импульсным аддитивным шумами. Они способны хорошо сохранять резкость краев, а также могут быть адаптивными. Таким образом, они подходят для использования в различных приложениях, в которых классические линейные фильтры неэффективны. Наиболее известным и наиболее часто используемым фильтром, основанным на ранговой статистике, является медианный фильтр [4, с.63].

Стандартный медианный фильтр – это особый случай симметричного движущегося окна. Этот фильтр был представлен Джоном Тьюки на конференции 1974 года. Позже медианный фильтр и его модификации нашли многочисленные применения в цифровой обработке и анализе изображений, в цифровом телевидении, в обработке и кодировании речи и многих других приложениях. Чаще всего медианные фильтры используются при обработке изображений для удаления шума, например, типа "соль и перец" (случайные белые и черные пиксели). При обработке звукового сигнала медианные фильтры также широко используются для понижения уровня шума, например, для удаления шума старых виниловых пластинок.

Медианная фильтрация реализуется в виде процедуры обработки отсчетов в скользящем окне, которое охватывает определенное число отсчетов сигнала. Значения входного сигнала, попадающие в окно фильтра, сортируются в порядке возрастания (или убывания): [2, с.105]

$$\{x_{i-K},...,x_i,...,x_{i+K}\} \to \left\{x_{(-K)} \le ... \le x_{(0)} \le ... \le x_{(K)}\right\}$$

Таким образом $x_{(-K)}$ является наименьшим значением из выборки, $x_{(-K+1)}$ – второе по величине и т.д, а $x_{(0)}$ – среднее значение последовательности является медианой и, соответственно, выходным значением медианного фильтра:

$$y_i = median\{x_{i-K},...,x_i,...,x_{i+K}\} = x_{(0)}$$

Обычно на практике выбирают ширину окна медианного фильтра так, чтобы в него попадало нечетное число отсчетов. Если же в окно медианного фильтра попадает четное число отсчетов, то медиану обычно определяют как среднее арифметическое двух средних отсчетов упорядоченной последовательности.

Единственным изменяемым параметром данного фильтра является ширина окна, с увеличением которой влияние фильтра на последовательность отсчетов возрастает. Медианный фильтр имеет простую структуру, которая позволяет реализовывать его как аппаратными, так и программными средствами. [2, с.105]

Медианные фильтры достаточно часто применяются на практике как средство предварительной обработки цифровых данных. Их полезной особенностью является явно выраженная избирательность по отношению к элементам массива, представляющим собой немонотонную составляющую последовательности чисел в пределах апертуры фильтра, и резко выделяющихся на фоне соседних отсчетов. В то же время на монотонную составляющую последовательности медианный фильтр не действует, оставляя её без изменений. Благодаря этой особенности, медианные фильтры при оптимально выбранной апертуре могут сохранять без искажений резкие границы объектов, эффективно устраняя аномальные значения, уменьшая выбросы и сглаживая импульсные помехи.

Во многих случаях применение медианного фильтра оказывается более эффективным по сравнению с линейными фильтрами, поскольку линейная обработка является оптимальной при равномерном или гауссовом распределении помех, что в реальных сигналах может быть не так. В случаях, когда перепады значений сигналов велики по сравнению с дисперсией аддитивного белого шума, медианный фильтр дает меньшее значение среднеквадратической ошибки по сравнению с оптимальными линейными фильтрами. Однако медианный фильтр может также вносить значительные искажения, поэтому его практическая полезность сильно зависит от области применения.

Проблема джиттера края

При отсутствии импульсных помех (то есть одиночных шумовых выбросов большой амплитуды) медианный фильтр демонстрирует превосходное сохранение краев, чего не могут обеспечить линейные фильтры. Тем не менее, при наличии импульсных шумовых выбросов вблизи края у медианного фильтра проявляется джиттер, т.е. край в выходном сигнале смещается относительно его истинного положения [2, с.106].

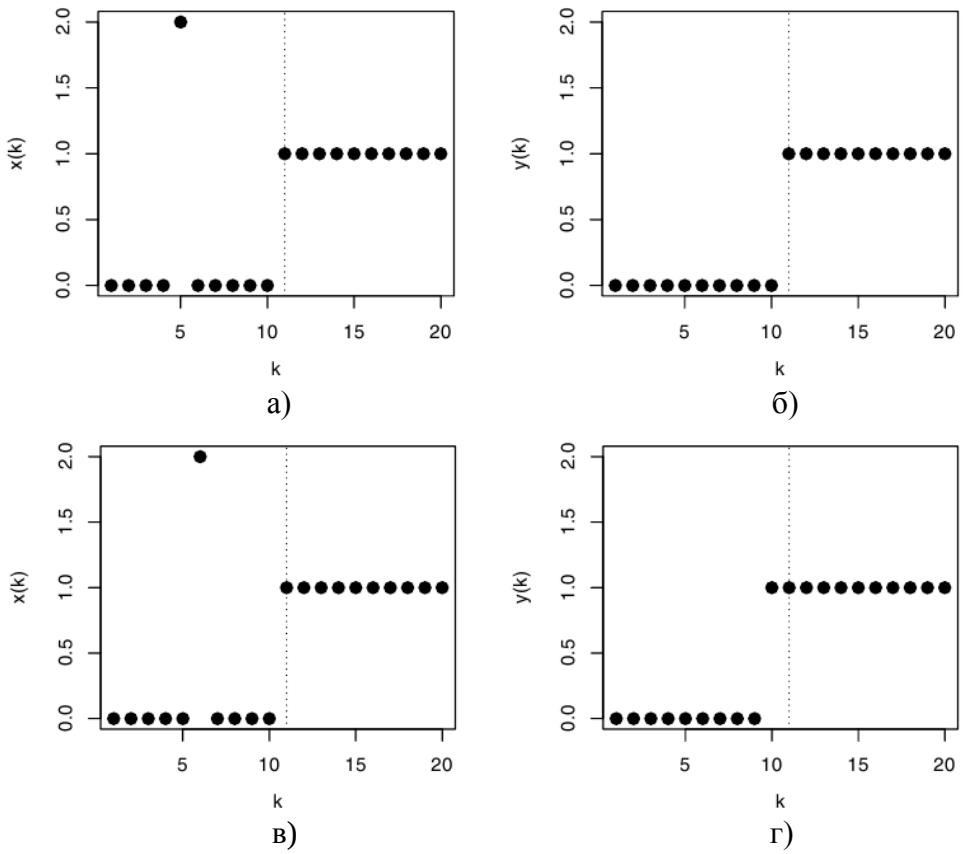


Рисунок 1. Входная последовательность с изолированным пиком амплитуды 2 на 5 отсчете (а) и результат ее фильтрации (б); входная последовательность с изолированным пиком амплитуды 2 на 6 отсчете (в) и результат ее фильтрации (г)

На рис. 1а изображена входная последовательность с изолированным пиком при k = 5 и единичным скачком при k = 11, а на рис. 1б отклик стандартного медианного фильтра с шириной окна 9 отсчетов на эту последовательность. В данном случаем мы получаем именно тот результат, который характерен для медианных фильтров: изолированный пик полностью подавляется, а край единичного скачка сохраняется без искажений. Линейные фильтры уменьшили бы амплитуду изолированного пика, расширив его, и размыли бы край единичного скачка. На рис. 1в показаны та же входная последовательность, но с изолированным импульсом при k = 6, а на рис. 1г результат ее фильтрации тем же медианным фильтром. В данном случае импульс также полностью сглаживается, а единичный скачок остается совершенно резким, но его край сдвигается с k = 11 до k = 10.

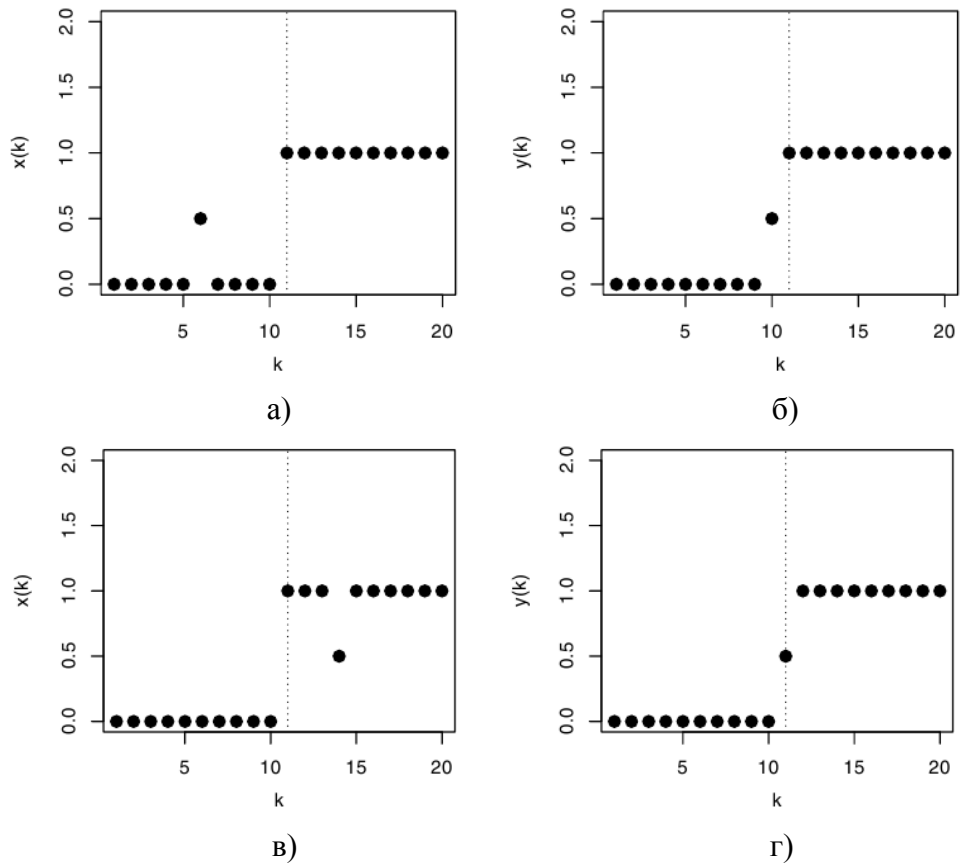


Рисунок 2. Входная последовательность с изолированным пиком амплитуды 0,5 на 6 отсчете (а) и результат ее фильтрации (б); входная последовательность с изолированным пиком амплитуды 0,5 на 14 отсчете (в) и результат ее фильтрации (г)

На рис. 2а изображены входная последовательность с изолированным пиком на 6 отсчете с амплитудой 0,5, а на рис. 2б отклик медианного фильтра с апертурой 9 отсчетов на такой сигнал. В результате фильтрации край оказался "сглаженным". Искажение сходно с искажениями, появляющимися при использовании линейного фильтра. На рис. 2в изображена та же входная последовательность, но с изолированным пиком на k=14. Результат фильтрации (рис. 2г) аналогичный – после обработки медианным фильтром край размывается [2, с.108].

Стабильные точки медианных фильтров

Стабильные точки – это последовательности, которые после пропускания через фильтр не меняют своей формы [4, с.89]. Стабильные точки могут быть определены для любого фильтра, включая линейные фильтры, но они оказались наиболее полезными при характеристике фильтров, принцип работы которых основан на определении медиан.

Для обычного медианного фильтра, основанного на симметричном движущемся окне шириной 2К+1, эти последовательности известны. Необходимо ввести следующие определения [4, с.90]:

- Серия – последовательность как минимум К+1 отсчетов имеющих одинаковое значение и следующих друг за другом.
- Край – монотонно возрастающая или убывающая последовательность, которой предшествуют и последуют серии различных значений.
- Импульс – последовательность не более К отсчетов, которой предшествуют и последуют серии одинаковых значений, отличающихся от значения этих К отсчетов.
- Колебание – это любая последовательность значений, не являющаяся серией, краем или импульсом.

Было определено, что последовательность является стабильной точкой для медианного фильтра, только тогда, когда она состоит лишь из серий и краев.

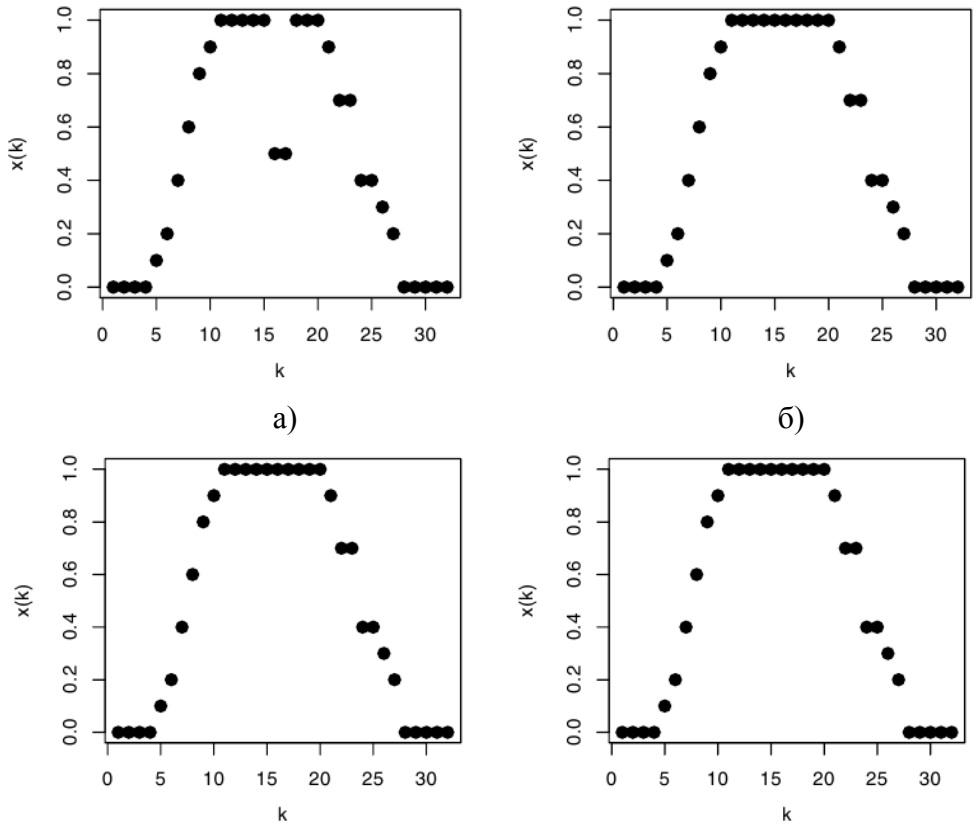


Рисунок 3. Входная последовательность, не являющаяся стабильной точкой (а), и результат ее фильтрации (б); стабильная точка (в) и результат ее фильтрации (г)

Последовательность, изображенная на рис. 3а имеет импульс, который удаляется медианным фильтром (рис. 3б). Последовательность, изображенная на рис. 3в состоит

только из серий и краев, поэтому является стабильной точкой для медианного фильтра, и сигнал проходит через фильтр без изменений (рис. 3г) [2, с.111].

Этот пример иллюстрирует важный момент, а именно понятие сходимости медианного фильтра. Это означает, что повторные проходы медианного фильтра по любой входной последовательности в конечном счете преобразуют её в стабильную точку, инвариантную к дальнейшему фильтрованию. В данном случае медианный фильтр выполнил такое преобразование за один проход, но в общем случае требуется несколько проходов медианного фильтра, чтобы преобразовать сигнал в стабильную точку.

L-фильтры

Медианный фильтр и его модификации являются частным случаем класса нелинейных фильтров, основанных на ранговой статистике. Этот класс включает в себя множество нелинейных фильтров таких как фильтры максимума/минимума, фильтры скользящего среднего и медианные фильтры. Нелинейные фильтры, основанные на ранговой статистике, были разработаны с учетом различных критериев, например, устойчивости, адаптивности к распределению вероятностей шума, сохранения краевой информации, сохранения деталей изображения. Таким образом, каждый из них имеет оптимальную производительность для конкретных показателей качества и специфических шумовых характеристик [4, с.135].

Класс L-фильтров состоит из таких фильтров, основанных на движущемся окне, чей отклик представляет собой линейную комбинацию упорядоченных входных отсчетов, попадающих в окно фильтра. Выходные отсчеты L-фильтра формируются по принципу:

$$y_i = \sum_{j=-K}^{K} a_j x_{(j)},$$

где $x_{(j)}$ – упорядоченные отсчеты входного сигнала, попадающие в окно фильтра, a_j – весовые коэффициенты отсчетов, K – полуширина окна.

Задавая весовые коэффициенты, можно получать различные виды фильтров [2, с.156]. Например:

1. Медианный фильтр: $a_0 = 1; a_{j\neq 0} = 0$
2. Невзвешенный линейный фильтр скользящего среднего: $a_j = \frac{1}{2K+1}$
3. Фильтр усеченного среднего: $a_j = \frac{1}{2J+1}$ для $j = -J...J$ при $J < K$; $a_j = 0$ для $j < -J,\ j > J$

Модифицированный медианный фильтр

Модифицированный медианный фильтр (другое название – усеченный фильтр среднего) является гибридом фильтра скользящего среднего и медианного фильтра. Он был разработан для эффективной борьбы как с гауссовским, так и с импульсным шумом.

Данный фильтр относится к классу L-фильтров, что значит, что в процессе работы значения входного сигнала, попадающие в апертуру фильтра, формируют упорядоченную по возрастанию или убыванию последовательность. Аномально большие и аномально малые значения оказываются на краях последовательности и в формировании выходного отсчета не участвуют. Таким образом удаляются импульсные помехи. Средние по величине значения последовательности усредняются как в фильтре скользящего среднего, что приводит к снижению гауссовского шума, имеющего по определению нулевое математическое ожидание.

Согласованный фильтр

Согласованный фильтр – это линейный оптимальный фильтр, предназначенный для выделения сигналов известной формы на фоне шумов. Основной задачей данного фильтра является получение максимального отношения сигнал/шум на выходе. Форма сигнала после прохождения через такой фильтр изменяется – максимума выходной сигнал достигает в момент окончания входного. В этот момент значение выходного сигнала равно энергии входного. Затем выходной сигнал уменьшается до нуля за время, равное продолжительности входного сигнала. Импульсная характеристика согласованного фильтра представляет собой зеркальное отражение принимаемого сигнала.

Так для входного сигнала, представленного прямоугольным импульсом, отклик согласованного фильтра имеет форму треугольной функции, а импульсная характеристика – прямоугольного импульса.

Выходной сигнал линейного фильтра может быть получен как свертка входного сигнала и импульсной характеристики фильтра: [3, с.147]

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k]$$

Постановка цели и задач

Целью данной работы является рассмотрение работы медианного фильтра, а также исследование модифицированного медианного фильтра с прямоугольной взвешивающей функцией. В ходе работы будет произведено сравнение этих двух фильтров по коэффициенту вносимых нелинейных искажений, а также по коэффициенту передачи для

различных частот входного сигнала. Кроме того необходимо сравнить данные фильтры с согласованным линейным фильтром по эффективности подавления шумов: гауссовского и импульсного.

Для определения уровня нелинейных искажений необходимо определить спектр сигнала. Чтобы получить хорошее разрешение по частоте требуется взять как можно больший временной интервал, однако мы ограничены вычислительной мощностью персонального компьютера. Поэтому входной сигнал будем задавать последовательностью 25000 отсчетов. Тогда шаг между составляющими спектра будет составлять $\frac{f_{дискр}}{N} = \frac{1}{25000}$.

Для эффективной фильтрации ширина окна медианного фильтра должна быть как можно больше. С другой стороны, при увеличении апертуры фильтра для сохранения разрешения по частоте необходимо увеличивать количество отсчетов входного сигнала, что мы можем делать лишь до определенного момента. Принимаем ширину окна фильтров равной 100 отсчетам. Тогда ширина главного лепестка в спектре окна медианного фильтра равна $\frac{1}{\tau_{имп}} = \frac{1}{100}$. Поэтому расчет будем производить для частот входного сигнала $f = (0,001...0,01)\, f_{дискр}$.

В ходе выполнения экспериментальной части работы требуется оперировать большими массивами данных, поэтому все расчеты целесообразно выполнять с помощью персонального компьютера. В качестве среды разработки программы будет использоваться Microsoft Visual Studio Express. Язык программирования C++ является одним из самых популярных языков программирования, с помощью которого создаются различные прикладные программы и приложения. Он имеет очень богатую стандартную библиотеку, включающую в себя всевозможные функции, алгоритмы, средства ввода-вывода и др. Немаловажным достоинством является то, что Microsoft Visual Studio Express распространяется бесплатно.

Поскольку встроенная в пакет Microsoft Visual Studio Express функция rand() генерирует равномерно распределенные значения, воспользуемся преобразованием Бокса-Мюллера, чтобы задать нормально распределенный шум. Для этого сгенерируем две независимые случайные величины U и V, равномерно распределенные на отрезке $[-1;1]$ и вычислим $R = U^2 + V^2$. Если $R \notin (0;1]$, то необходимо произвести генерацию U и V заново. Если же R попадает в интервал, то можно рассчитать величины

$C_1 = U\sqrt{\frac{-2\ln(R)}{R}}$; $C_2 = V\sqrt{\frac{-2\ln(R)}{R}}$, которые будут также независимы и при этом будут удовлетворять гауссовскому распределению.

Импульсный шум получим из гауссовского путем добавления зоны нечувствительности – выходное значение функции генерирования шума будет ненулевым только в том случае, когда его значение превышает определенную величину.

2. Расчетно-экспериментальный раздел

Расчет нелинейных искажений, вносимых фильтрами

В качестве входного сигнала примем синусоиду, поскольку ее спектр представляет собой одну дискрету, и определение коэффициента нелинейных искажений не вызывает сложности:

$$K_{НИ} = \sqrt{\frac{\sum_{k \neq f_c} X^2[k]}{X^2[f_c]}},$$

где f_c – отсчет, соответствующий первой гармонике сигнала.

Поскольку у нас нет возможности исследовать сигнал на бесконечном интервале времени, мы вынуждены ограничивать его длительность. Это ограничение равносильно произведению сигнала и оконной функции:

$$x_{взвеш}[n] = x[n] \cdot W[n]$$

Простое ограничение выборки N отсчетами эквивалентно применению прямоугольного окна, т.е. взвешивающей (оконной) функции вида:

$$W[n] = \begin{cases} 1, & n \in [0, N-1] \\ 0, & n \notin [0, N-1] \end{cases}$$

Данная взвешивающая функция имеет высокий уровень боковых лепестков (-13 дБ) и для исследования не подходит. Наличие боковых лепестков приводит к "размыванию" частотных составляющих сигнала в пределах всей ширины анализируемого диапазона частот.

Чтобы существенно уменьшить уровень боковых лепестков используем оконную функцию Блэкмана-Харриса, максимальный уровень боковых лепестков для которой равен -92 дБ:

$$W[n] = 0,35875 + 0,48829\left(\frac{2\pi}{N}(n-\alpha)\right) + 0,14128\left(\frac{4\pi}{N}(n-\alpha)\right) + 0,01168\left(\frac{6\pi}{N}(n-\alpha)\right),$$

где для четного N: $\alpha = \frac{N}{2} - 1$

Однако выиграв в уровне боковых лепестков, мы проигрываем в ширине главного лепестка. Из-за этого при определении первой гармоники сигнала необходимо учитывать значение не только одного отсчета на частоте сигнала, но и отсчетов в некоторой области вокруг него.

Для определения спектра сигналов воспользуемся дискретным преобразованием Фурье. Определим вещественную и м043Dимую части: [3, с.190]

$$\begin{cases} \operatorname{Re} X[k] = \sum_{i=0}^{N-1} x[i] \cdot \cos\left(\frac{2\pi k i}{N}\right) \\ \operatorname{Im} X[k] = -\sum_{i=0}^{N-1} x[i] \cdot \sin\left(\frac{2\pi k i}{N}\right) \end{cases}$$

Находим модуль:

$$X[k] = \sqrt{\left(\operatorname{Re} X[k]\right)^2 + \left(\operatorname{Im} X[k]\right)^2}$$

Построим главный лепесток в спектре входного сигнала, например, для частоты 0,001$f_{дискр}$:

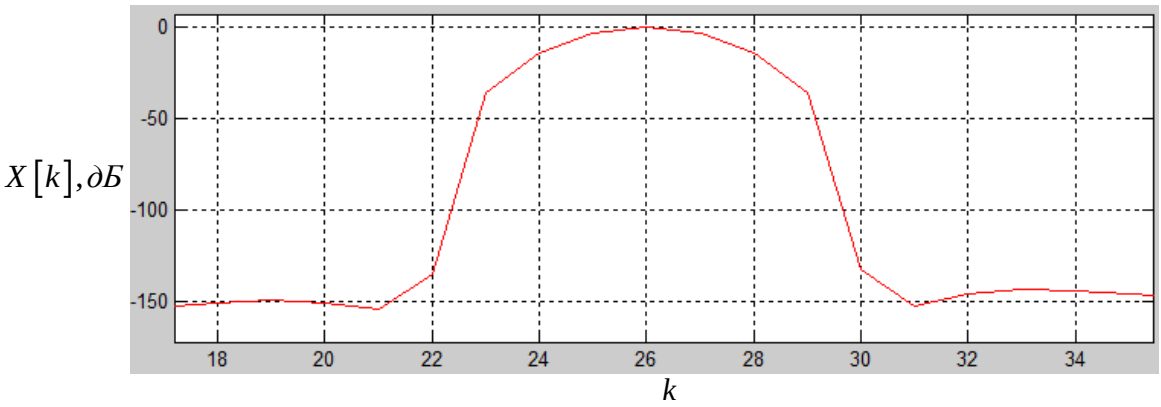


Рисунок 4. Главный лепесток в спектре входного сигнала частоты 0,001$f_{дискр}$

На рис. 4 проиллюстрировано, что из-за применения взвешивающей функции ширина главного лепестка существенно увеличилась. Основываясь на полученном графике, принимаем ширину главного лепестка равной 9 отсчетам. Тогда для определения первой гармоники сигнала будем пользоваться формулой:

$$U_{m1} = \sqrt{\sum_{k=f_c-4}^{f_c+4} X^2[k]}$$

В таком случае коэффициент нелинейных искажений:

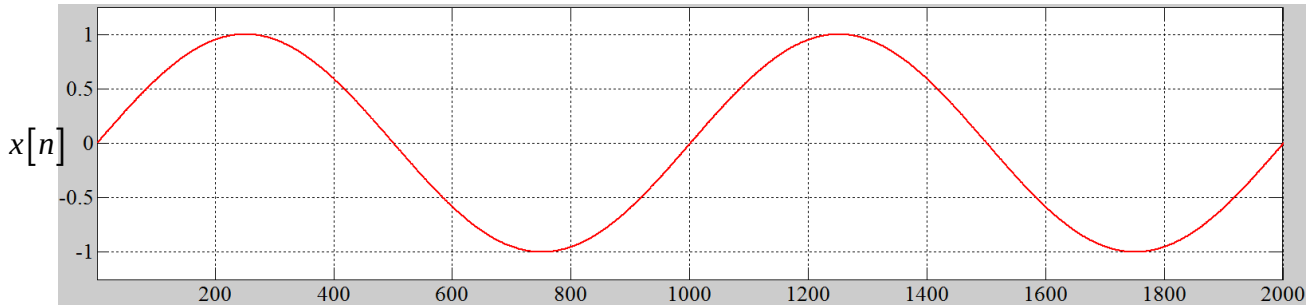
$$K_{НИ} = \sqrt{\frac{\sum_{k=0}^{f_c-5} X^2[k] + \sum_{k=f_c+5}^{N/2} X^2[k]}{\sum_{k=f_c-4}^{f_c+4} X^2[k]}}$$

Кроме коэффициента нелинейных искажений рассчитаем также коэффициент передачи фильтров:

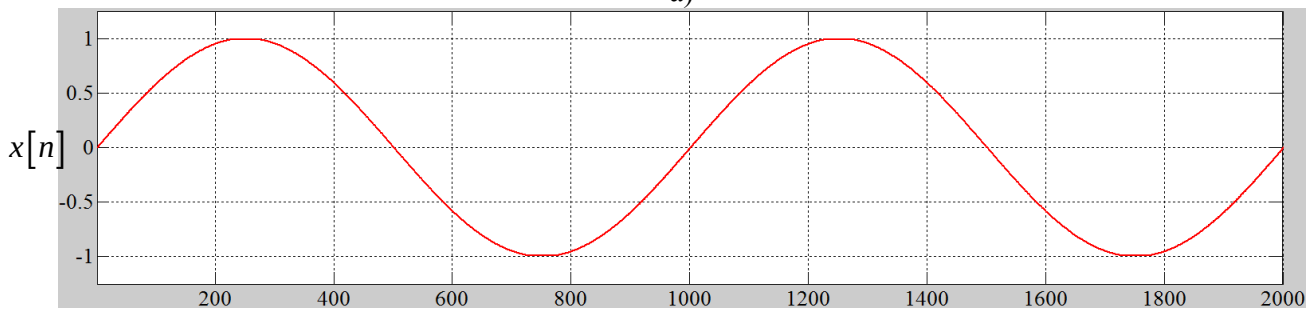
$$K = \sqrt{\frac{\sum_{k=f_c-4}^{f_c+4} X^2_{\text{после фильтр}}[k]}{\sum_{k=f_c-4}^{f_c+4} X^2_{вх}[k]}}$$

Построим временные и спектральные представления входного сигнала и сигналов на выходе исследуемых фильтров для некоторых частот входного сигнала, вычисленные в результате работы программы, представленной в приложении 1.

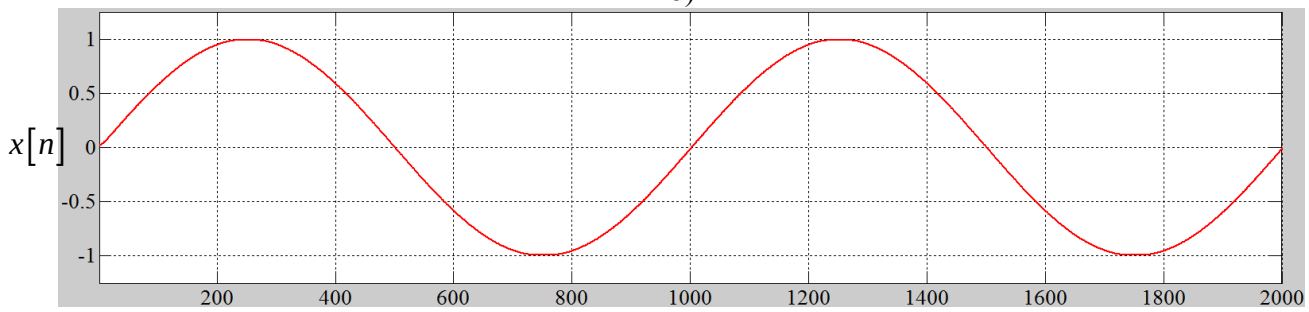
Для частоты 0,001$f_{дискр}$:



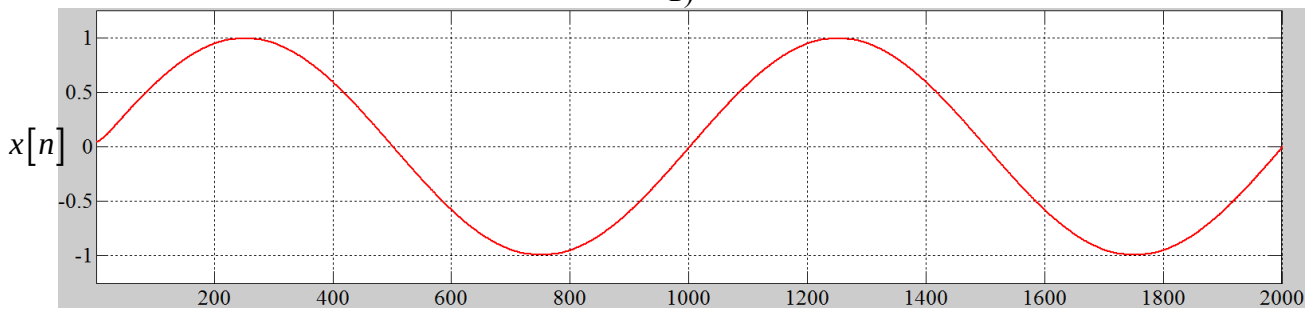
а)



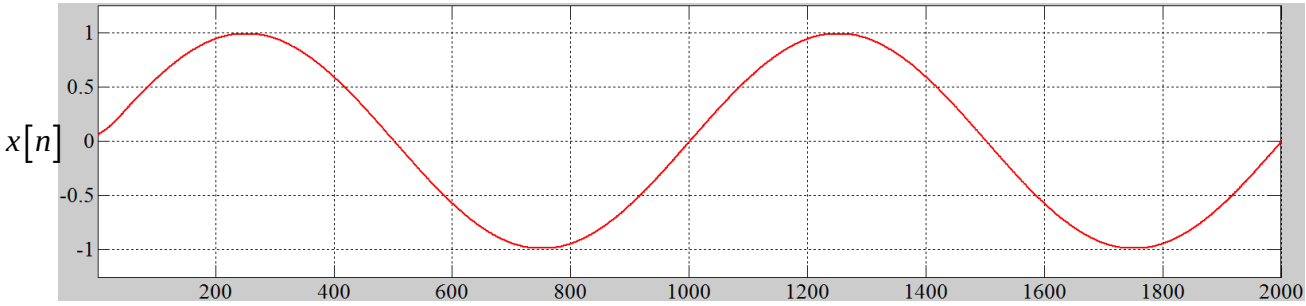
б)



в)

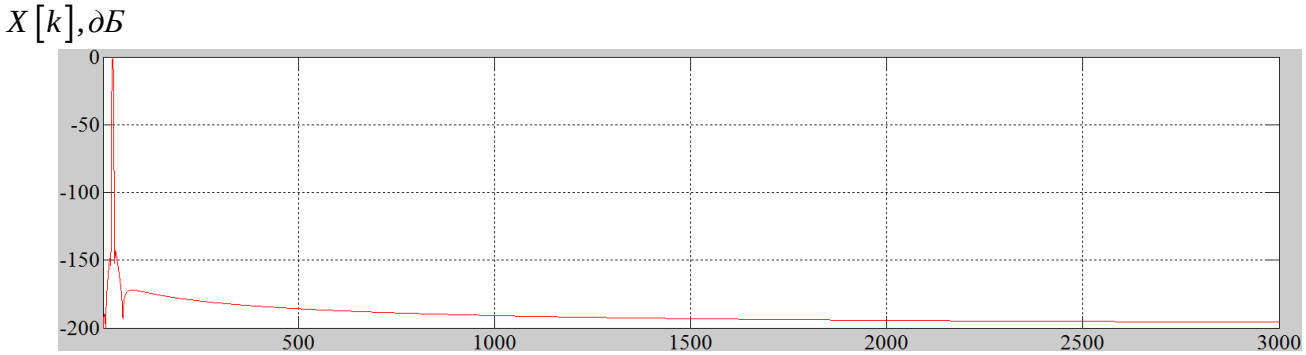


г)

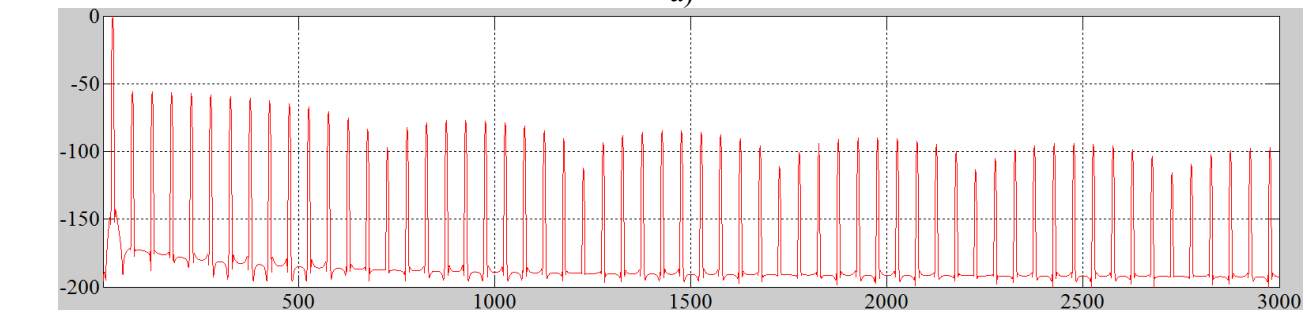


д)

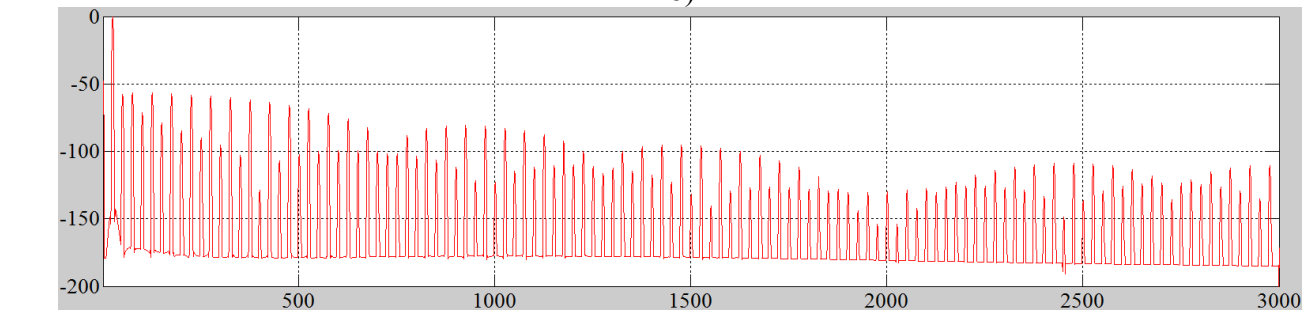
Рисунок 5. Временное представление входного сигнала частотой 0,001$f_{дискр}$(а), результата обработки медианным фильтром (б) и модифицированными медианными фильтрами с шириной окна 25% (в), 50% (г) и 75% (д)



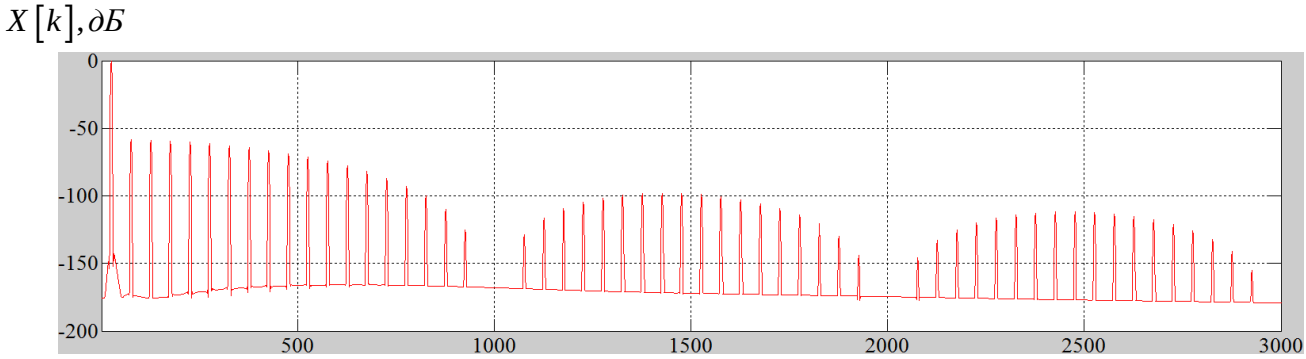
а)



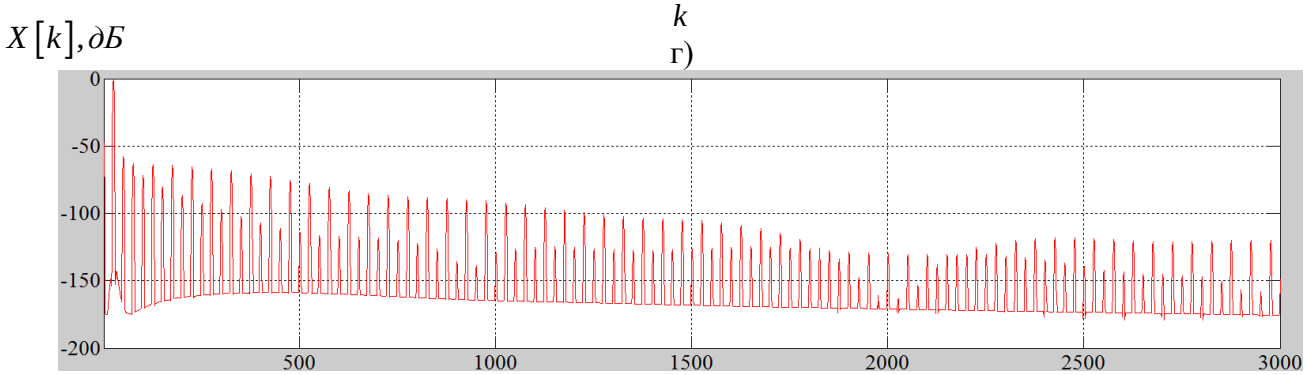
б)



в)



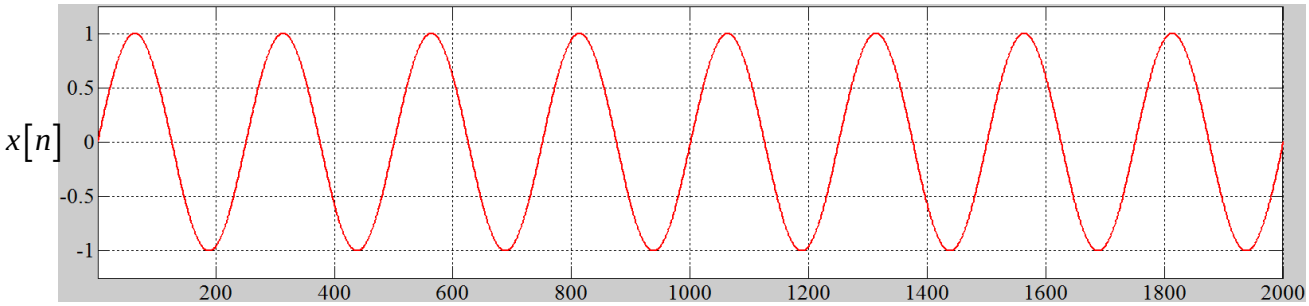
г)



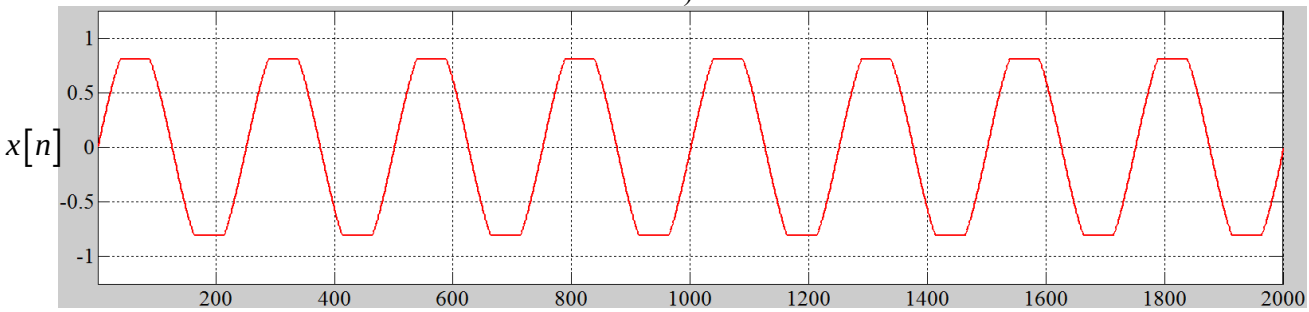
д)

Рисунок 6. Спектральное представление входного сигнала частотой 0,001$f_{дискр}$ (а), результата обработки медианным фильтром (б) и модифицированными медианными фильтрами с шириной окна 25% (в), 50% (г) и 75% (д)

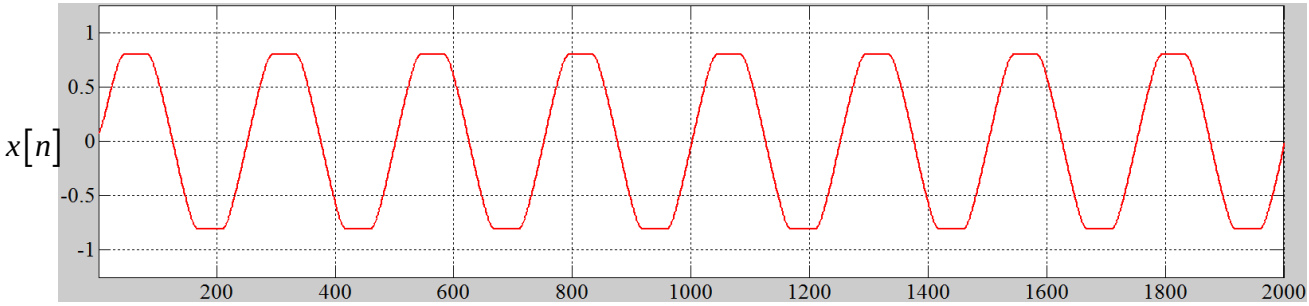
Для частоты 0,004$f_{дискр}$:



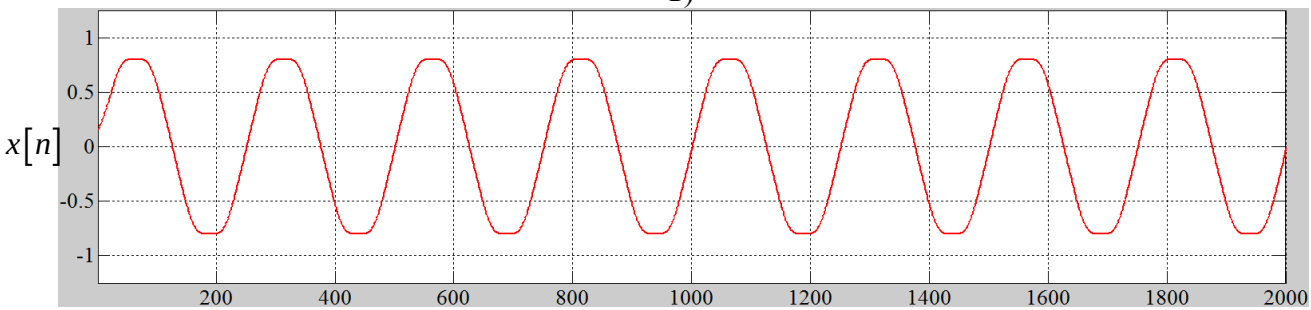
а)



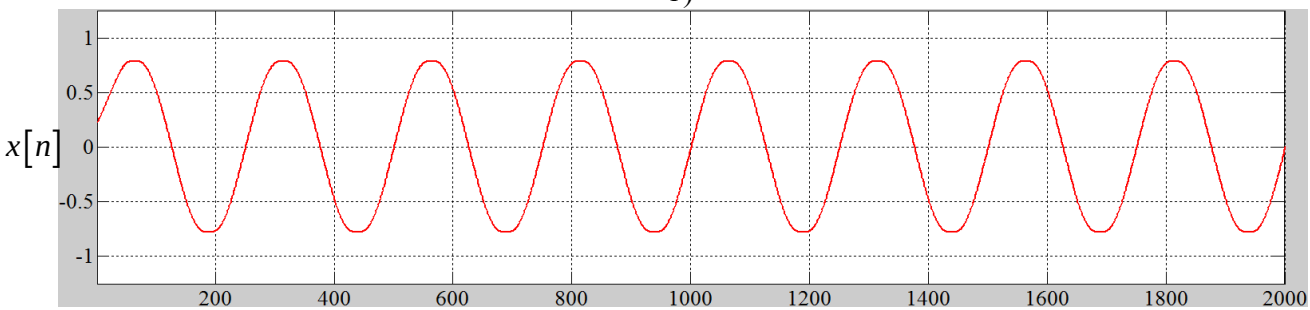
б)



в)

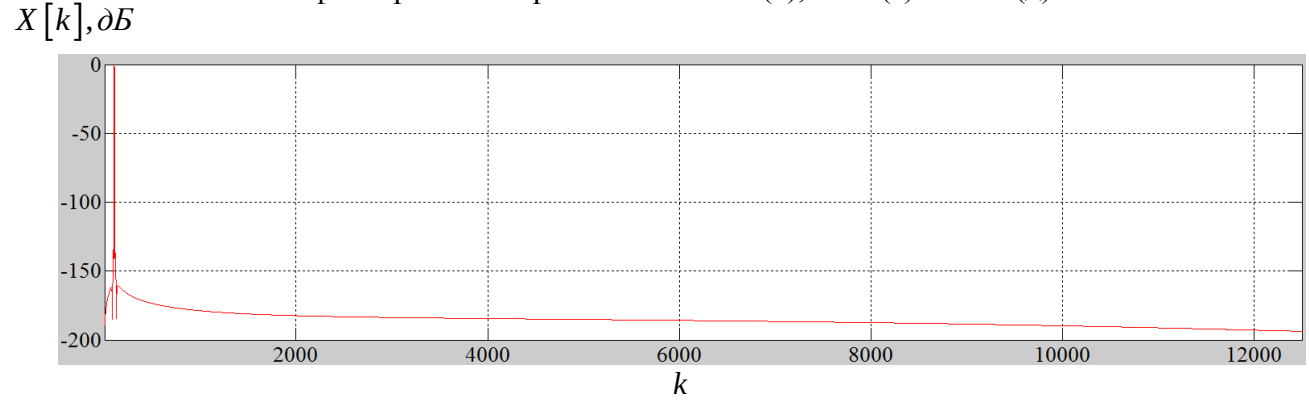


г)

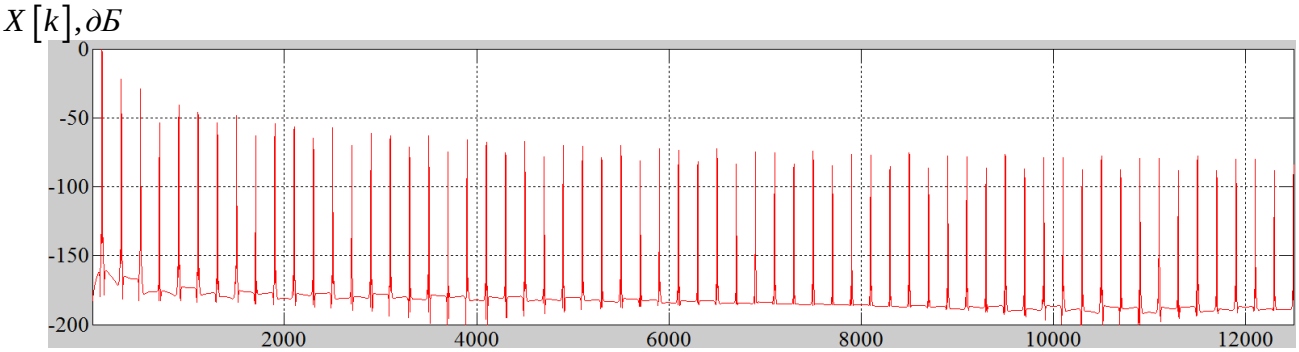


д)

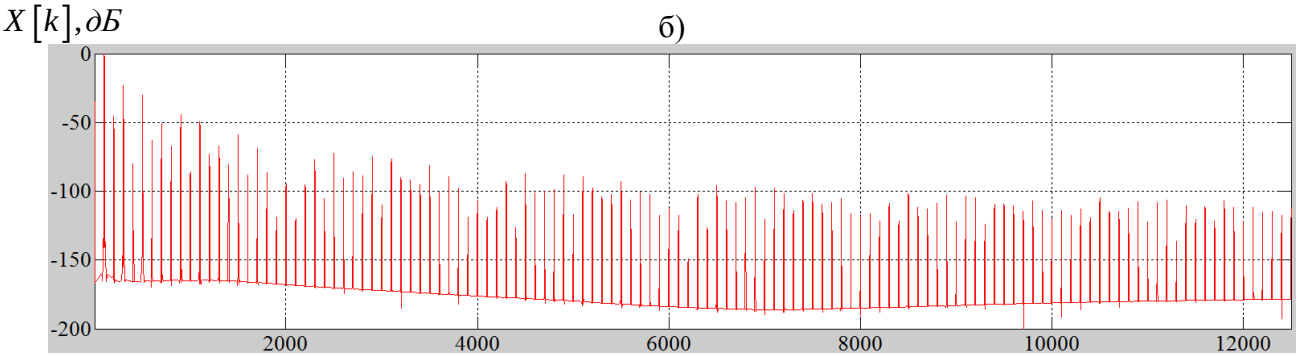
Рисунок 7. Временное представление входного сигнала частотой 0,004$f_{дискр}$ (а), результата обработки медианным фильтром (б) и модифицированными медианными фильтрами с шириной окна 25% (в), 50% (г) и 75% (д)



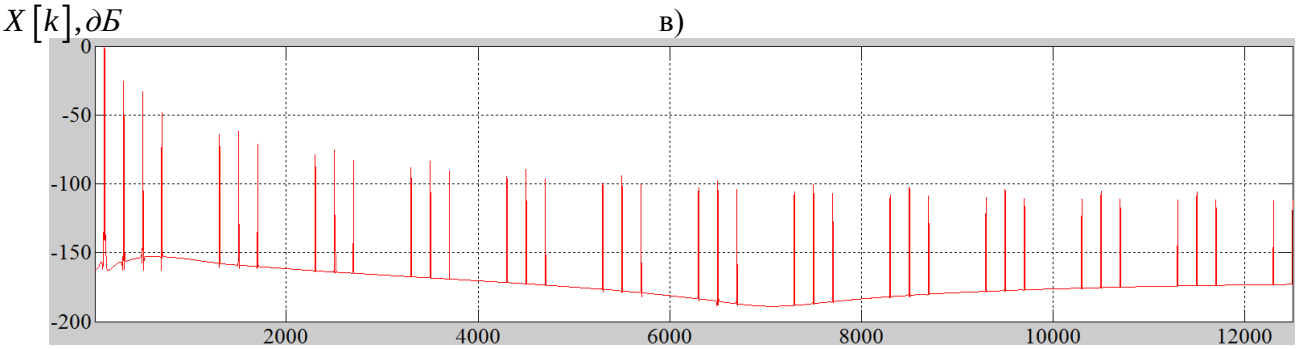
а)



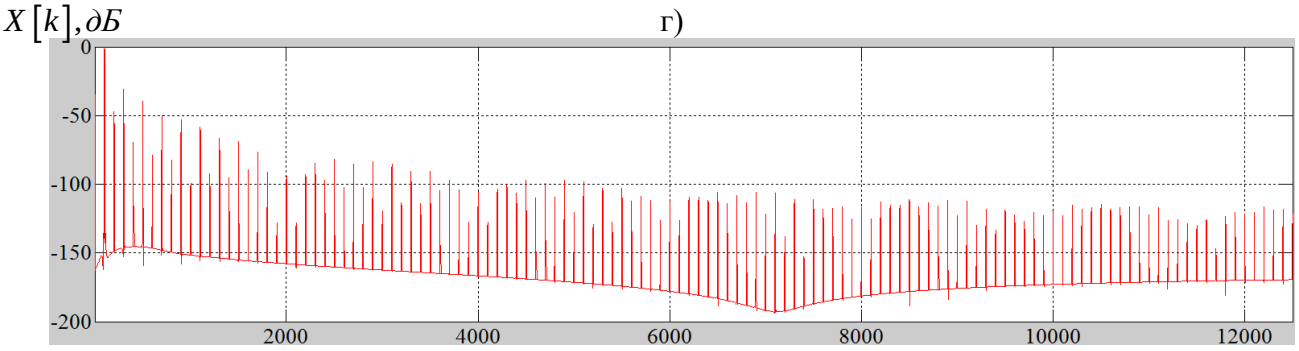
б)



в)



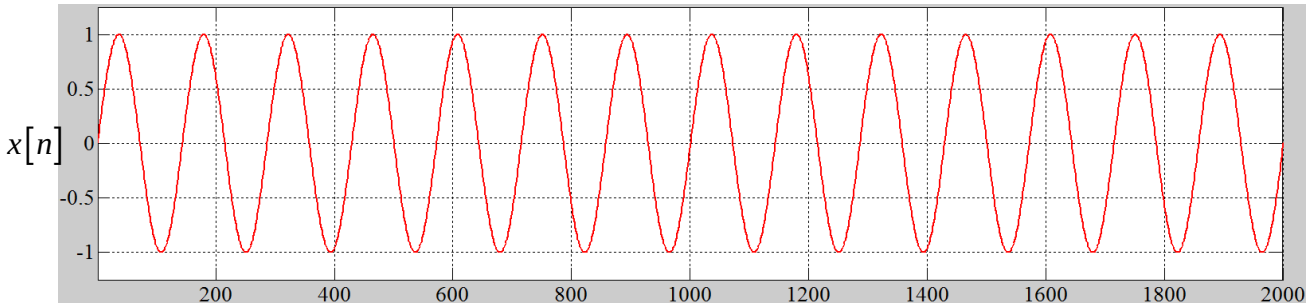
г)



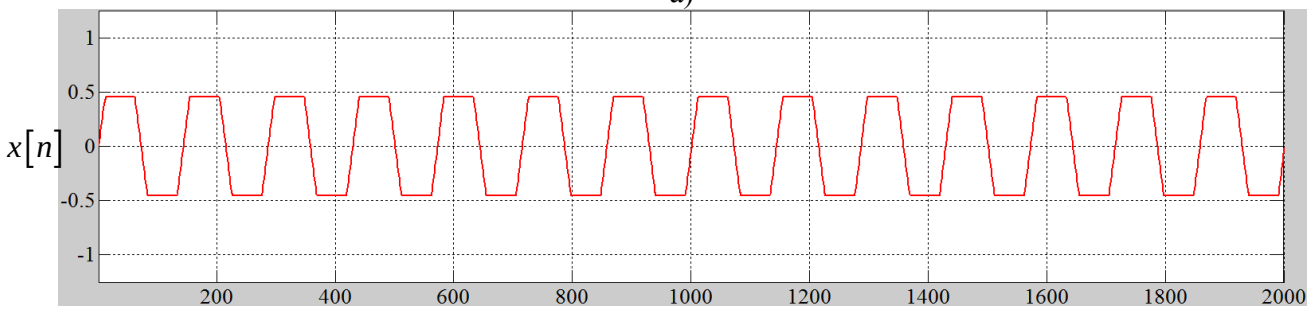
д)

Рисунок 8. Спектральное представление входного сигнала частотой 0,004$f_{дискр}$ (а), результата обработки медианным фильтром (б) и модифицированными медианными фильтрами с шириной окна 25% (в), 50% (г) и 75% (д)

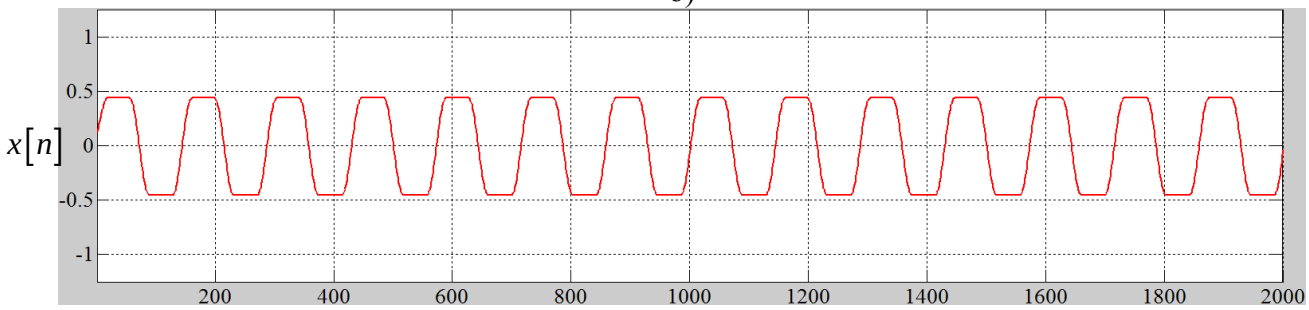
Для частоты 0,007$f_{дискр}$:



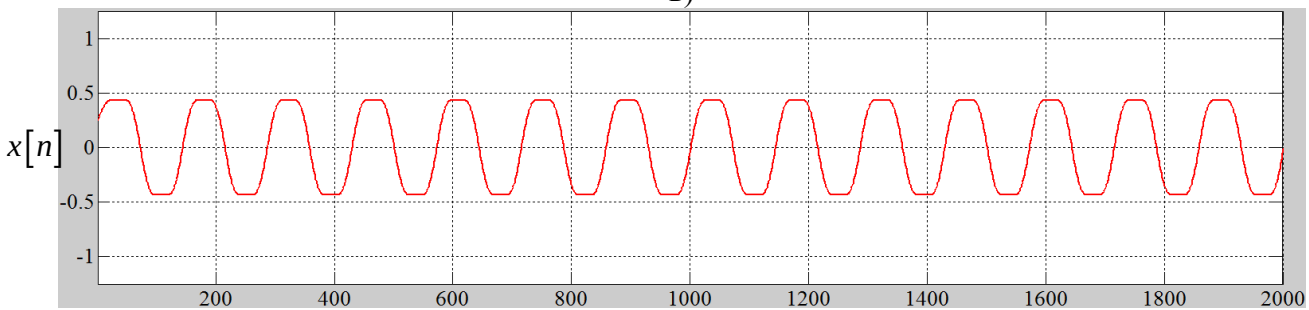
а)



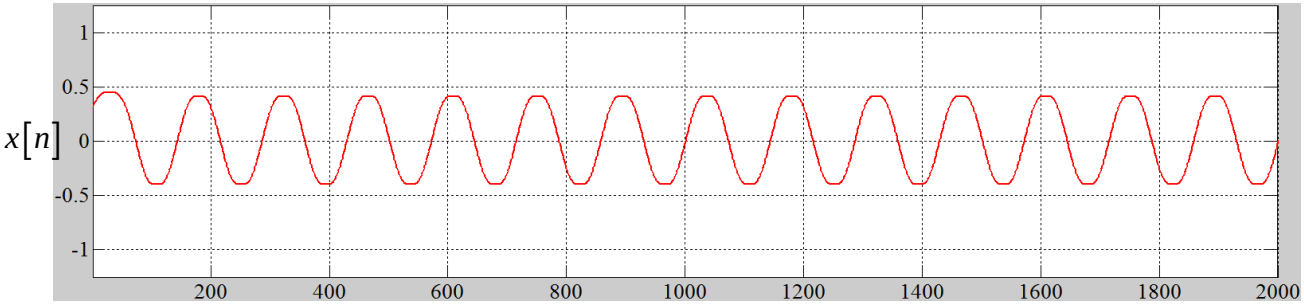
б)



в)

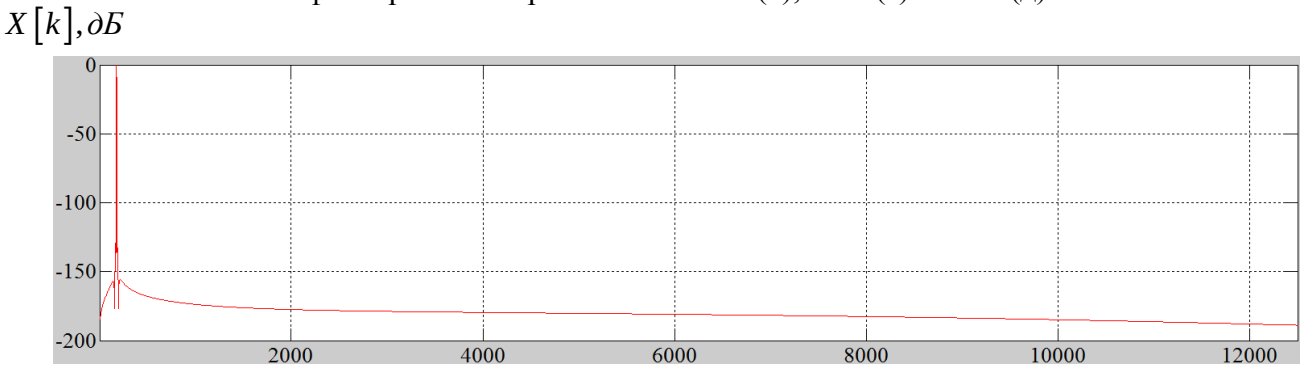


г)

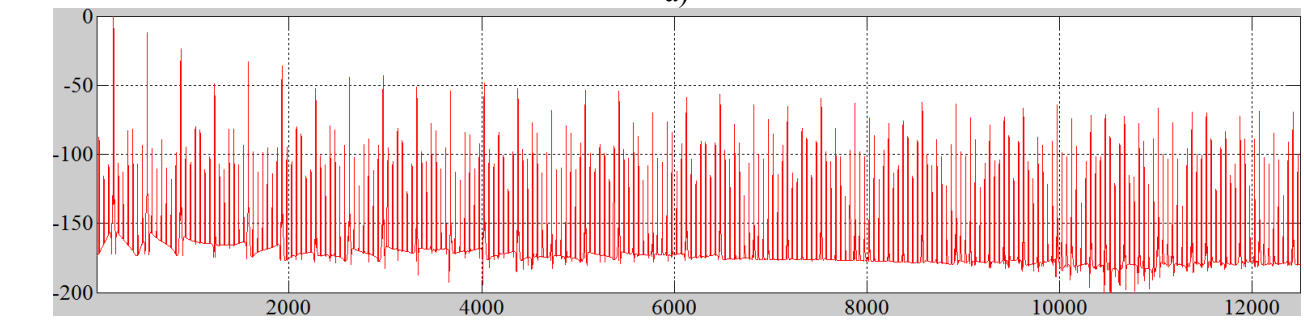


д)

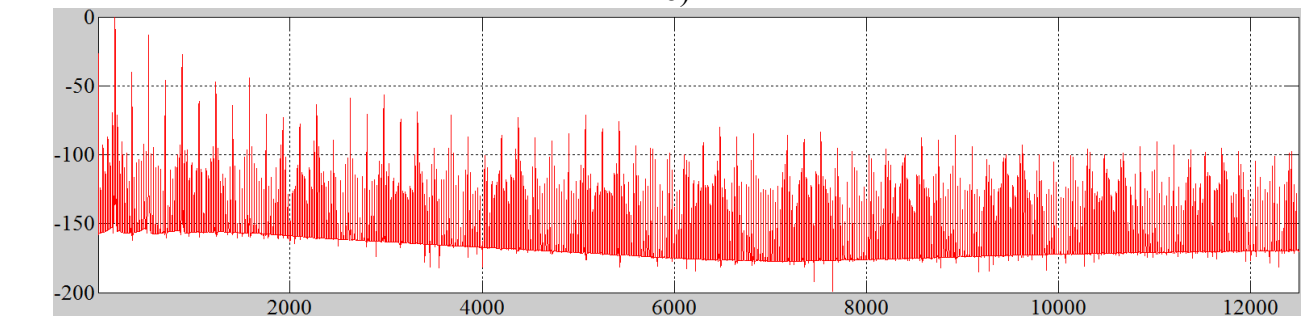
Рисунок 9. Временное представление входного сигнала частотой 0,007$f_{дискр}$ (а), результата обработки медианным фильтром (б) и модифицированными медианными фильтрами с шириной окна 25% (в), 50% (г) и 75% (д)



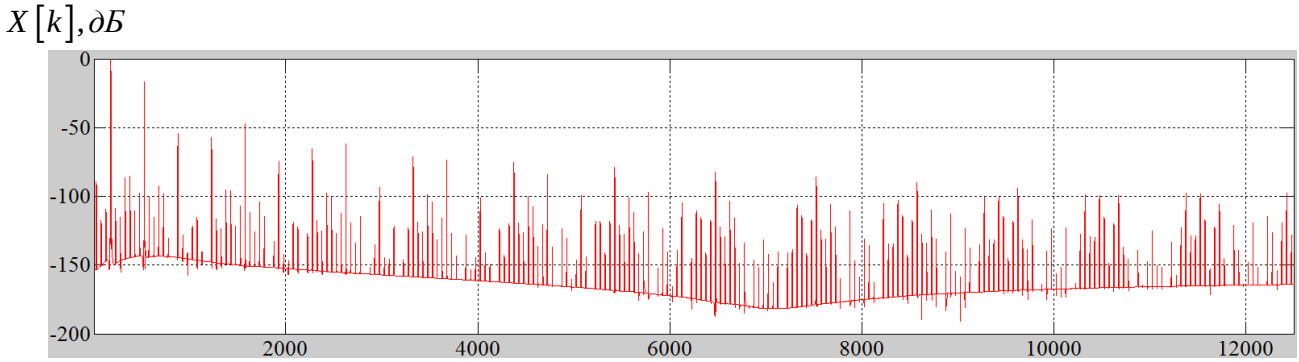
а)



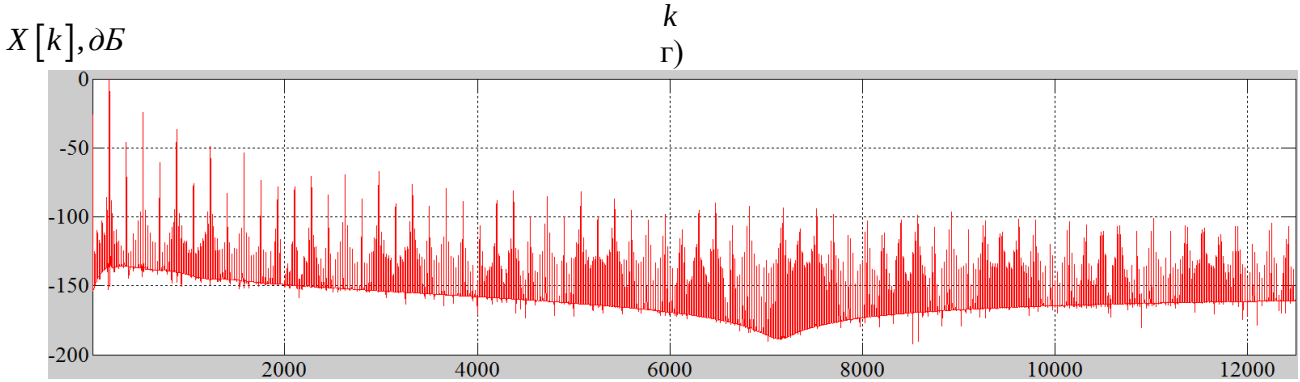
б)



в)



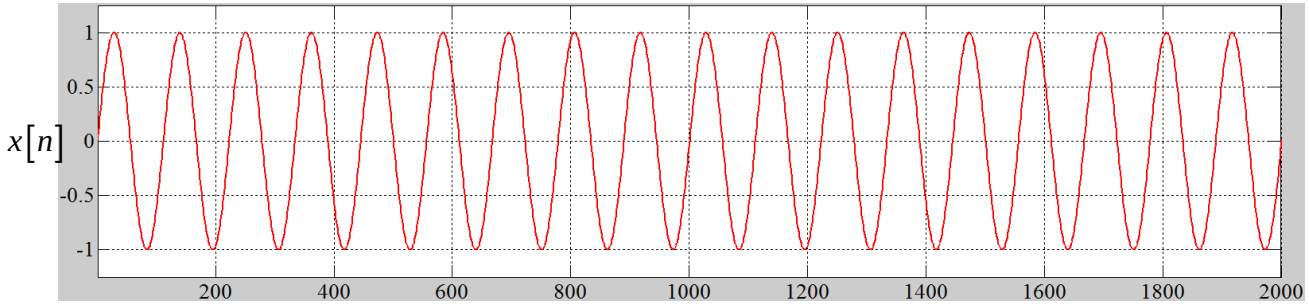
г)



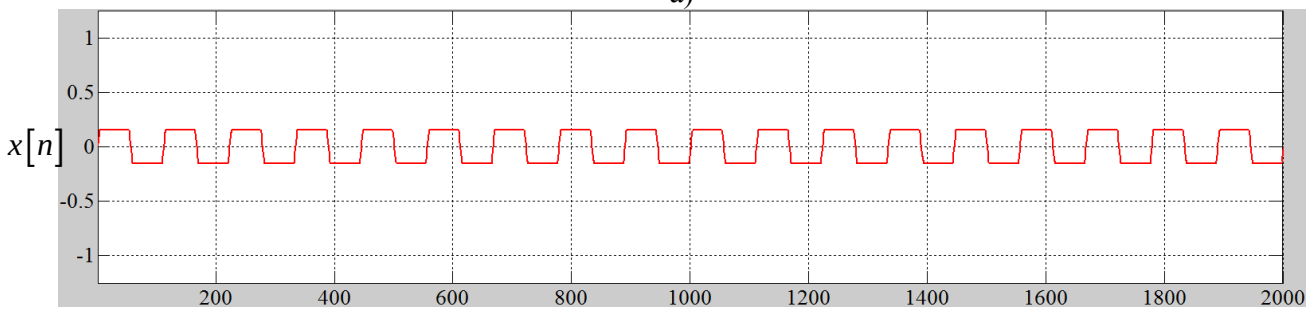
д)

Рисунок 10. Спектральное представление входного сигнала частотой 0,007$f_{дискр}$ (а), результата обработки медианным фильтром (б) и модифицированными медианными фильтрами с шириной окна 25% (в), 50% (г) и 75% (д)

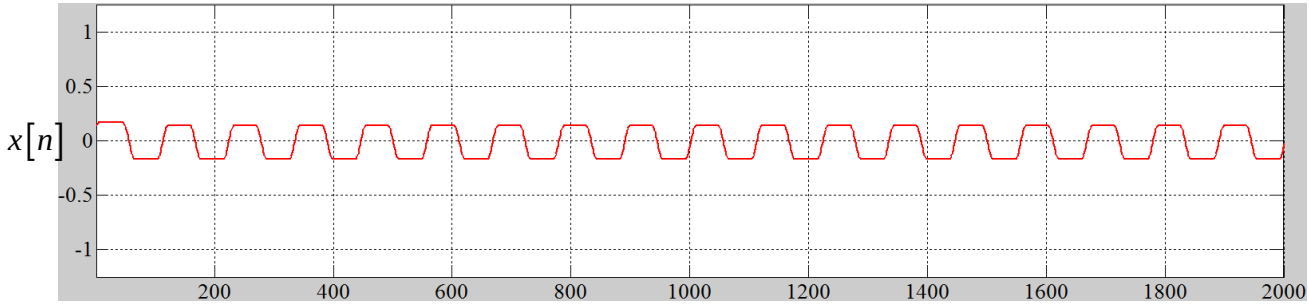
Для частоты 0,009$f_{дискр}$:



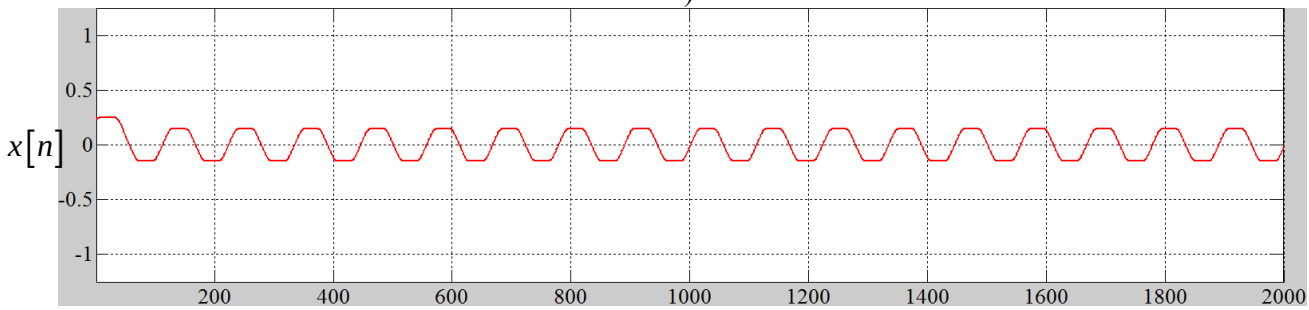
а)



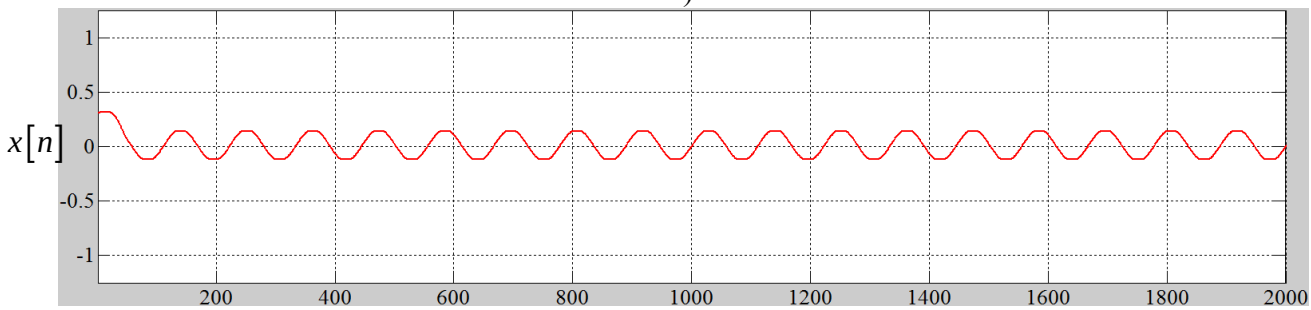
б)



в)

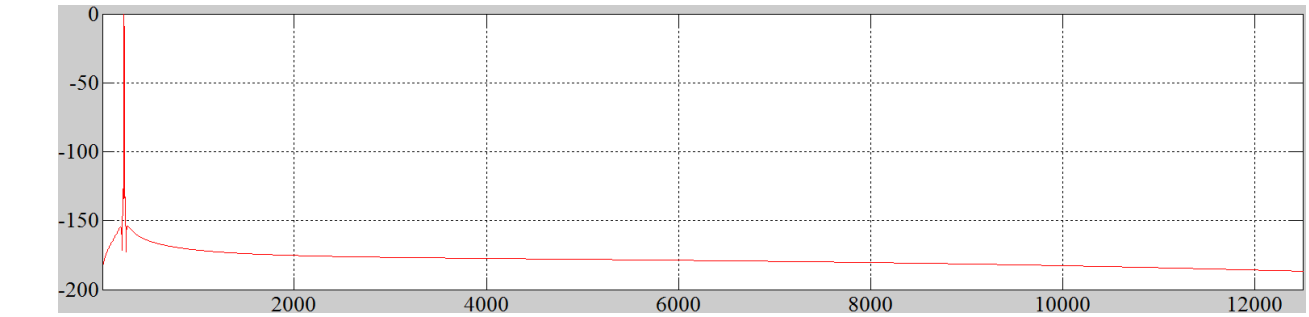


г)



д)

Рисунок 11. Временное представление входного сигнала частотой 0,009$f_{дискр}$ (а), результата обработки медианным фильтром (б) и модифицированными медианными фильтрами с шириной окна 25% (в), 50% (г) и 75% (д)



а)

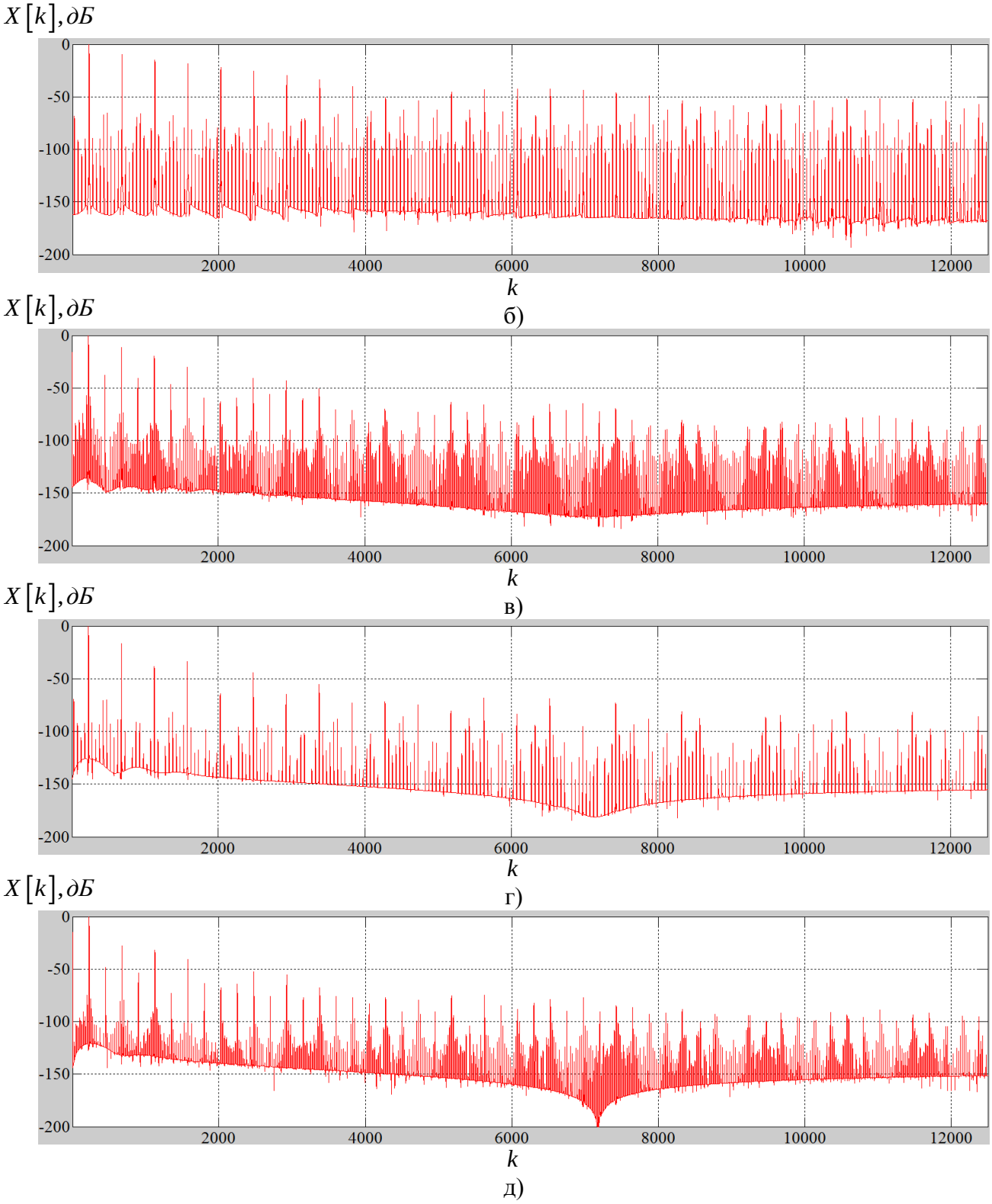


Рисунок 12. Спектральное представление входного сигнала частотой 0,009$f_{дискр}$ (а), результата обработки медианным фильтром (б) и модифицированными медианными фильтрами с шириной окна 25% (в), 50% (г) и 75% (д)

Построим зависимости коэффициента нелинейных искажений (КНИ) и коэффициента передачи по мощности (Кр) от частоты входного сигнала:

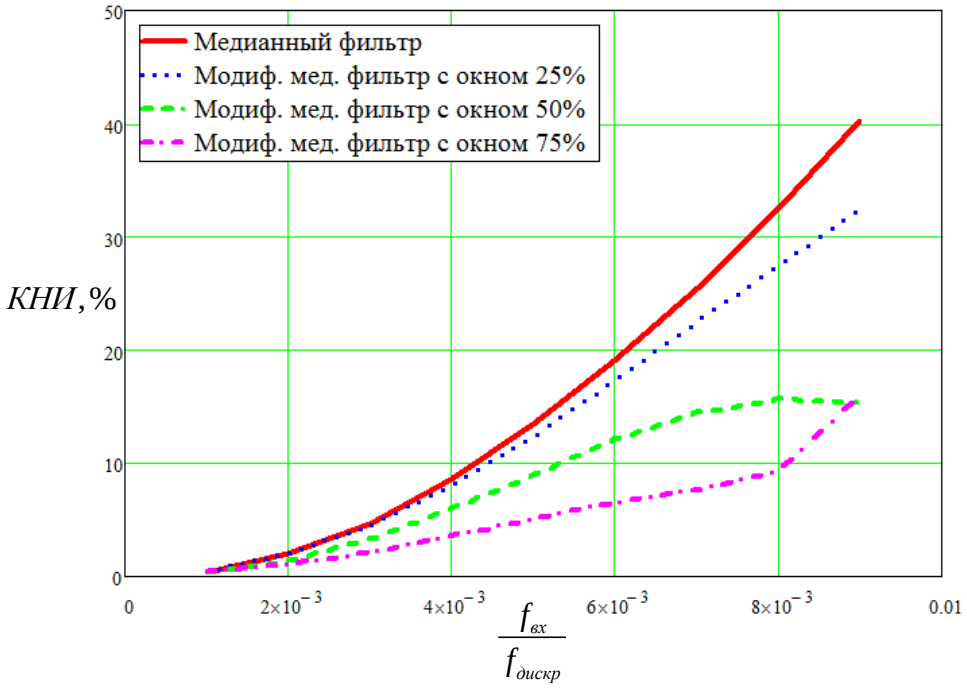


Рисунок 13. Частотные зависимости коэффициентов нелинейных искажений фильтров

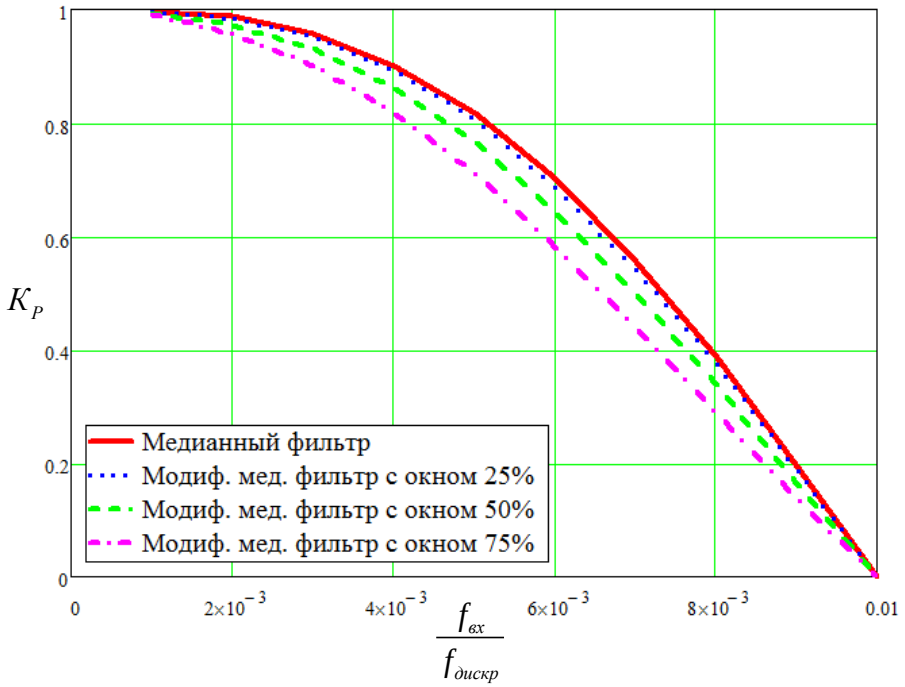


Рисунок 14. Частотные зависимости коэффициентов передачи по мощности фильтров

В наихудшем случае (для частоты $0{,}01f_{дискр}$) коэффициент нелинейных искажений для входного сигнала составляет $K_{НИ} = 1{,}051 \cdot 10^{-6} = 1{,}051 \cdot 10^{-4}\%$. Эта ошибка возникает из-за использования взвешивающей функции и округлений при расчете. Таким образом можно утверждать, что ошибка в расчетах составляет не более $2 \cdot 10^{-4}\%$.

При частоте $f = 0,01 f_{дискр}$ коэффициент передачи стремится к нулю, поэтому определение коэффициента нелинейных искажений не имеет смысла.

Выводы:

Как и ожидалось, применение нелинейных медианного и модифицированных медианных фильтров к синусоидальному сигналу приводит нелинейным искажениям из-за "обрезания" вершин. Отсчеты, находящиеся в пике синусоиды, после упорядочивания внутри окна фильтра не используются при формировании выходного отсчета, поскольку не могут оказаться медианой или попасть в окно усреднения модифицированного медианного фильтра. Соответственно, для медианного фильтра количество таких "неиспользуемых" отсчетов максимально, а для модифицированных медианных фильтров с увеличением ширины окна усреднения их все меньше. Другими словами, чем ближе фильтр к линейному фильтру скользящего среднего, в котором усреднение происходит по всем отсчетам и, следовательно, используются все отсчеты входного сигнала, тем меньше уровень нелинейных искажений. Поэтому модифицированные медианные фильтры с шириной окна усреднения 50% и 75% дают существенный выигрыш в величине коэффициента нелинейных искажений по сравнению с медианным фильтром.

При увеличении частоты входного сигнала в окно фильтра попадает все большая часть периода синусоиды, поэтому когда захватывается ее пик, выходное значение фильтра становится все меньше. Соответственно, уменьшается коэффициент передачи. Коэффициент нелинейных искажений также растет из-за все большего "обрезания" пиков синусоиды. При частоте $0,01 f_{дискр}$ в окно фильтра попадает ровно период, поэтому медианное значение равно нулю – выходного сигнала нет при любом положении окна.

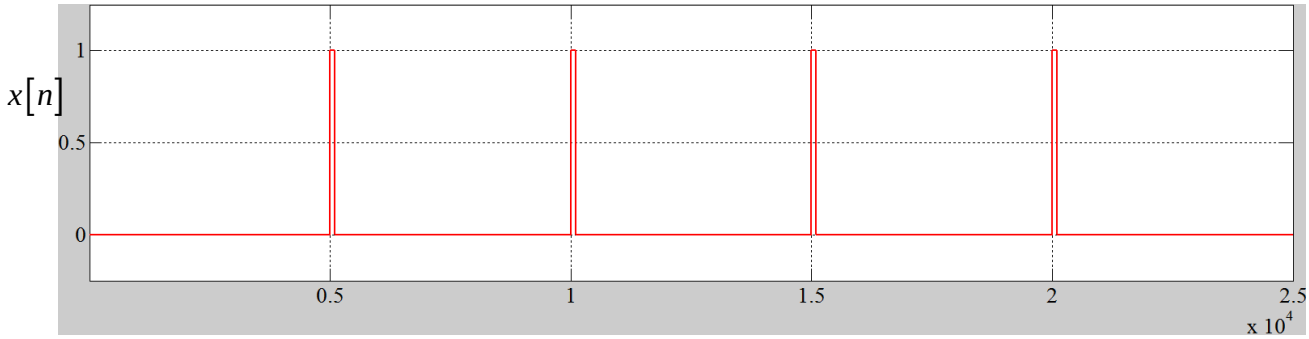
Расчет отношения сигнал/шум при воздействии гауссовского шума

В качестве входного сигнала принимаем периодическую последовательность прямоугольных импульсов с длительностью 100 отсчетов и амплитудой 1. В качестве шума возьмем аддитивный гауссовский шум. Исследование будем проводить для двух мощностей шума, обеспечивающих ОСШ на входе равным 10 и 2.

Полученные результаты сравним с результатами согласованного фильтра. Импульсная характеристика согласованного фильтра представляет собой зеркальное отражение обрабатываемого сигнала, следовательно, поскольку сигнал представляет собой прямоугольный импульс единичной амплитуды, в данном случае:

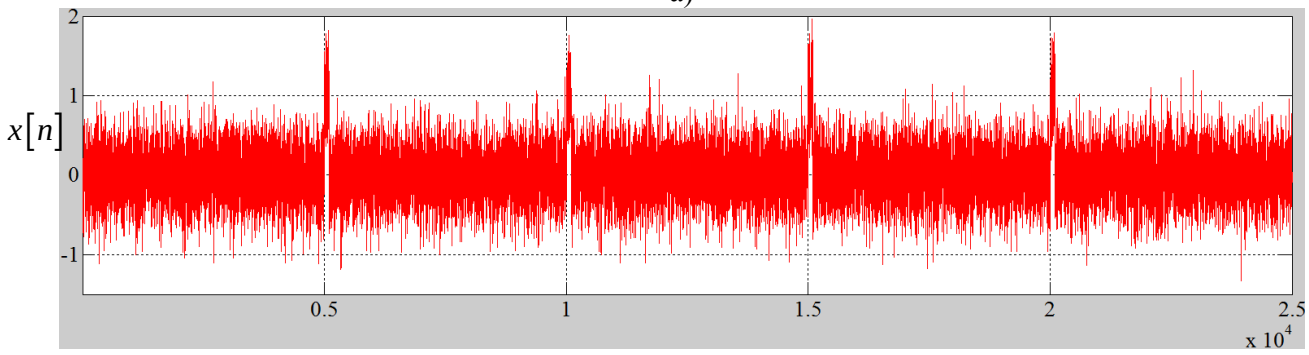
$$h[n]=\begin{cases}1, & n\in[0,100]\\ 0, & \textit{иначе}\end{cases}$$

Построим временные зависимости входного сигнала и сигналов, прошедших фильтрацию медианным, модифицированными медианными и согласованным фильтрами для случая $q_{вх}^2=10$ (см. приложение 2):



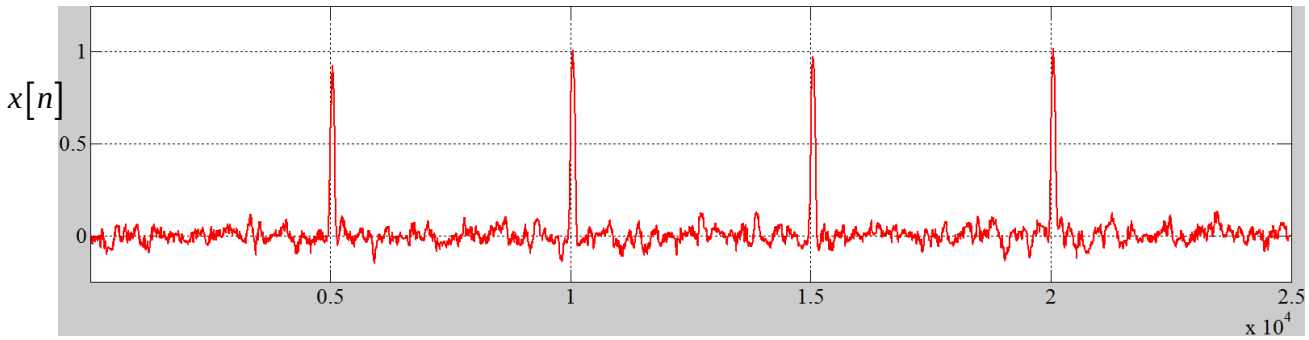
n

а)



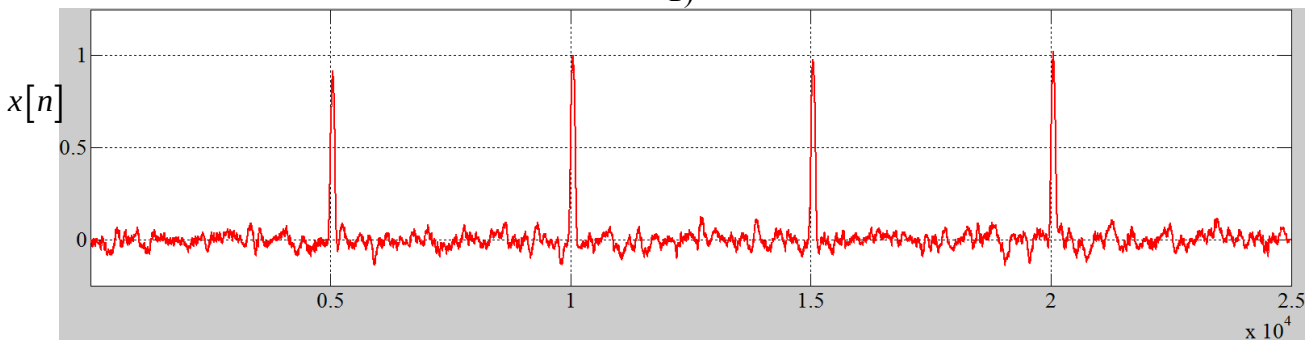
n

б)


x[n]
1
0.5
0
0.5
1
1.5
2
2.5
x 10^4

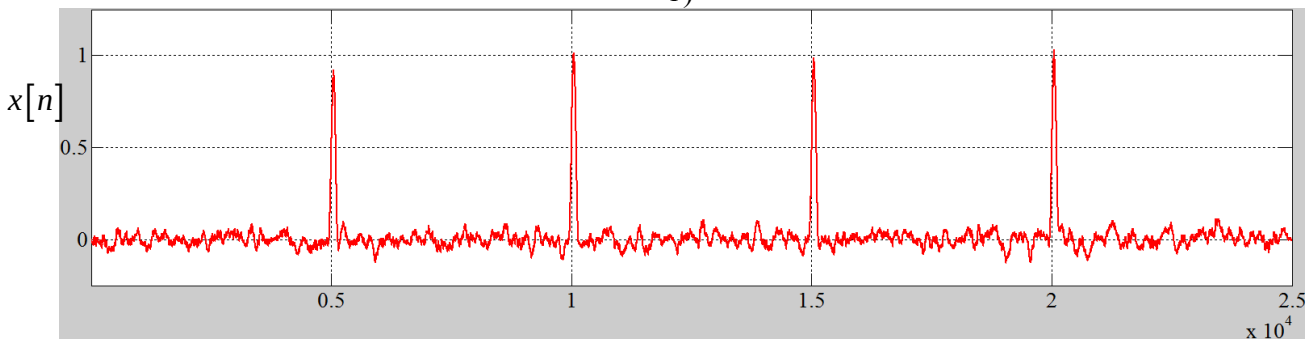
n

в)


x[n]
1
0.5
0
0.5
1
1.5
2
2.5
x 10^4

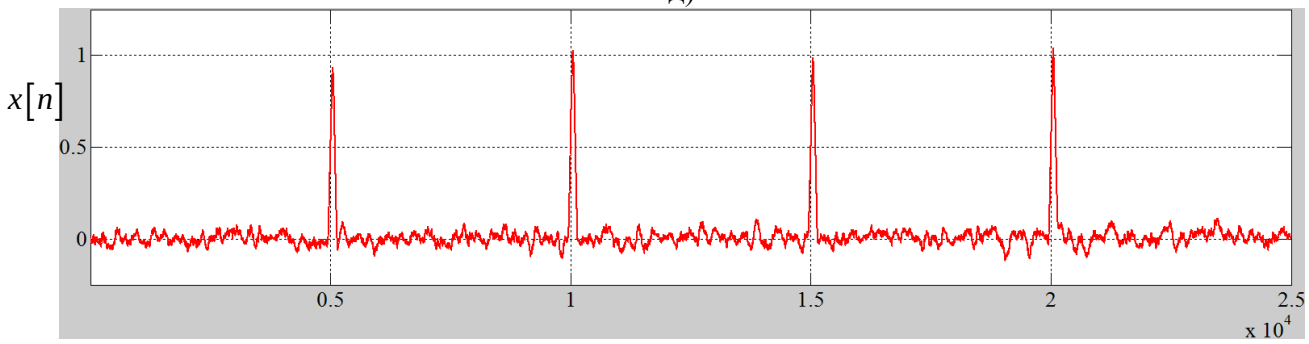
n

г)


x[n]
1
0.5
0
0.5
1
1.5
2
2.5
x 10^4

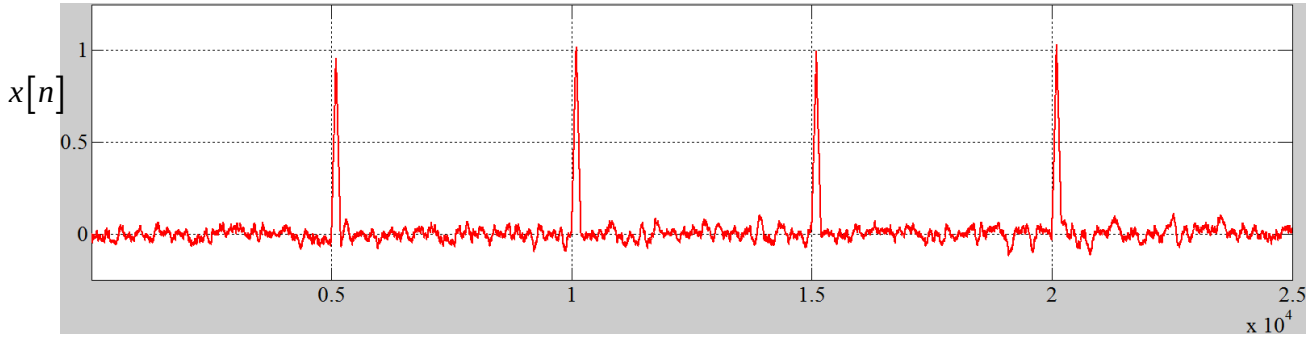
n

д)


x[n]
1
0.5
0
0.5
1
1.5
2
2.5
x 10^4

n

е)

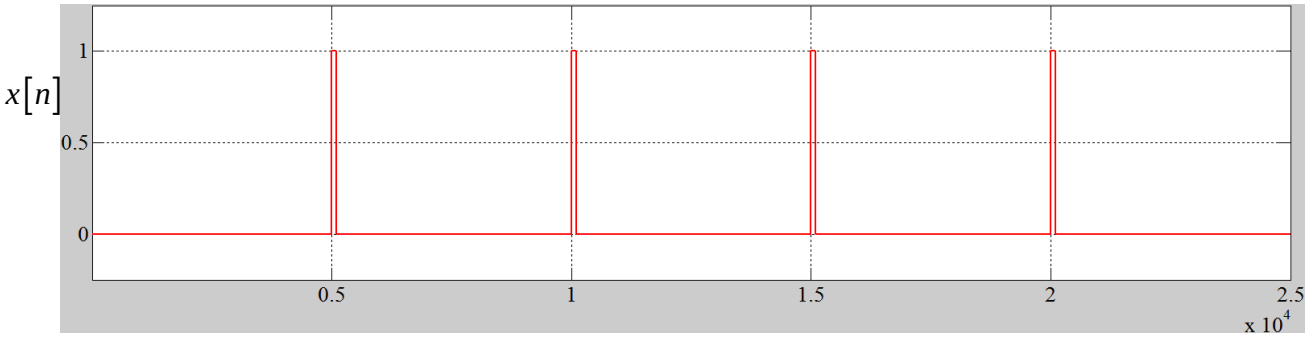


n

ж)

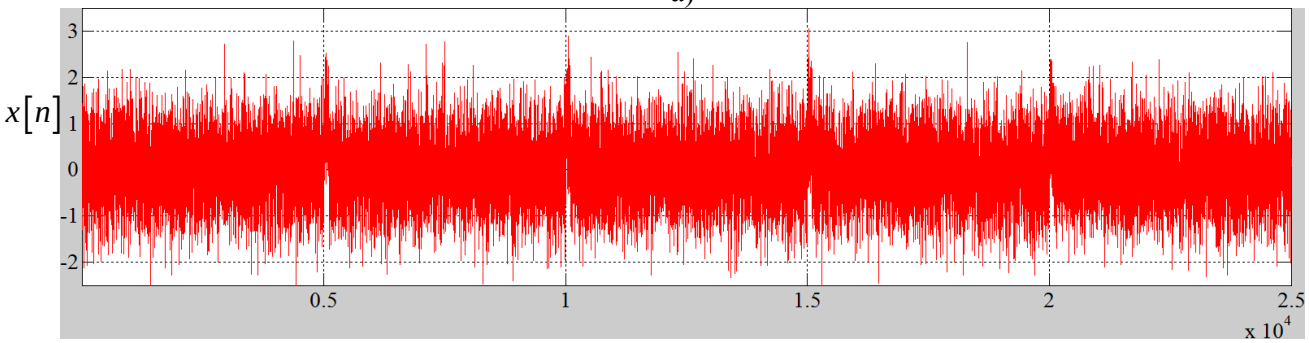
Рисунок 15. Временное представление входного сигнала без шума (а), с аддитивным гауссовским шумом для случая $q_{вх}^2 = 10$ (б), результата обработки медианным фильтром (в), модифицированными медианными фильтрами с шириной окна 25% (г), 50% (д) и 75% (е) и согласованным фильтром (ж)

Для случая $q_{вх}^2 = 2$:



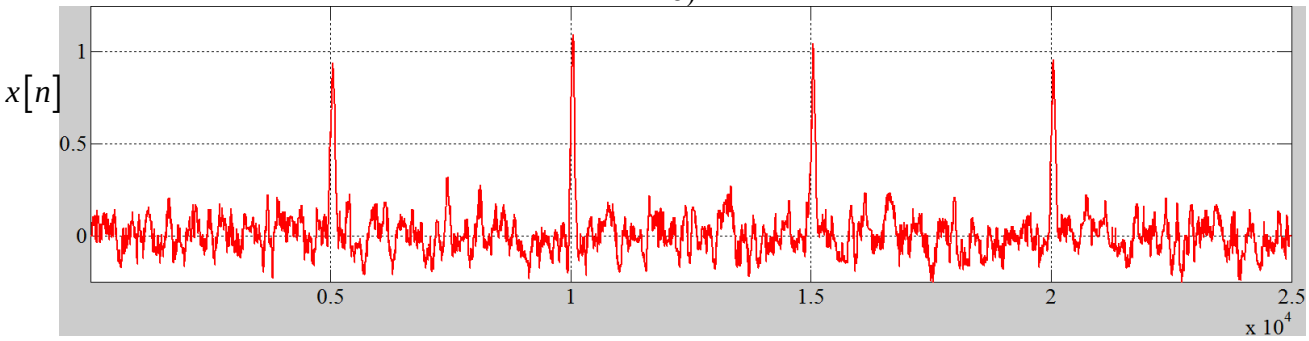
n

а)



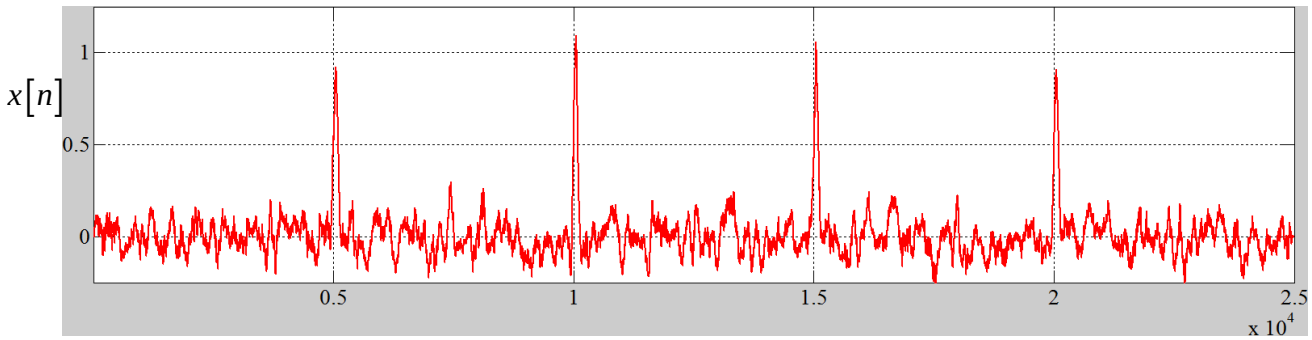
n

б)



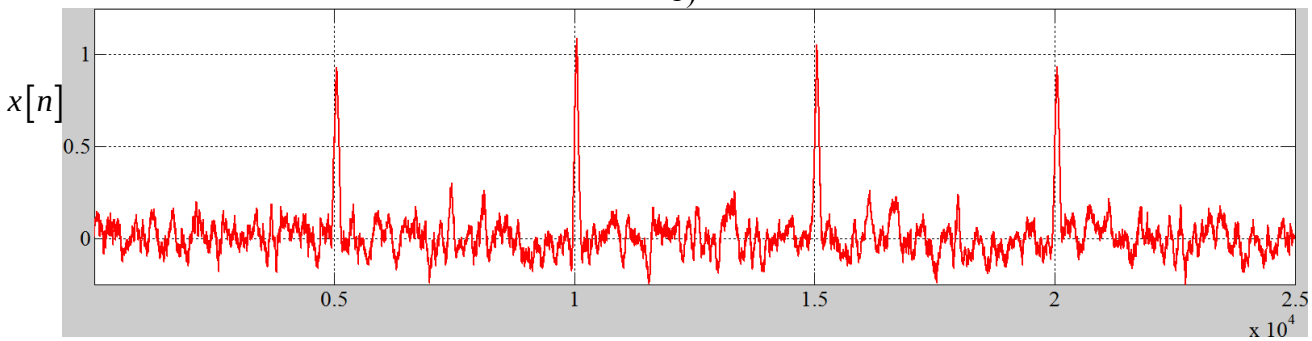
n

в)



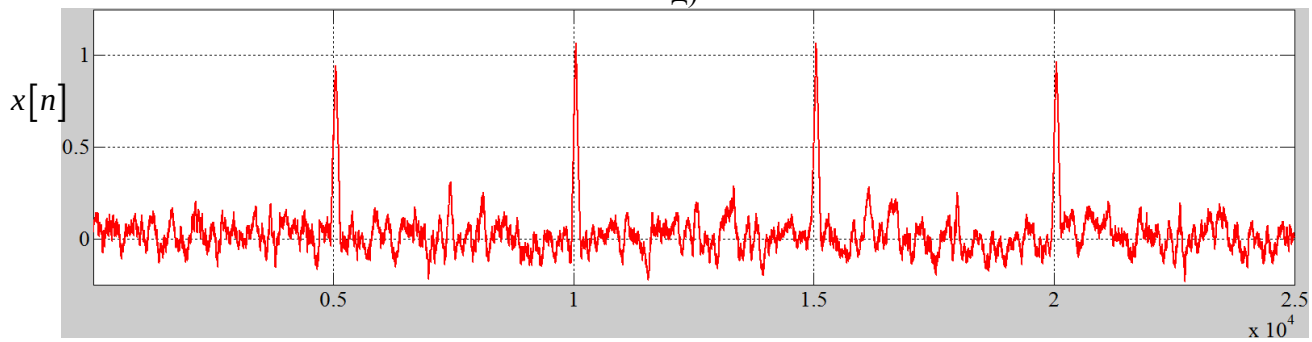
n

г)



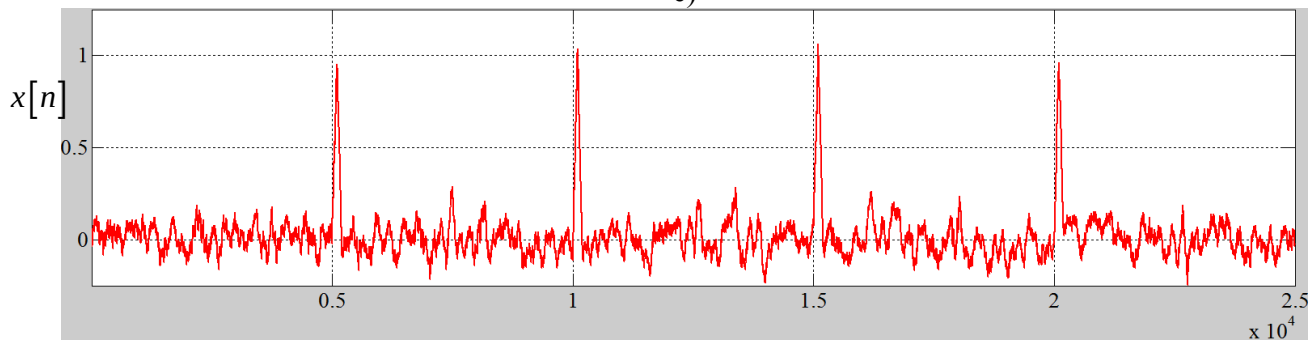
n

д)



n

е)



n

ж)

Рисунок 16. Временное представление входного сигнала без шума (а), с аддитивным гауссовским шумом для случая $q_{вх}^2 = 2$ (б), результата обработки медианным фильтром (в), модифицированными медианными фильтрами с шириной окна 25% (г), 50% (д) и 75% (е) и согласованным фильтром (ж)

Импульсы входного сигнала расположены в отсчетах 5000–5100, 10000–10100, 15000–15100 и 20000–20100. В остальных отсчетах мы ожидаем увидеть лишь шум, поэтому ОСШ входного сигнала с шумом:

$$q^2_{вх} = \frac{P_{сигнал}}{P_{шум}} = \frac{1}{\left(\sum_{n=0}^{5000} x^2[n] + \sum_{n=5101}^{10000} x^2[n] + \sum_{n=10101}^{15000} x^2[n] + \sum_{n=15101}^{20000} x^2[n] + \sum_{n=20101}^{25000} x^2[n] \right) / 24600}$$

После обработки медианным и модифицированными медианными фильтрами из-за шума ширина импульса удваивается, расширяясь в обе стороны, поэтому ОСШ на выходе медианного и модифицированных медианных фильтров:

$$q^2_{вых} = \frac{P_{сигнал}}{P_{шум}} = \frac{\left(\sum_{n=4950}^{5149} x^2[n] + \sum_{n=9950}^{10149} x^2[n] + \sum_{n=14950}^{15149} x^2[n] + \sum_{n=19950}^{20149} x^2[n] \right) / 800}{\left(\sum_{n=0}^{4949} x^2[n] + \sum_{n=5150}^{9949} x^2[n] + \sum_{n=10150}^{14949} x^2[n] + \sum_{n=15150}^{19949} x^2[n] + \sum_{n=20150}^{25000} x^2[n] \right) / 24200}$$

После обработки согласованным фильтром форма сигнала изменяется – сигнал на выходе имеет форму треугольника с вершиной в отсчете окончания входного импульса. Таким образом ширина импульса также удваивается, но расширение происходит только вправо. ОСШ на выходе согласованного фильтра:

$$q^2_{вых} = \frac{P_{сигнал}}{P_{шум}} = \frac{\left(\sum_{n=5000}^{5199} x^2[n] + \sum_{n=10000}^{10199} x^2[n] + \sum_{n=15000}^{15199} x^2[n] + \sum_{n=20000}^{20199} x^2[n] \right) / 800}{\left(\sum_{n=0}^{4999} x^2[n] + \sum_{n=5200}^{9999} x^2[n] + \sum_{n=10200}^{14999} x^2[n] + \sum_{n=15200}^{19999} x^2[n] + \sum_{n=20200}^{25000} x^2[n] \right) / 24200}$$

В результате получаем для случая $q^2_{вх} = 10$:

- После обработки медианным фильтром: $q^2_{вых} = 244$
- После обработки модифицированным медианным фильтром с окном 25%: $q^2_{вых} = 269,51$
- После обработки модифицированным медианным фильтром с окном 50%: $q^2_{вых} = 306,69$
- После обработки модифицированным медианным фильтром с окном 75%: $q^2_{вых} = 327,84$
- После обработки согласованным фильтром: $q^2_{вых} = 335,06$

Для случая $q_{вх}^2 = 2$:

- После обработки медианным фильтром: $q^2_{вых} = 46,38$
- После обработки модифицированным медианным фильтром с окном 25%: $q^2_{вых} = 50,75$
- После обработки модифицированным медианным фильтром с окном 50%: $q^2_{вых} = 59,19$
- После обработки модифицированным медианным фильтром с окном 75%: $q^2_{вых} = 65,56$
- После обработки согласованным фильтром: $q^2_{вых} = 68,36$

Выводы:

Применение как исследуемых нелинейных фильтров, так и согласованного фильтра к сигналу с аддитивным гауссовским шумом приводит к увеличению ОСШ более, чем на порядок. Сигнал на выходе согласованного фильтра имеет наибольшее ОСШ среди всех рассматриваемых фильтров, что является естественным, т.к. для сигнала с гауссовским шумом согласованный фильтр является оптимальным. Под оптимальностью понимается максимальное отношение сигнал/шум на выходе фильтра.

Рассчитаем, какой выигрыш в ОСШ дает согласованный фильтр по сравнению с медианным и модифицированными медианными фильтрами:

Таблица 1

Величина выигрыша согласованного фильтра в ОСШ по сравнению с нелинейными фильтрами

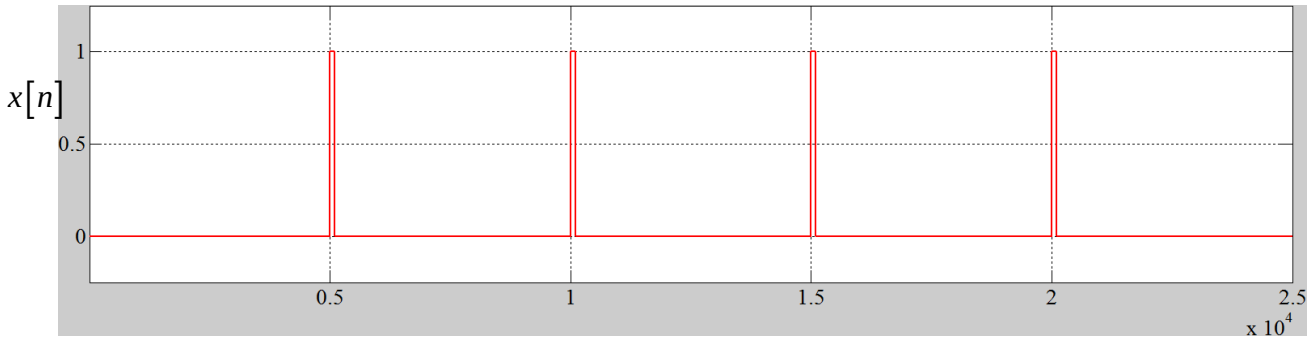
$q^2_{вх}$	Медианный	Модифицированный с окном 25%	Модифицированный с окном 50%	Модифицированный с окном 75%
10	37,3%	24,3%	9,3%	2,2%
2	47,4%	34,7%	15,5%	4,6%

Таким образом, согласованный фильтр дает существенно лучшие результаты по сравнению с медианным и модифицированным медианным фильтром с окном усреднения 25%, однако модифицированный медианный фильтр с окном усреднения 75% показывает близкие результаты.

Расчет отношения сигнал/шум при воздействии импульсного шума

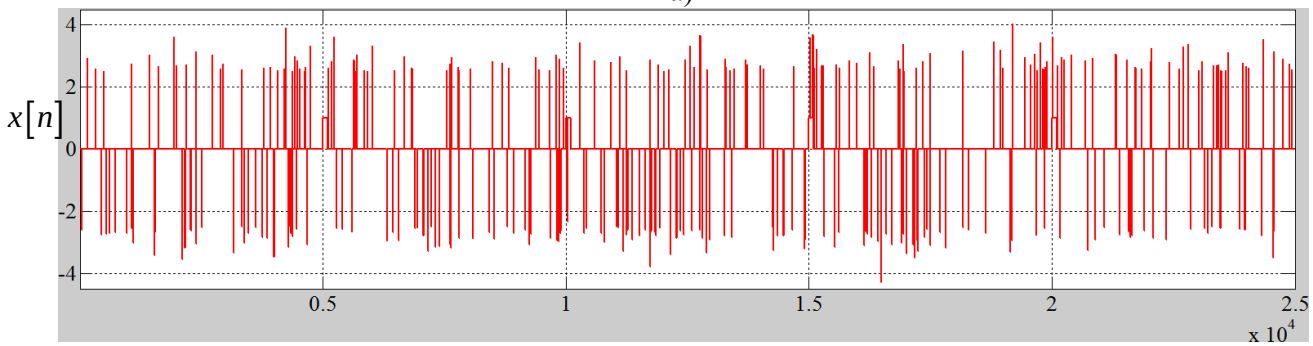
В качестве входного сигнала все так же принимаем последовательность прямоугольных импульсов длительностью 100 отсчетов. В качестве шума возьмем аддитивный импульсный шум. Исследование будем проводить для двух мощностей шума, обеспечивающих ОСШ на входе равным 10 и 2.

Построим временные зависимости входного сигнала, сигнала с шумом и сигналов, прошедших фильтрацию медианным, модифицированными медианными и согласованным фильтрами для случая $q_{вх}^2 = 10$ (см. приложение 3):



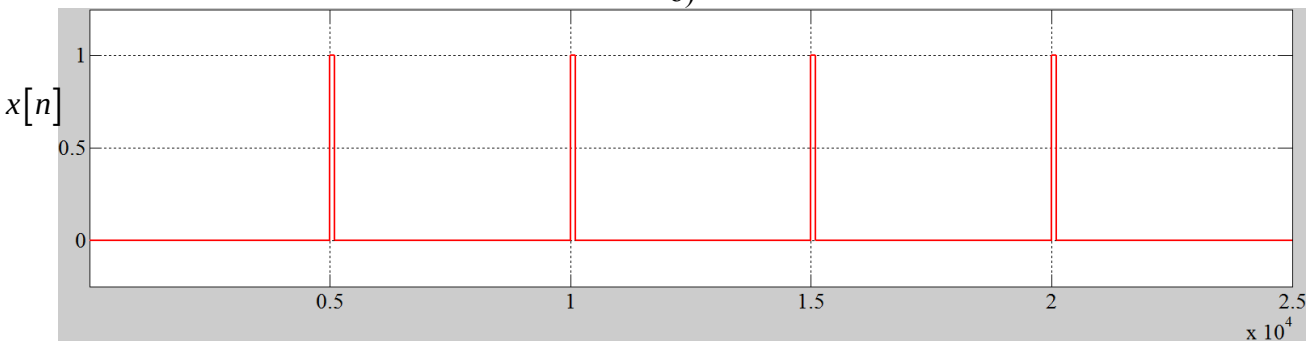
n

а)



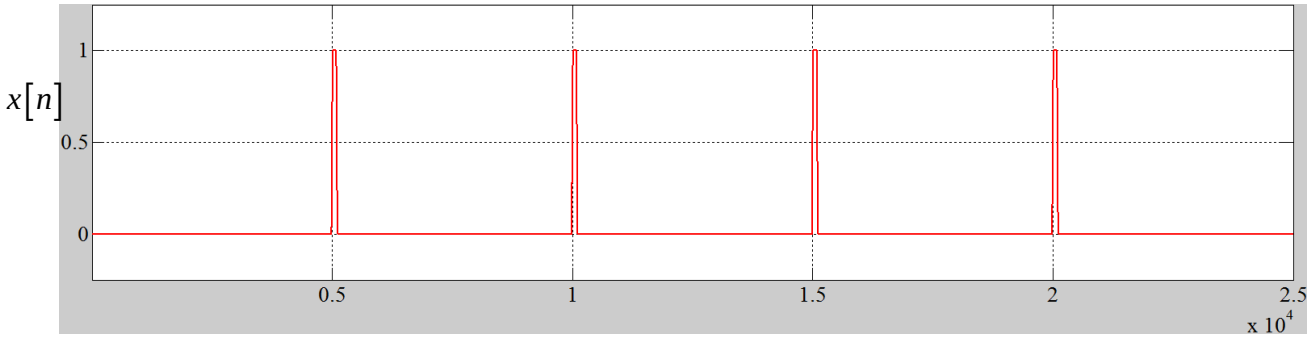
n

б)



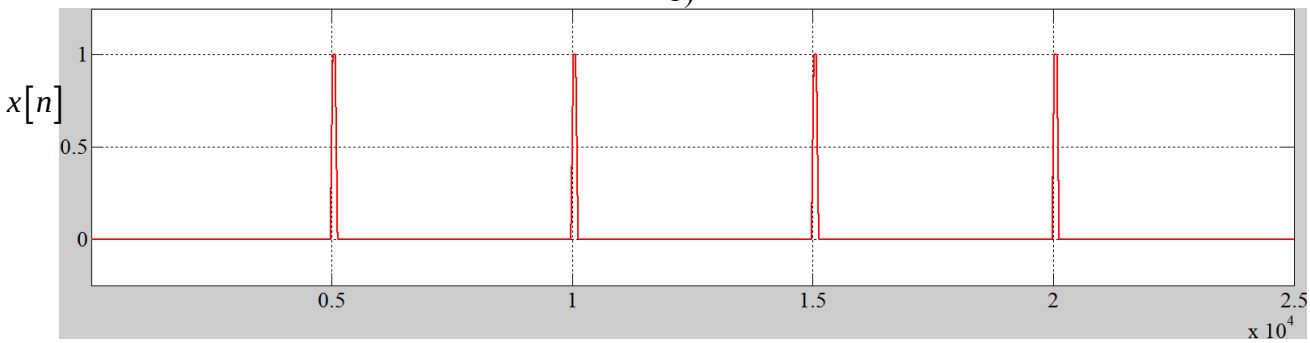
n

в)



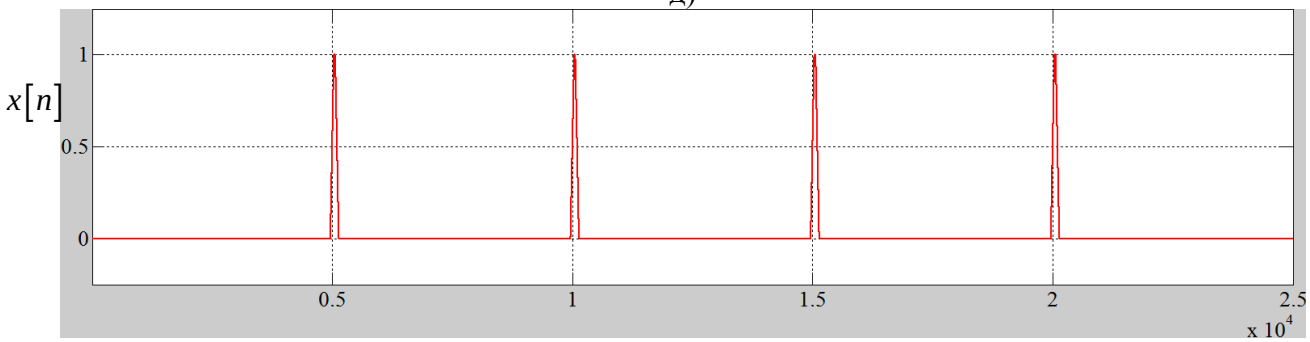
n

г)



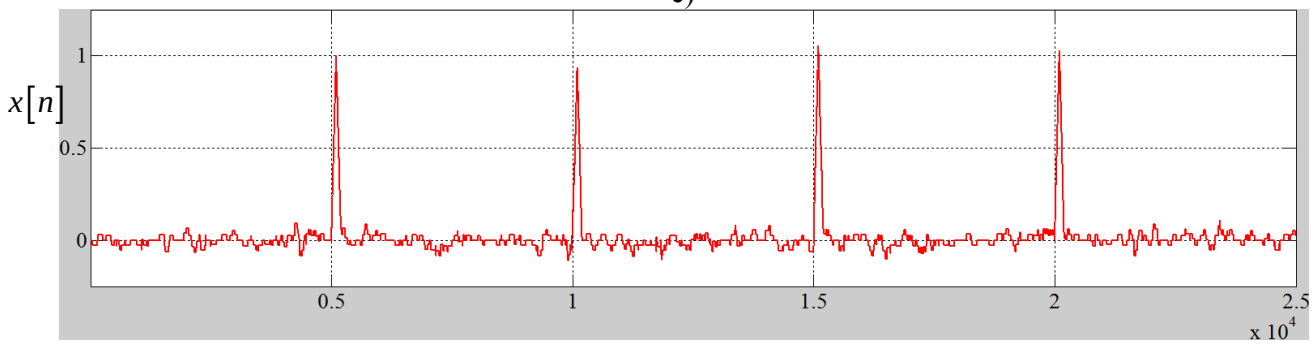
n

д)



n

е)

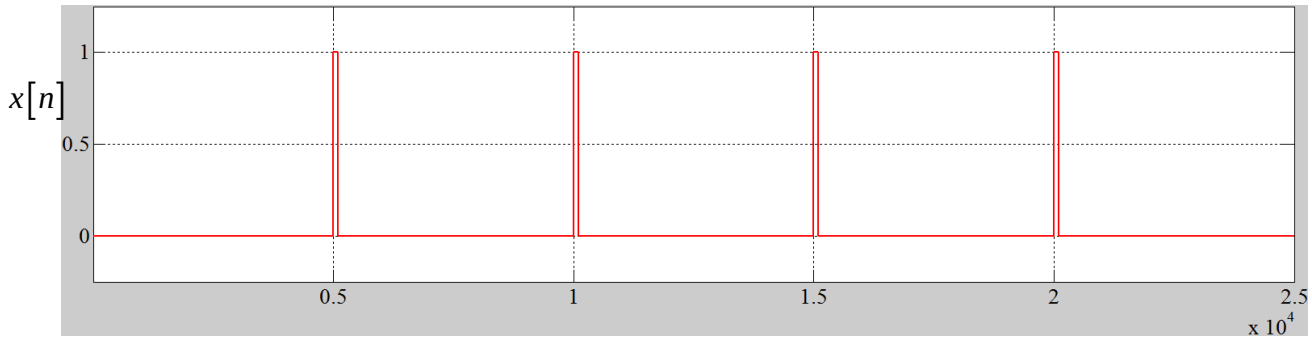


n

ж)

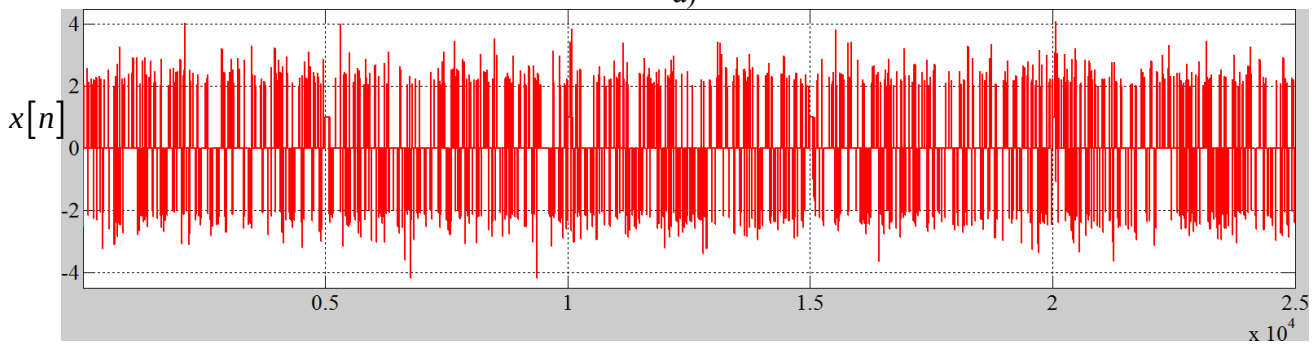
Рисунок 17. Временное представление входного сигнала без шума (а), с аддитивным импульсным шумом для случая $q_{вх}^2 = 10$ (б), результата обработки медианным фильтром (в), модифицированными медианными фильтрами с шириной окна 25% (г), 50% (д) и 75% (е) и согласованным фильтром (ж)

Для случая $q_{ex}^2 = 2$:



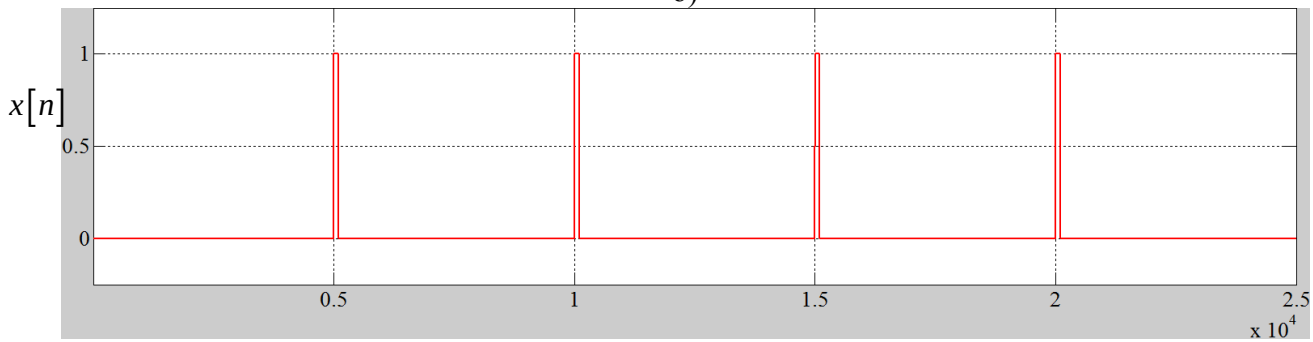
n

а)



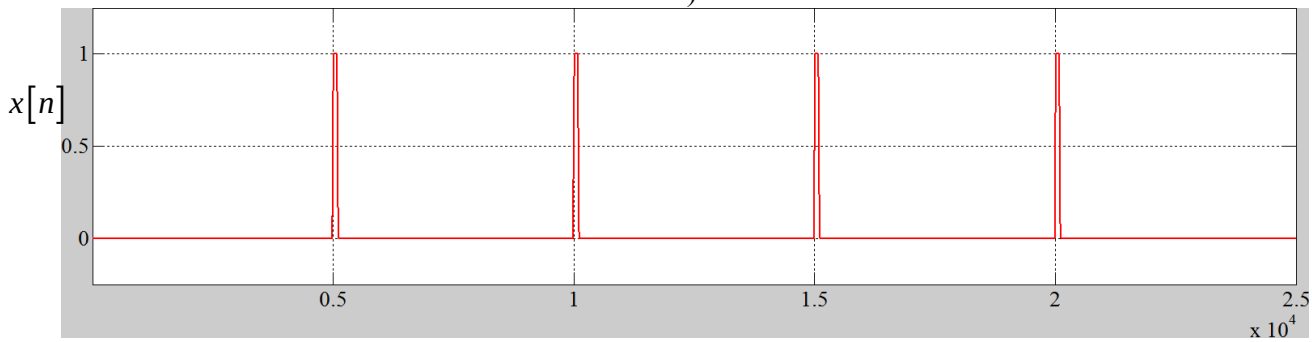
n

б)



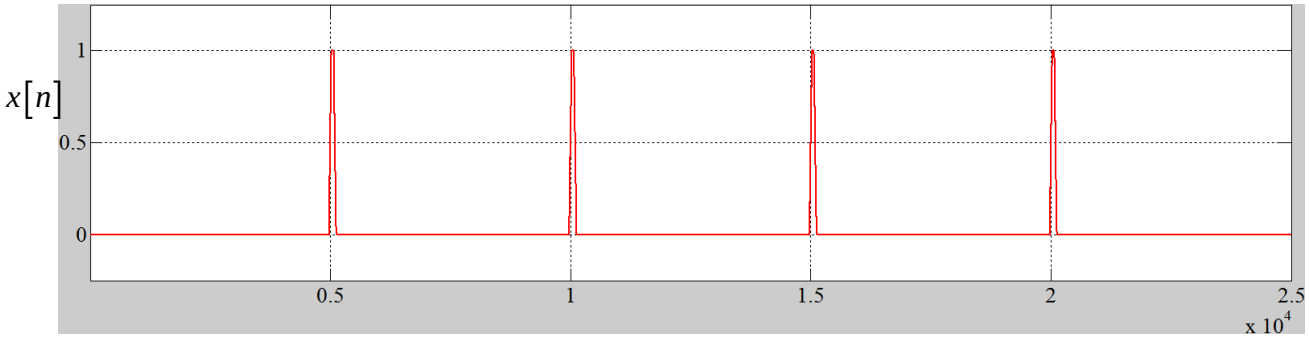
n

в)

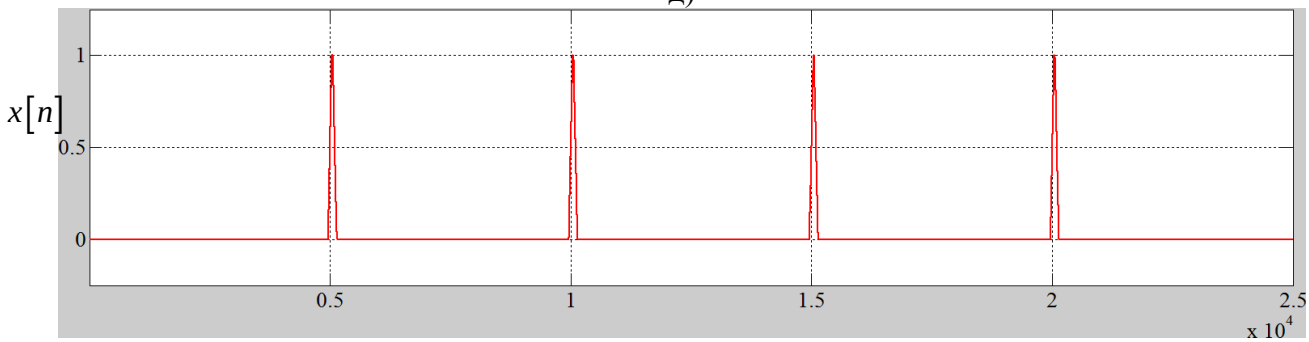


n

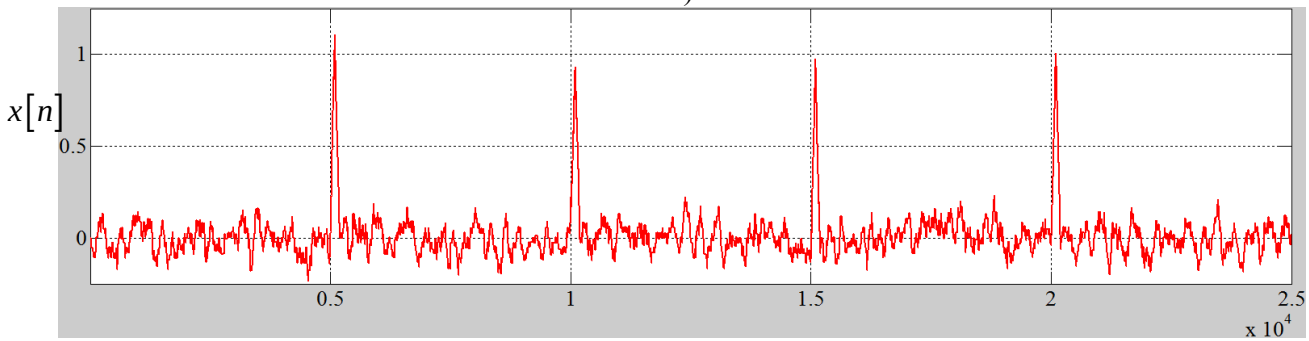
г)



д)



е)



ж)

Рисунок 18. Временное представление входного сигнала без шума (а), с аддитивным импульсным шумом для случая $q_{вх}^2 = 2$ (б), результата обработки медианным фильтром (в), модифицированными медианными фильтрами с шириной окна 25% (г), 50% (д) и 75% (е) и согласованным фильтром (ж)

В результате получаем для случая $q_{вх}^2 = 10$:

- После обработки медианным фильтром: $q^2_{вых} \to \infty$
- После обработки модифицированным медианным фильтром с окном 25%: $q^2_{вых} \to \infty$
- После обработки модифицированным медианным фильтром с окном 50%: $q^2_{вых} \to \infty$
- После обработки модифицированным медианным фильтром с окном 75%: $q^2_{вых} \to \infty$
- После обработки согласованным фильтром: $q^2_{вых} = 378,8$

Для случая $q_{вх}^2 = 2$:

- После обработки медианным фильтром: $q^2{}_{вых} \to \infty$
- После обработки модифицированным медианным фильтром с окном 25%: $q^2{}_{вых} \to \infty$
- После обработки модифицированным медианным фильтром с окном 50%: $q^2{}_{вых} \to \infty$
- После обработки модифицированным медианным фильтром с окном 75%: $q^2{}_{вых} \to \infty$
- После обработки согласованным фильтром: $q^2{}_{вых} = 63,81$

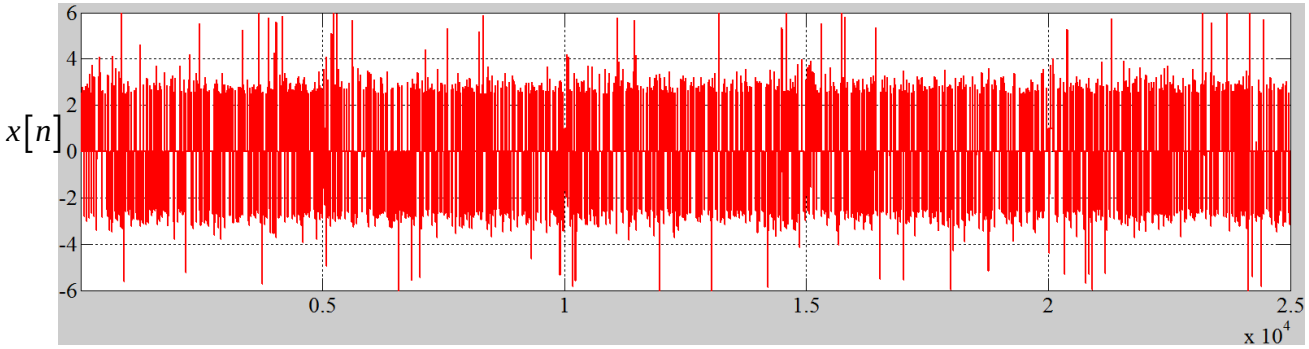
Выводы:

При заданных ОСШ на входе медианный и модифицированные медианные фильтры полностью подавляют импульсные шумы и тем самым оказываются намного эффективнее согласованного фильтра. Идеальное подавление происходит из-за достаточно большой ширины окна фильтров. Так для медианного фильтра с данной апертурой, чтобы шум повлиял на выходной сигнал, необходимо, чтобы в окно попадали одновременно 50 отсчетов, имеющих значение больше уровня сигнала без шума, или же 50 отсчетов, имеющих значение меньше уровня сигнала без шума. Для модифицированных фильтров с шириной окна усреднения 25%, 50% и 75% количество таких отсчетов должно составлять 38, 26 и 13 соответственно.

В данном опыте мощность шума увеличивается путем добавления новых импульсов. Другой способ увеличения мощности – увеличение их амплитуды. В таком случае с уменьшением ОСШ на входе будет ухудшаться ОСШ на выходе согласованного фильтра, а на выходе медианного и модифицированных медианных фильтров ОСШ будет неизменным, поскольку если отсчеты шума не оказываются медианой в случае медианного фильтра или не попадают в окно усреднения в случае модифицированных медианных фильтров, то они не будут влиять на выходной сигнал независимо от их амплитуды.

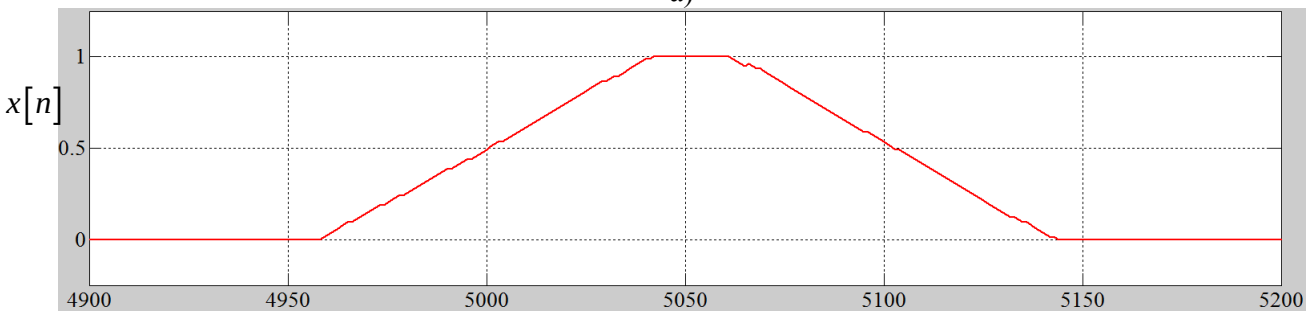
Определим, какое ОСШ на входе требуется, чтобы подавление импульсных шумов модифицированными медианными фильтрами не было идеальным.

При $q_{вх}^2 = 1,25$:



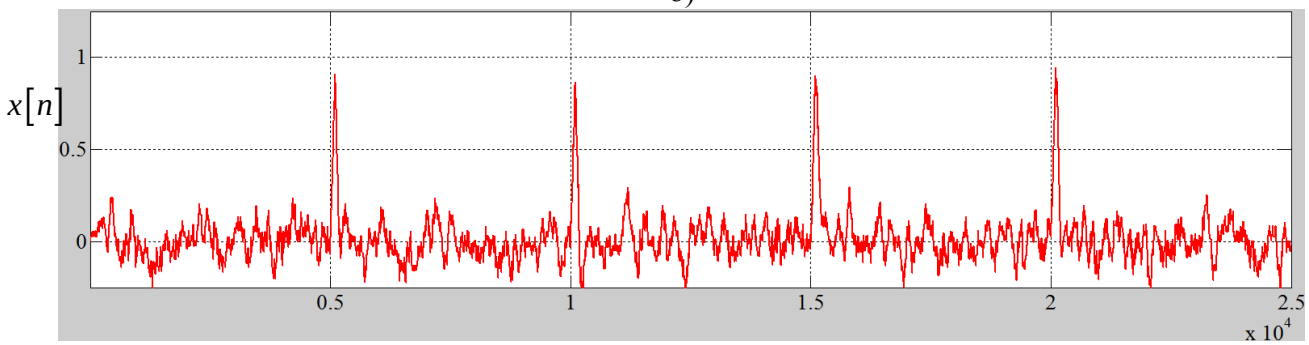
n

а)



n

б)



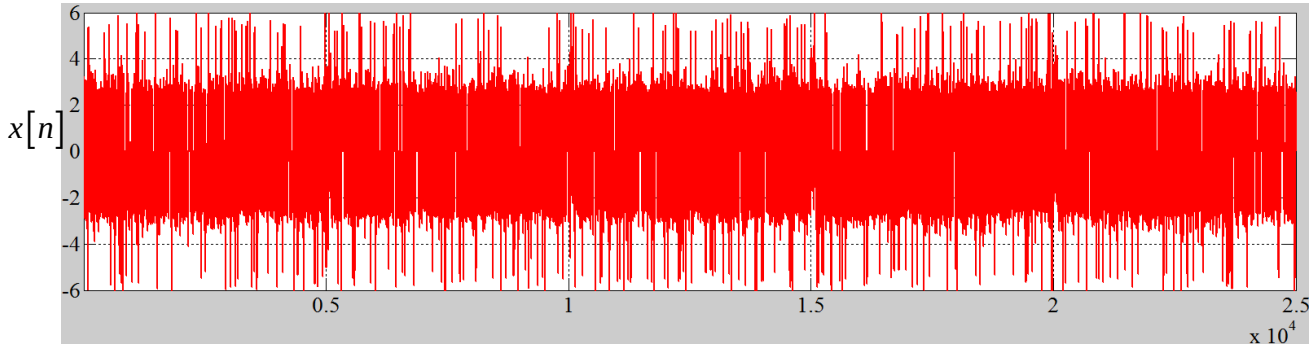
n

в)

Рисунок 19. Временное представление входного сигнала с аддитивным импульсным шумом для случая $q_{вх}^2 = 1,25$ (а), результата обработки модифицированным медианным фильтром с шириной окна 75% (б) и согласованным фильтром (в)

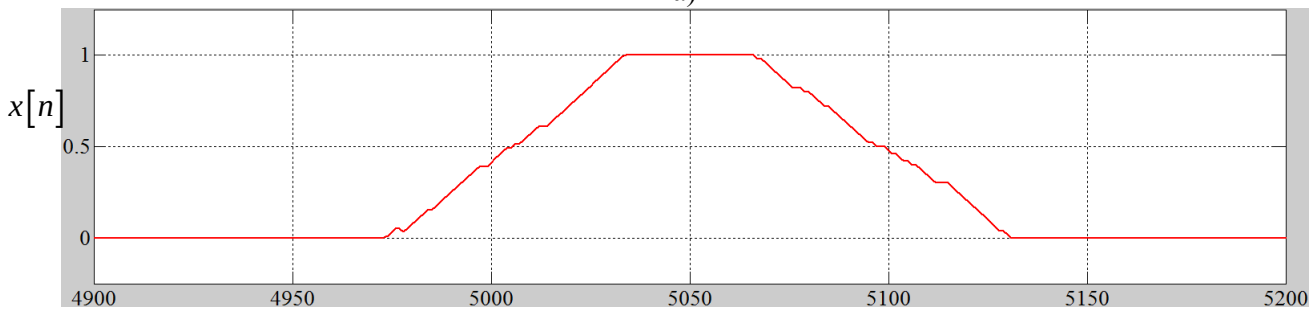
- После обработки медианным фильтром: $q^2_{вых} \to \infty$
- После обработки модифицированным медианным фильтром с окном 25%: $q^2_{вых} \to \infty$
- После обработки модифицированным медианным фильтром с окном 50%: $q^2_{вых} \to \infty$
- После обработки модифицированным медианным фильтром с окном 75%: $q^2_{вых} = 5,72 \cdot 10^7$
- После обработки согласованным фильтром: $q^2_{вых} = 39,75$

При $q_{вх}^2 = 0,46$:



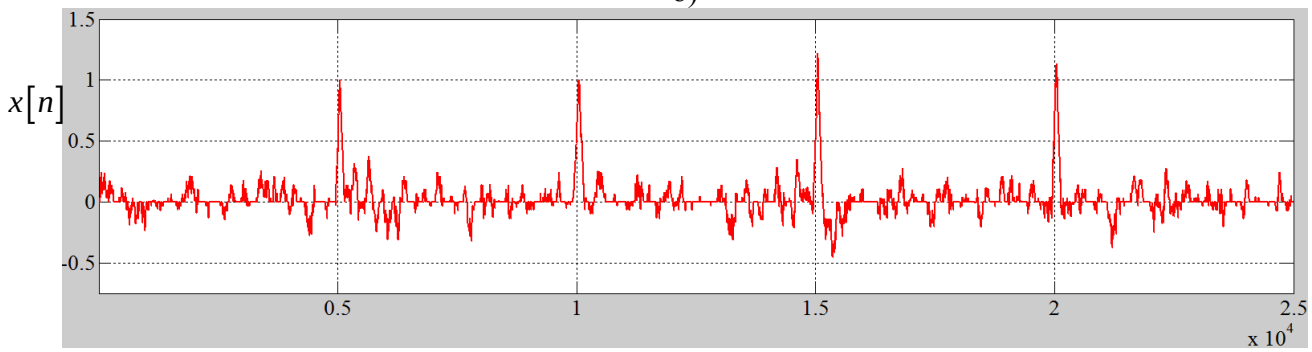
n

а)



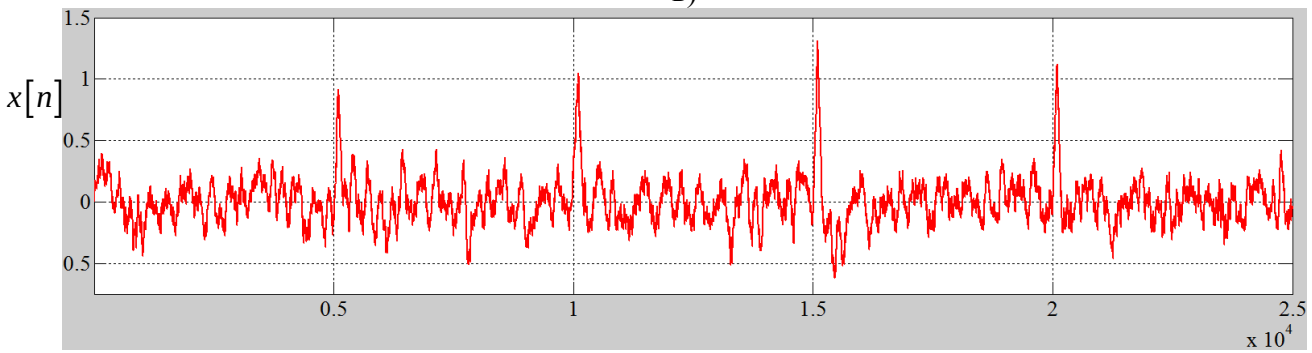
n

б)



n

в)



n

г)

Рисунок 20. Временное представление входного сигнала с аддитивным импульсным шумом для случая $q_{вх}^2 = 0,46$ (а), результата обработки модифицированными медианными фильтрами с шириной окна 50% (б) и 75% (в) и согласованным фильтром (г)

- После обработки медианным фильтром: $q^2_{вых} \to \infty$
- После обработки модифицированным медианным фильтром с окном 25%: $q^2_{вых} \to \infty$
- После обработки модифицированным медианным фильтром с окном 50%: $q^2_{вых} \to 717321$
- После обработки модифицированным медианным фильтром с окном 75%: $q^2_{вых} = 56{,}19$
- После обработки согласованным фильтром: $q^2_{вых} = 16{,}41$

Расчет отношения сигнал/шум при воздействии гауссовского и импульсного шума одновременно

В реальных ситуациях на сигнал могут воздействовать сразу несколько видов шумов. Рассмотрим случай, когда сигнал в линии передачи оказался поврежден одновременно и импульсным, и гауссовским шумом, и сравним эффективность работы фильтров в зависимости от отношения мощностей этих шумов.

Входной сигнал все так же представлен периодической последовательностью прямоугольных импульсов длительностью 100 отсчетов. Исследование будем проводить при ОСШ на входе равном 10 и 2.

Поскольку шум – это случайный процесс, его влияние на сигнал изменяется от опыта к опыту, что приводит к различным результатам при одних и тех же входных данных. Чтобы получить более достоверные данные и построить более достоверные зависимости, усредним результаты 100 вычислений.

В результате выполнения программы, представленной в приложении 4 для случая $q_{вх}^2 = 10$:

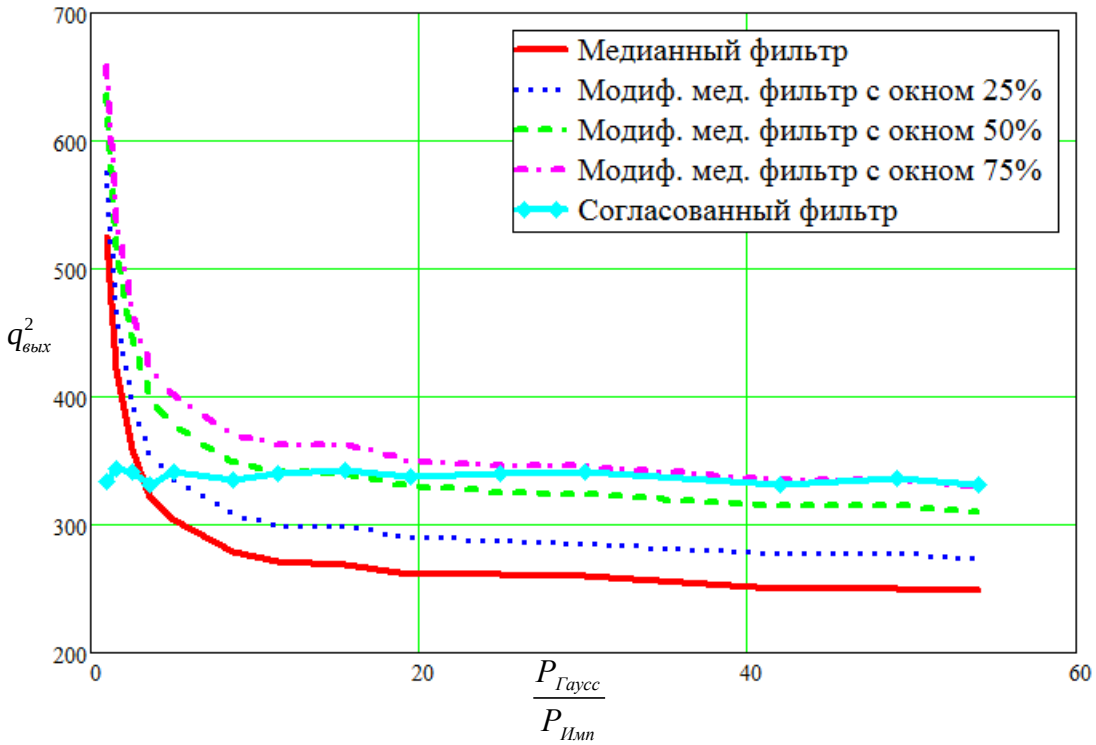


Рисунок 21. Зависимость ОСШ на выходе фильтров от отношения мощностей гауссовского и импульсного шума на входе при $q_{вх}^2 = 10$

Для случая $q_{вх}^2 = 2$:

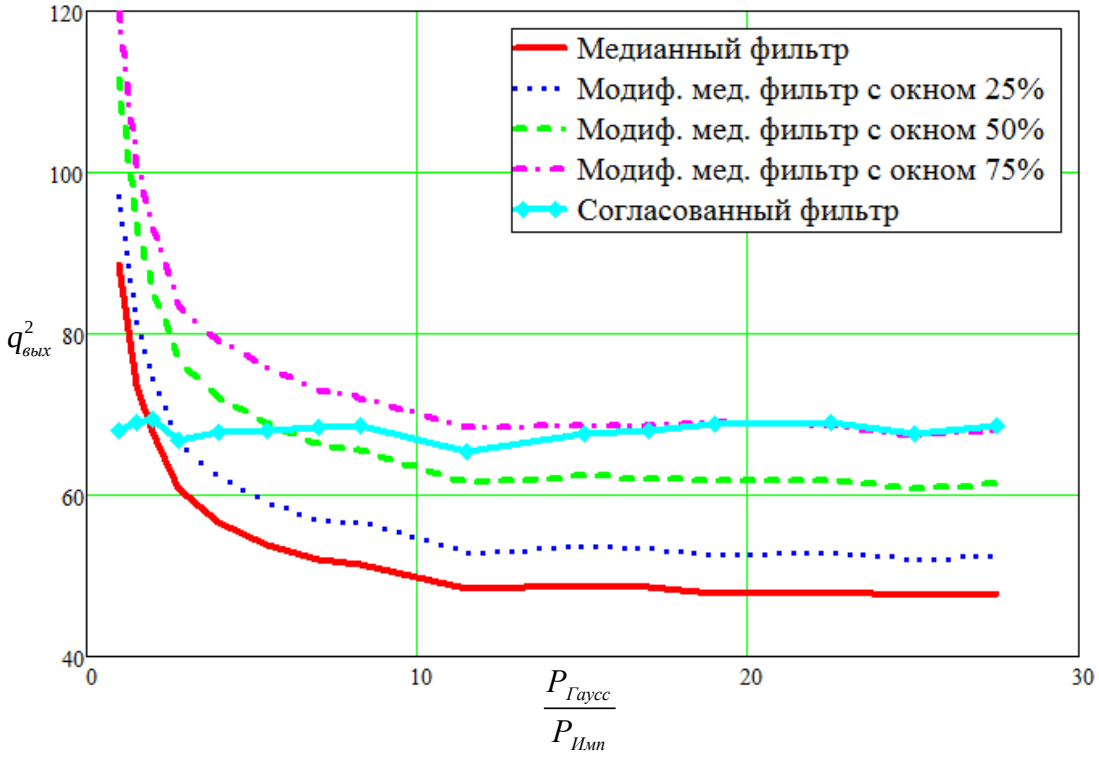


Рисунок 22. Зависимость ОСШ на выходе фильтров от отношения мощностей гауссовского и импульсного шума на входе при $q_{вх}^2 = 2$

Выводы:

Основываясь на результатах данного опыта, можно сделать вывод, что медианный и модифицированные медианные фильтры оказываются эффективнее согласованного фильтра при подавлении не только чисто импульсных шумов, но и смеси импульсного шума с гауссовским до определенного отношения их мощностей.

Таблица 2

Отношения $\frac{P_{Гаусс}}{P_{Имп}}$, до которых нелинейные фильтры эффективнее согласованного

	Медианный	Модифицированный с окном 25%	Модифицированный с окном 50%	Модифицированный с окном 75%
$q_{вх}^2 = 10$	≈ 3	4–5	12–13	45–47
$q_{вх}^2 = 2$	$\approx 1,7$	2,5–3	6–6,5	20–21

При увеличении уровня шума на входе фильтра диапазон отношений $\frac{P_{Гаусс}}{P_{Имп}}$, при которых медианный и модифицированные медианные фильтры оказываются эффективнее согласованного, сужается.

3. Раздел охраны труда

В данной выпускной квалификационной работе рассматриваются вопросы исследования цифровых фильтров. Данная работа выполняется с помощью персонального компьютера, правила и требования при работе с которым регламентированы СанПиН 2.2.2/2.4.1340-03 "Гигиенические требования к персональным электронно-вычислительным машинам и организации работы". С точки зрения охраны труда при работе с персональным компьютером должны соблюдаться следующие условия:

- Достаточный уровень освещения
- Оптимальные параметры микроклимата в рабочей зоне
- Соблюдение требований электробезопасности
- Допустимый уровень шума и вибрации
- Организованный режим работы и отдыха

Освещение

Освещению рабочего места уделяется повышенное внимание, т.к. его уровень серьезно влияет на интенсивность зрительного утомления. Правильно организованное освещение создает благоприятные условия труда, повышает работоспособность и производительность труда. Требуемый уровень освещенности в первую очередь определяется характером выполняемых работ: чем меньше размер фиксируемой детали, тем выше должен быть уровень освещенности. При этом величина освещенности должна быть постоянной на всем рабочем месте, поскольку при переводе взгляда с яркой на более темную поверхность или наоборот глаз адаптируется. Частая переадаптация приводит к быстрому утомлению. На рабочей поверхности должны отсутствовать резкие тени, а также прямая и отраженная блескость, т.е. повышенная яркость светящихся поверхностей, вызывающая ослепленность.

Для освещения рабочих мест используется естественное освещение, создаваемое прямыми солнечными лучами и рассеянным светом небосвода. Освещенность на рабочей поверхности должна быть 300 - 500 лк. При недостатке естественного освещения используют искусственное освещение, создаваемое искусственными источниками света.

В качестве искусственных источников света могут быть использованы лампы накаливания, газоразрядные и светодиодные лампы. Наилучшим выбором является применение светодиодных светильников. По сравнению с лампами накаливания они имеют гораздо больший срок службы, низкую потребляемую мощность и низкий уровень

нагрева. В отличие от газоразрядных ламп светодиодные лампы имеют низкий уровень пульсаций и не содержат вредных химических элементов.

Микроклимат

Согласно СанПин 2.2.4.548-96 "Гигиенические требования к микроклимату производственных помещений" на рабочих местах должны обеспечиваться оптимальные параметры микроклимата: температура окружающего воздуха и поверхностей, относительная влажность, скорость движения воздуха и интенсивность теплового облучения.

Оптимальные условия микроклимата обеспечивают ощущение теплового комфорта во время работы, не вызывают отклонений здоровья и создают предпосылки для высокой производительности труда. Для категории тяжести работ 1а (низкий уровень энергозатрат) в холодный период года температура воздуха должна быть 22–24^{o}C, температура поверхностей 21–25^{o}C, а в теплый период года 23–25^{o}C и 22–26^{o}C соответственно. Относительная влажность должна составлять 40-60%, скорость движения воздуха — 0,1 м/с. Оптимальных значения микроклимата поддерживаются за счет систем отопления, кондиционирования и вентилирования воздуха. Для повышения влажности воздуха применяются увлажнители воздуха.

Электробезопасность

При работе с любым устройством, питаемым напряжением свыше 36В, существует опасность поражения электрическим током. Это может произойти из-за различных причин: перегрузка, некачественная изоляция, прикосновение к токоведущим частям и т.д. Чтобы минимизировать вероятность поражения электрическим током, необходимо соблюдать требования, установленные "Правилами эксплуатации электроустановок потребителей" и "Правилами техники безопасности при эксплуатации электроустановок потребителей".

Для исключения поражения электрическим током запрещается: часто включать и выключать компьютер без необходимости, прикасаться к тыльной стороне блоков компьютера, работать на устройствах, имеющих нарушения изоляции проводов, признаки наличия напряжения на корпусе. При работе за компьютером нужно следить, чтобы руки были сухие, необходимо предотвращать попадание любой жидкости на корпус и периферийные устройства ПК. Нельзя класть на компьютерную технику посторонние предметы. Для избежания короткого замыкания и получения электротравм рабочие места должны иметь защитное заземление.

Шум и вибрация

При выполнении работ с использованием персональных компьютеров уровни шума и вибрации на рабочем месте не должны превышать предельно допустимых значений. Согласно СН 2.2.4/2.1.8.562-96 "Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки" в помещениях, оборудованных ПК допустимый уровень шума составляет 50 дБ. В результате многочисленных исследований было определено, что такой уровень шума не вызывает стойких физиологических отклонений. При уровне шума свыше 70 дБ в организме человека появляются различные отклонения, выражающиеся в нарушении работы ЦНС, снижении слуха, остроты зрения.

Допустимый уровень виброскорости на частотах 2, 4, 8, 16, 31,5, 63 Гц соответственно составляет 79, 73, 67, 67, 67, 67 дБ. Вибрация, превышающая предельно допустимые уровни, приводит к повреждениям органов и тканей, оказывает негативное влияние на ЦНС, органы слуха и зрения, приводит к быстрой утомляемости.

Режим работы и отдыха

При работе за компьютером человек продолжительное время находится в одной позе. Вкупе с другими факторами, такими как электромагнитное излучение, нагрузка на глаза и т.д., это оказывает негативное воздействие на здоровье. Поэтому необходимо организовывать перерывы при работе за компьютером.

Для студентов старших курсов допустимое время работы за персональным компьютером составляет 2 часа, с обязательным соблюдением перерыва длительностью 15-20 минут и выполнением общих физических упражнений и гимнастики для глаз.

Кроме того необходимо соблюдать следующие правила:

- Необходимо соблюдать расстояние от экрана до глаз не менее 70 см;
- Следить за осанкой;
- Следует избегать большой контрастности между яркостью экрана и окружающего пространства. Нельзя работать с компьютером в темном или полутемном помещении;
- Обязательно проводить упражнения для глаз каждые 20-25 минут работы;

4. Организационно-экономический раздел

Разработка программы – это не только непосредственное написание ее кода. Согласно ГОСТ ЕСПД 19.102-77 "Единая система программной документации. Стадии разработки" при разработке программ и программной документации выделяются стадии разработки:

- техническое задание;
- эскизный проект;
- технический проект;
- рабочий проект;
- внедрение.

Каждая стадия включает в себя несколько этапов. Так, например, на стадии разработки технического задания необходимо определить исходные данные, изучить техническую сторону вопроса, выбрать язык, на котором будет написана программа.

Для разработки программы необходим персонал. Разрабатываемая программа не отличается особенной сложностью или объемностью, среда разработки используется уже созданная и настроенная, поэтому исходя из соображений практической целесообразности принимаем количество задействованных работников минимальным: руководитель проекта и непосредственный исполнитель – инженер-программист.

Представим перечень этапов работы с указанием исполнителей и примерной продолжительности в виде таблицы:

Таблица 3

Перечень этапов работы при разработке программы

Стадия разработки	Этап работы	Исполнители	Продолжительность работы, дней
Техническое задание	Постановка задачи	Руководитель проекта	2
	Определение и анализ исходных данных, подбор и изучение технической литературы	Инженер-программист	
	Предварительный выбор методов выполнения поставленной задачи	Инженер-программист	
	Определение этапов, сроков разработки программы и документации	Инженер-программист, руководитель проекта	1
	Выбор языка программирования	Инженер-программист	
	Согласование и утверждение технического задания	Руководитель проекта	

Эскизный проект	Разработка структуры входных и выходных данных	Инженер-программист	2
	Уточнение методов решения задачи	Инженер-программист	
	Согласование и утверждение эскизного проекта	Руководитель проекта	
Технический проект	Разработка алгоритма решения задачи	Инженер-программист, руководитель проекта	3
	Разработка структуры программы	Инженер-программист	
	Согласование и утверждение технического проекта	Руководитель проекта	1
Рабочий проект	Программирование	Инженер-программист	5
	Тестирование и отладка	Инженер-программист	2
	Разработка программной документации	Инженер-программист, руководитель проекта	3
Внедрение	Подготовка и передача программы и технической документации для сдачи	Инженер-программист, руководитель проекта	1

Рассчитаем примерные затраты на разработку программы.

Согласно данным статистики сайтов вакансий средняя месячная заработная плата инженера-программиста C++ в Новосибирске на апрель-май 2017 года составляет 55000 руб., а руководителя IT-проекта – 70000. Общее время на разработку программы составляет 20 дней. В мае 2017 года 20 рабочих дней, соответственно, общие затраты на заработную плату работников составляют: $55000 \cdot \frac{20}{20} + 70000 \cdot \frac{20}{20} = 125000\ руб$.

Кроме того, необходимо произвести отчисления, зависящие от величины заработной платы:

- Пенсионный фонд – 22%
- Фонд социального страхования – 2,9%
- Федеральный фонд обязательного социального страхования – 5,1%
- Взносы по травматизму – 0,2–8,5%

Величина взносов по травматизму зависит от класса профессионального риска. Допустим, что организации присвоен 1 класс профессионального риска, тогда размер отчислений будет составлять 0,2%.

Общая величина отчислений составляет:

$125000 \cdot (0,22 + 0,029 + 0,051 + 0,002) = 37750\ руб$

Для написания кода программы требуется персональный компьютер. Предположим, что он уже имеется. Среда разработки Microsoft Visual Studio Express распространяется бесплатно.

Примем, что потребляемая мощность персонального компьютера 250Вт, а освещения 100Вт. Общее время работы при 8-часовом рабочем дне:

$20 \cdot 8 = 160\ ч$

Расход электроэнергии на питание компьютера и на освещение за все время работы:

$(250 + 100) \cdot 160 = 56\ кВт \cdot ч$

Стоимость электроэнергии в Новосибирске на апрель-май 2017 года составляет 2,42 руб/кВт*ч по одноставочному тарифу. Тогда затраты на электроэнергию составят:

$56 \cdot 2,42 \approx 136\ руб$

Прочие расходы обычно принимаются в процентном отношении к заработной плате исполнителей. В настоящем расчете величину прочих расходов можно взять в размере 15% от величины основной заработной платы исполнителей:

$0,6 \cdot 125000 = 18750\ руб$

В итоге ориентировочные затраты на производство программы составляют:

$125000 + 37750 + 136 + 18750 = 181636\ руб$

Заключение

В ходе данной выпускной квалификационной работы были рассмотрены некоторые элементы теории нелинейной фильтрации сигналов, а конкретно принципы работы фильтров, основанных на ранговой статистике: медианного фильтра и модифицированного медианного фильтра, известного также как усеченный фильтр среднего. В результате выполнения экспериментальной части работы были построены зависимости коэффициента нелинейных искажений и коэффициента передачи исследуемых фильтров от частоты входного синусоидального сигнала. Для нелинейных фильтров и линейного согласованного фильтра проведено сравнение эффективности подавления гауссовского и импульсного шумов различной мощности, а также их смеси, при приеме сигнала, представленного периодической последовательностью прямоугольных импульсов. Для двух значений ОСШ на входе фильтров определены диапазоны отношения мощностей гауссовского и импульсного шумов, при которых рассмотренные нелинейные фильтры оказываются более эффективными, чем согласованный фильтр. Кроме того, в данной работе поднимаются организационно-экономические вопросы и вопросы охраны труда.

Список литературы

1. Айфичер Э.С., Джервис Б.У. Цифровая обработка сигналов: практический подход, 2-е издание.: пер. с англ. – М.: Издательский дом "Вильямс", 2008. – 992 с.
2. Pearson R.K., Gabbouj M. Nonlinear digital filtering with Python. Taylor&Francis Group, 2016. – 286 p.
3. Смит С. Цифровая обработка сигналов. Практическое руководство для инженеров и научных работников. Пер. с англ. А.Ю. Линовича, С.В. Витязева, И.С. Гусинского. – М.: Додэка-XXI, 2012. – 720 с.
4. Pitas I., Venetsanopoulos A.N. Nonlinear digital filters: principles and applications. Springer Science + Business Media New York, 1999. – 392 p.
5. Франка П. C++: учебный курс. – СПб.: Питер, 2003. – 521с.: ил.
6. Степанов В.П., Экономический раздел дипломного проекта: Методические указания. - М.: МГУПИ, 2012. - 54 с.
7. Гигиенические требования к персональным электронно-вычислительным машинам и организации работы: СанПиН 2.2.2/2.4.1340-03 от 13.06.2003 г.

Код программы определения КНИ и Кр

```
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <fstream>
using namespace std;

void Everything(double fs); //Прототип функции формирования сигнала,
фильтрации и нахождения спектра
void Kn(double *AFC, double fs); //Прототип функции нахождения КНИ
void K(double *AFC, double *AFCOriginal, double fs); //Прототип функции
нахождения Кр

void main()
{
setlocale(LC_ALL,"rus_rus.1251");
srand(time(NULL));

//Расчет для частоты 0,001
cout<<"Коэффициенты нелинейных искажений для исходного сигнала и после
обработки медианным фильтром, модифицированными медианными фильтрами с длиной
окна 25%, 50% и 75% соответственно:"<<endl;
cout<<"Для частоты 0,001: "<<endl;
Everything(0.001);
rename("BezSh.txt", "BezSh1.txt");
rename("Med.txt", "Med1.txt");
rename("Mod25pr.txt", "Mod25pr1.txt");
rename("Mod50pr.txt", "Mod50pr1.txt");
rename("Mod75pr.txt", "Mod75pr1.txt");
rename("AFCBezSh.txt", "AFCBezSh1.txt");
rename("AFCMed.txt", "AFCMed1.txt");
rename("AFCMod25pr.txt", "AFCMod25pr1.txt");
rename("AFCMod50pr.txt", "AFCMod50pr1.txt");
rename("AFCMod75pr.txt", "AFCMod75pr1.txt");
cout<<endl;

//Расчет для частоты 0,002
cout<<"Для частоты 0,002: "<<endl;
Everything(0.002);
rename("BezSh.txt", "BezSh2.txt");
rename("Med.txt", "Med2.txt");
rename("Mod25pr.txt", "Mod25pr2.txt");
rename("Mod50pr.txt", "Mod50pr2.txt");
rename("Mod75pr.txt", "Mod75pr2.txt");
rename("AFCBezSh.txt", "AFCBezSh2.txt");
rename("AFCMed.txt", "AFCMed2.txt");
rename("AFCMod25pr.txt", "AFCMod25pr2.txt");
rename("AFCMod50pr.txt", "AFCMod50pr2.txt");
rename("AFCMod75pr.txt", "AFCMod75pr2.txt");
cout<<endl;

//Расчет для частоты 0,003
cout<<"Для частоты 0,003: "<<endl;
Everything(0.003);
rename("BezSh.txt", "BezSh3.txt");
rename("Med.txt", "Med3.txt");
rename("Mod25pr.txt", "Mod25pr3.txt");
rename("Mod50pr.txt", "Mod50pr3.txt");
```

```
rename("Mod75pr.txt", "Mod75pr3.txt");
rename("AFCBezSh.txt", "AFCBezSh3.txt");
rename("AFCMed.txt", "AFCMed3.txt");
rename("AFCMod25pr.txt", "AFCMod25pr3.txt");
rename("AFCMod50pr.txt", "AFCMod50pr3.txt");
rename("AFCMod75pr.txt", "AFCMod75pr3.txt");
cout<<endl;

//Расчет для частоты 0,004
cout<<"Для частоты 0,004: "<<endl;
Everything(0.004);
rename("BezSh.txt", "BezSh4.txt");
rename("Med.txt", "Med4.txt");
rename("Mod25pr.txt", "Mod25pr4.txt");
rename("Mod50pr.txt", "Mod50pr4.txt");
rename("Mod75pr.txt", "Mod75pr4.txt");
rename("AFCBezSh.txt", "AFCBezSh4.txt");
rename("AFCMed.txt", "AFCMed4.txt");
rename("AFCMod25pr.txt", "AFCMod25pr4.txt");
rename("AFCMod50pr.txt", "AFCMod50pr4.txt");
rename("AFCMod75pr.txt", "AFCMod75pr4.txt");
cout<<endl;

//Расчет для частоты 0,005
cout<<"Для частоты 0,005: "<<endl;
Everything(0.005);
rename("BezSh.txt", "BezSh5.txt");
rename("Med.txt", "Med5.txt");
rename("Mod25pr.txt", "Mod25pr5.txt");
rename("Mod50pr.txt", "Mod50pr5.txt");
rename("Mod75pr.txt", "Mod75pr5.txt");
rename("AFCBezSh.txt", "AFCBezSh5.txt");
rename("AFCMed.txt", "AFCMed5.txt");
rename("AFCMod25pr.txt", "AFCMod25pr5.txt");
rename("AFCMod50pr.txt", "AFCMod50pr5.txt");
rename("AFCMod75pr.txt", "AFCMod75pr5.txt");
cout<<endl;

//Расчет для частоты 0,006
cout<<"Для частоты 0,006: "<<endl;
Everything(0.006);
rename("BezSh.txt", "BezSh6.txt");
rename("Med.txt", "Med6.txt");
rename("Mod25pr.txt", "Mod25pr6.txt");
rename("Mod50pr.txt", "Mod50pr6.txt");
rename("Mod75pr.txt", "Mod75pr6.txt");
rename("AFCBezSh.txt", "AFCBezSh6.txt");
rename("AFCMed.txt", "AFCMed6.txt");
rename("AFCMod25pr.txt", "AFCMod25pr6.txt");
rename("AFCMod50pr.txt", "AFCMod50pr6.txt");
rename("AFCMod75pr.txt", "AFCMod75pr6.txt");
cout<<endl;

//Расчет для частоты 0,007
cout<<"Для частоты 0,007: "<<endl;
Everything(0.007);
rename("BezSh.txt", "BezSh7.txt");
rename("Med.txt", "Med7.txt");
rename("Mod25pr.txt", "Mod25pr7.txt");
rename("Mod50pr.txt", "Mod50pr7.txt");
rename("Mod75pr.txt", "Mod75pr7.txt");
rename("AFCBezSh.txt", "AFCBezSh7.txt");
rename("AFCMed.txt", "AFCMed7.txt");
rename("AFCMod25pr.txt", "AFCMod25pr7.txt");
```

```
rename("AFCMod50pr.txt", "AFCMod50pr7.txt");
rename("AFCMod75pr.txt", "AFCMod75pr7.txt");
cout<<endl;

//Расчет для частоты 0,008
cout<<"Для частоты 0,008: "<<endl;
Everything(0.008);
rename("BezSh.txt", "BezSh8.txt");
rename("Med.txt", "Med8.txt");
rename("Mod25pr.txt", "Mod25pr8.txt");
rename("Mod50pr.txt", "Mod50pr8.txt");
rename("Mod75pr.txt", "Mod75pr8.txt");
rename("AFCBezSh.txt", "AFCBezSh8.txt");
rename("AFCMed.txt", "AFCMed8.txt");
rename("AFCMod25pr.txt", "AFCMod25pr8.txt");
rename("AFCMod50pr.txt", "AFCMod50pr8.txt");
rename("AFCMod75pr.txt", "AFCMod75pr8.txt");
cout<<endl;

//Расчет для частоты 0,009
cout<<"Для частоты 0,009: "<<endl;
Everything(0.009);
rename("BezSh.txt", "BezSh9.txt");
rename("Med.txt", "Med9.txt");
rename("Mod25pr.txt", "Mod25pr9.txt");
rename("Mod50pr.txt", "Mod50pr9.txt");
rename("Mod75pr.txt", "Mod75pr9.txt");
rename("AFCBezSh.txt", "AFCBezSh9.txt");
rename("AFCMed.txt", "AFCMed9.txt");
rename("AFCMod25pr.txt", "AFCMod25pr9.txt");
rename("AFCMod50pr.txt", "AFCMod50pr9.txt");
rename("AFCMod75pr.txt", "AFCMod75pr9.txt");
cout<<endl;

//Расчет для частоты 0,01
cout<<"Для частоты 0,01: "<<endl;
Everything(0.01);
rename("BezSh.txt", "BezSh10.txt");
rename("Med.txt", "Med10.txt");
rename("Mod25pr.txt", "Mod25pr10.txt");
rename("Mod50pr.txt", "Mod50pr10.txt");
rename("Mod75pr.txt", "Mod75pr10.txt");
rename("AFCBezSh.txt", "AFCBezSh10.txt");
rename("AFCMed.txt", "AFCMed10.txt");
rename("AFCMod25pr.txt", "AFCMod25pr10.txt");
rename("AFCMod50pr.txt", "AFCMod50pr10.txt");
rename("AFCMod75pr.txt", "AFCMod75pr10.txt");
cout<<endl;
}

void Everything(double fs) //Функция для создания, фильтрации сигналов и
нахождения спектра
{
FILE *fp;   //указатель на файл
double *BezSh=new double[25000]; //задаем динамические массивы
double *Med=new double[25000];
double *Mod25pr=new double[25000];
double *Mod50pr=new double[25000];
double *Mod75pr=new double[25000];

for (int i=0; i<25000; i++)
      BezSh[i]=sin(2*3.141592*fs*i); //задаем синусоиду

//Обработка медианными фильтрами
```

```
double N[25099];
double P[100];
double q;

for (int i=0; i<25099; i++) //заполняем нулями для корректной работы фильтра
на краях
      N[i]=0;

for (int i=50; i<25050; i++)
      N[i]=BezSh[i-50]; //забиваем отсчеты сигнала, оставляя нули по краям

for (int i=0; i<25000; i++)
    {
       double sum=0;
       for (int j=0; j<100; j++)
         P[j]=N[i+j]; //задаем массив из 100 значений (окно фильтра)

       for (int k=0; k<99; k++)
             {
          for (int h=0; h<99-k; h++)
                 {
                  if (P[h]>P[h+1])
                     {
                  // Упорядочиваем отсчеты
                  q=P[h];
                          P[h]=P[h+1];
                  P[h+1]=q;
                       }
                 }
            }

      Med[i]=(P[49]+P[50])/2; //после обычного медианного фильтра

      for (int w=37; w<62; w++)
            sum+=P[w];
      Mod25pr[i]=sum/25; //после обр модиф мед фильтром с окном 25%
      sum=0;

      for (int w=25; w<75; w++)
            sum+=P[w];
      Mod50pr[i]=sum/50; //после обр модиф мед фильтром с окном 50%
      sum=0;

      for (int w=13; w<88; w++)
            sum+=P[w];
      Mod75pr[i]=sum/75; //после обр модиф мед фильтром с окном 75%
      }

//Применение взвешивающей функции
double WN=25000; //количество коэффициентов фильтра
double a=WN/2-1;

double *W=new double[25000]; //задаем динамический массив под взвешивающую
функцию
double *BezShVzv=new double[25000]; //динамические массивы под выходной
сигнал (умноженный на взвешенную функцию)
double *MedVzv=new double[25000];
double *Mod25prVzv=new double[25000];
double *Mod50prVzv=new double[25000];
double *Mod75prVzv=new double[25000];

for (int i=0; i<25000; i++)
      {
```

```
       W[i]=0.35875+0.48829*cos(2*3.141592/WN*(i-
a))+0.14128*cos(4*3.141592/WN*(i-a))+0.01168*cos(6*3.141592/WN*(i-a));
//Задаем ВФ Блекмана-Харриса
       BezShVzv[i]=BezSh[i]*W[i];//перемножаем отсчеты сигналов на отсчеты
взвешивающей функции
       MedVzv[i]=Med[i]*W[i];
       Mod25prVzv[i]=Mod25pr[i]*W[i];
       Mod50prVzv[i]=Mod50pr[i]*W[i];
       Mod75prVzv[i]=Mod75pr[i]*W[i];
      }
delete []W;


//Находим спектры
double *AFCBezSh=new double[12500]; //Динамические массивы под спектры
double *AFCMed=new double[12500];
double *AFCMod25pr=new double[12500];
double *AFCMod50pr=new double[12500];
double *AFCMod75pr=new double[12500];

for (int k=0; k<12500; k++)
      {
      double SumREBezSh=0; //Под значения вещественной
      double SumIMBezSh=0; //и мнимой составляющих спектра
      double SumREMed=0;
      double SumIMMed=0;
      double SumREMod25pr=0;
      double SumIMMod25pr=0;
      double SumREMod50pr=0;
      double SumIMMod50pr=0;
      double SumREMod75pr=0;
      double SumIMMod75pr=0;
      for (int i=0; i<24999; i++)
            {
            SumREBezSh=SumREBezSh+BezShVzv[i]*cos(2*3.1415*k*i/25000);
//находим суммы вещественной и мнимой составляющих из формулы ДПФ
            SumREMed=SumREMed+MedVzv[i]*cos(2*3.1415*k*i/25000);
            SumREMod25pr=SumREMod25pr+Mod25prVzv[i]*cos(2*3.1415*k*i/25000);
            SumREMod50pr=SumREMod50pr+Mod50prVzv[i]*cos(2*3.1415*k*i/25000);
            SumREMod75pr=SumREMod75pr+Mod75prVzv[i]*cos(2*3.1415*k*i/25000);

            SumIMBezSh=SumIMBezSh-BezShVzv[i]*sin(2*3.1415*k*i/25000);
            SumIMMed=SumIMMed-MedVzv[i]*sin(2*3.1415*k*i/25000);
            SumIMMod25pr=SumIMMod25pr-Mod25prVzv[i]*sin(2*3.1415*k*i/25000);
            SumIMMod50pr=SumIMMod50pr-Mod50prVzv[i]*sin(2*3.1415*k*i/25000);
            SumIMMod75pr=SumIMMod75pr-Mod75prVzv[i]*sin(2*3.1415*k*i/25000);
            }

      AFCBezSh[k]=sqrt(SumREBezSh*SumREBezSh+SumIMBezSh*SumIMBezSh); //находим
модуль
      AFCMed[k]=sqrt(SumREMed*SumREMed+SumIMMed*SumIMMed);
      AFCMod25pr[k]=sqrt(SumREMod25pr*SumREMod25pr+SumIMMod25pr*SumIMMod25pr);
      AFCMod50pr[k]=sqrt(SumREMod50pr*SumREMod50pr+SumIMMod50pr*SumIMMod50pr);
      AFCMod75pr[k]=sqrt(SumREMod75pr*SumREMod75pr+SumIMMod75pr*SumIMMod75pr);
      }

//Определяем коэффициент нелинейных искажений
Kn(AFCBezSh, fs);
Kn(AFCMed, fs);
Kn(AFCMod25pr, fs);
Kn(AFCMod50pr, fs);
Kn(AFCMod75pr, fs);

//Определяем коэффициент передачи
```

```
cout<<"Коэффициент передачи для медианного фильтра и модифицированных с
окнами 25%, 50% и 75% соответственно:"<<endl;
K(AFCMed, AFCBezSh, fs);
K(AFCMod25pr, AFCBezSh, fs);
K(AFCMod50pr, AFCBezSh, fs);
K(AFCMod75pr, AFCBezSh, fs);

//Переводим в дБ и нормируем max=0 дБ
double maxBezSh=0;
double maxMed=0;
double maxMod25pr=0;
double maxMod50pr=0;
double maxMod75pr=0;

for (int k=0; k<12500; k++)
      {
       AFCBezSh[k]=20*log10(AFCBezSh[k]);
       if (AFCBezSh[k]>maxBezSh)
             maxBezSh=AFCBezSh[k];
       AFCMed[k]=20*log10(AFCMed[k]);
       if (AFCMed[k]>maxMed)
             maxMed=AFCMed[k];
       AFCMod25pr[k]=20*log10(AFCMod25pr[k]);
       if (AFCMod25pr[k]>maxMod25pr)
             maxMod25pr=AFCMod25pr[k];
       AFCMod50pr[k]=20*log10(AFCMod50pr[k]);
       if (AFCMod50pr[k]>maxMod50pr)
             maxMod50pr=AFCMod50pr[k];
       AFCMod75pr[k]=20*log10(AFCMod75pr[k]);
       if (AFCMod75pr[k]>maxMod75pr)
             maxMod75pr=AFCMod75pr[k];
      }

for (int k=0; k<12500; k++)
      {
       AFCBezSh[k]=AFCBezSh[k]-maxBezSh;
       AFCMed[k]=AFCMed[k]-maxMed;
       AFCMod25pr[k]=AFCMod25pr[k]-maxMod25pr;
       AFCMod50pr[k]=AFCMod50pr[k]-maxMod50pr;
       AFCMod75pr[k]=AFCMod75pr[k]-maxMod75pr;
      }

//Заносим результаты в файлы
fp=fopen("BezSh.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",BezSh[i]);
       }
fclose(fp);

fp=fopen("Med.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Med[i]);
       }
fclose(fp);

fp=fopen("Mod25pr.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Mod25pr[i]);
```

```
       }
fclose(fp);

fp=fopen("Mod50pr.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Mod50pr[i]);
       }
fclose(fp);

fp=fopen("Mod75pr.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Mod75pr[i]);
       }
fclose(fp);

fp=fopen("AFCBezSh.txt","w");
for (int i=0; i<12500; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",AFCBezSh[i]);
       }
fclose(fp);

fp=fopen("AFCMed.txt","w");
for (int i=0; i<12500; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",AFCMed[i]);
       }
fclose(fp);

fp=fopen("AFCMod25pr.txt","w");
for (int i=0; i<12500; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",AFCMod25pr[i]);
       }
fclose(fp);

fp=fopen("AFCMod50pr.txt","w");
for (int i=0; i<12500; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",AFCMod50pr[i]);
       }
fclose(fp);

fp=fopen("AFCMod75pr.txt","w");
for (int i=0; i<12500; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",AFCMod75pr[i]);
       }
fclose(fp);

//Очищаем память
delete []AFCBezSh;
delete []AFCMed;
delete []AFCMod25pr;
delete []AFCMod50pr;
```

```
delete []AFCMod75pr;

delete []BezSh;
delete []Med;
delete []Mod25pr;
delete []Mod50pr;
delete []Mod75pr;

delete []BezShVzv;
delete []MedVzv;
delete []Mod25prVzv;
delete []Mod50prVzv;
delete []Mod75prVzv;
}

void Kn (double *AFC, double fs) //Функция нахождения коэффициента нелинейных
искажений
{
double Usign=0;
double Ushum=0;

for (int i=0; i<12500; i++)
      {
       if (i>=25000*fs-4 && i<=25000*fs+4)
             Usign+=AFC[i]*AFC[i];
       else Ushum+=AFC[i]*AFC[i];
      }
double K=sqrt(Ushum/Usign);
cout<<K<<endl;
}

void K (double *AFC, double *AFCOriginal, double fs) //Функция нахождения
коэффициента передачи
{
double A=0;
double B=0;
int k=25000*fs;

for (int i=25000*fs-4; i<25000*fs+5; i++)
      {
       A+=AFC[i]*AFC[i];
       B+=AFCOriginal[i]*AFCOriginal[i];
      }
A=sqrt(A/B);
cout<<A<<endl;
     }
```

Код программы определения ОСШ для случая гауссовского шума

```
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <fstream>
using namespace std;

double NormalShum(); //Прототип функции задания нормального распределенния
double SNRSh(double *Signal); //Прототип функции нахождения ОСШ входного
сигнала
double SNR(double *Signal); //Прототип функции нахождения ОСШ нелинейных
фильтров
double SNRSogl(double *Signal); //Прототип функции нахождения ОСШ
согласованного фильтра

void main()
{
setlocale(LC_ALL,"rus_rus.1251");
srand(time(NULL));

FILE *fp;   //Указатель на файл
double *BezSh=new double[25000]; //Задаем динамические массивы
double *SSh=new double[25000];
double *Med=new double[25000];
double *Mod25pr=new double[25000];
double *Mod50pr=new double[25000];
double *Mod75pr=new double[25000];
double *Soglas=new double[25000];
double *Shum=new double[25000];

for (int i=0; i<25000; i++) //Задаем нормально распределенный шум
      Shum[i]=NormalShum()/3.17; //ОСШ на входе регулируется знаменателем: при
1,415 ОСШ=2; при 3,17 ОСШ=10

for (int i=0; i<25000; i++)
      {
      if (i>=5000&&i<5100 || i>=10000&&i<10100 || i>=15000&&i<15100 ||
i>=20000&&i<20100)
            BezSh[i]=1;
      else BezSh[i]=0; //Задаем прямоугольные импульсы
      SSh[i]=BezSh[i]+Shum[i]; //Добавляем шум
      }
delete []Shum; //Освобождаем память

//Обработка согласованным фильтром
for (int i=0; i<25000; i++)
      {
       double sum=0;
       for (int j=0; j<100; j++)
             if (i-j>=0)
                  sum+=SSh[i-j]; //при i-j<0 нет сигнала, поэтому принимаем
его значение равным 0, следовательно, сумма не изменится при его сложении
       Soglas[i]=sum/100;
      }

//Обработка медианными фильтрами
double N[25099];
double P[100];
```

```
double q;

for (int i=0; i<25099; i++) //Заполняем нулями для корректной работы фильтра
на краях
      N[i]=0;

for (int i=50; i<25050; i++)
      N[i]=SSh[i-50]; //Забиваем отсчеты сигнала, оставляя нули по краям

for (int i=0; i<25000; i++)
    {
       double sum=0;
       for (int j=0; j<100; j++)
         P[j]=N[i+j]; //Задаем массив из 100 значений (окно фильтра)

       for (int k=0; k<99; k++)
             {
          for (int h=0; h<99-k; h++)
                 {
                  if (P[h]>P[h+1])
                     {
                  // Упорядочиваем отсчеты
                  q=P[h];
                          P[h]=P[h+1];
                  P[h+1]=q;
                       }
                 }
            }

      Med[i]=(P[49]+P[50])/2; //После обычного медианного фильтра

      for (int w=37; w<62; w++)
            sum+=P[w];
      Mod25pr[i]=sum/25; //После обр модиф мед фильтром с окном 25%
      sum=0;

      for (int w=25; w<75; w++)
            sum+=P[w];
      Mod50pr[i]=sum/50; //после обр модиф мед фильтром с окном 50%
      sum=0;

      for (int w=13; w<88; w++)
            sum+=P[w];
      Mod75pr[i]=sum/75; //после обр модиф мед фильтром с окном 75%
      }

//Расчет ОСШ
cout<<"Сигнал с шумом:"<<endl;
double SNRSSh=SNRSh(SSh);
cout<<"После обработки медианным фильтром:"<<endl;
double SNRMed=SNR(Med);
cout<<"После обработки модифицированным медианным фильтром с окном
25%:"<<endl;
double SNRMod25pr=SNR(Mod25pr);
cout<<"После обработки модифицированным медианным фильтром с окном
50%:"<<endl;
double SNRMod50pr=SNR(Mod50pr);
cout<<"После обработки модифицированным медианным фильтром с окном
75%:"<<endl;
double SNRMod75pr=SNR(Mod75pr);
cout<<"После обработки согласованным фильтром:"<<endl;
double SNRSoglas=SNRSogl(Soglas);

//Заносим результаты в файлы
```

```
fp=fopen("BezSh.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",BezSh[i]);
       }
fclose(fp);

fp=fopen("SSh.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",SSh[i]);
       }
fclose(fp);

fp=fopen("Med.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Med[i]);
       }
fclose(fp);

fp=fopen("Mod25pr.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Mod25pr[i]);
       }
fclose(fp);

fp=fopen("Mod50pr.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Mod50pr[i]);
       }
fclose(fp);

fp=fopen("Mod75pr.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Mod75pr[i]);
       }
fclose(fp);

fp=fopen("Soglas.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Soglas[i]);
       }
fclose(fp);

//Очищаем память
delete []BezSh;
delete []SSh;
delete []Med;
delete []Mod25pr;
delete []Mod50pr;
delete []Mod75pr;
delete []Soglas;
```

```
}

double NormalShum() //Функция нормального распределенния согласно
преобразованию Бокса-Мюллера
{
    double u=((double) rand()/(RAND_MAX))*2-1;
    double v=((double) rand()/(RAND_MAX))*2-1;
    double r=u*u+v*v;
    if (r==0 || r>1)
            return NormalShum();
    double c=sqrt(-2*log(r)/r);
    return u*c;
}

double SNRSh(double *Signal) //Функция нахождения ОСШ входного сигнала
{
double Psign=1;
double Pshum=0;

for (int i=0; i<25000; i++)
      if (i<=5000 || i>5100&&i<=10000 || i>10100&&i<=15000 ||
i>15100&&i<=20000 || i>20100)
            Pshum+=Signal[i]*Signal[i]/24600;
double S=Psign/Pshum;
cout<<"Мощность сигнала = ";
cout<<Psign;
cout<<' ';
cout<<"Мощность шума = ";
cout<<Pshum;
cout<<' ';
cout<<"ОСШ = ";
cout<<S;
cout<<' ';
cout<<endl;
return S;
}

double SNR(double *Signal) //Функция нахождения ОСШ нелинейных фильтров
{
double Psign=0;
double Pshum=0;

for (int i=0; i<25000; i++)
      if (i>=4950&&i<5150 || i>=9950&&i<10150 || i>=14950&&i<15150 ||
i>=19950&&i<20150)
            Psign+=Signal[i]*Signal[i]/800;
      else Pshum+=Signal[i]*Signal[i]/24200;
double S=Psign/Pshum;
cout<<"Мощность сигнала = ";
cout<<Psign;
cout<<' ';
cout<<"Мощность шума = ";
cout<<Pshum;
cout<<' ';
cout<<"ОСШ = ";
cout<<S;
cout<<' ';
cout<<endl;
return S;
}

double SNRSogl(double *Signal) //Функция нахождения ОСШ согласованного
фильтра
{
```

```
double Psign=0;
double Pshum=0;

for (int i=0; i<25000; i++)
      if (i>=5000&&i<5200 || i>=10000&&i<10200 || i>=15000&&i<15200 ||
i>=20000&&i<20200)
            Psign+=Signal[i]*Signal[i]/800;
      else Pshum+=Signal[i]*Signal[i]/24200;
double S=Psign/Pshum;
cout<<"Мощность сигнала = ";
cout<<Psign;
cout<<' ';
cout<<"Мощность шума = ";
cout<<Pshum;
cout<<' ';
cout<<"ОСШ = ";
cout<<S;
cout<<' ';
cout<<endl;
return S;
     }
```

Код программы определения ОСШ для случая импульсного шума

```
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <fstream>
using namespace std;

double ImpShum(); //Прототип функции ХИП
double SNRSh(double *Signal); //Прототип функции нахождения ОСШ входного
сигнала
double SNR(double *Signal); //Прототип функции нахождения ОСШ нелинейных
фильтров
double SNRSogl(double *Signal); //Прототип функции нахождения ОСШ
согласованного фильтра

void main()
{
setlocale(LC_ALL,"rus_rus.1251");
srand(time(NULL));

FILE *fp;   //Указатель на файл
double *BezSh=new double[25000]; //Задаем динамические массивы
double *SSh=new double[25000];
double *Med=new double[25000];
double *Mod25pr=new double[25000];
double *Mod50pr=new double[25000];
double *Mod75pr=new double[25000];
double *Soglas=new double[25000];
double *Shum=new double[25000];

for (int i=0; i<25000; i++)
      {
      if (i>=5000&&i<5100 || i>=10000&&i<10100 || i>=15000&&i<15100 ||
i>=20000&&i<20100)
            BezSh[i]=1;
      else BezSh[i]=0; //Задаем прямоугольные импульсы
      SSh[i]=BezSh[i]+ImpShum()+ImpShum()+ImpShum()+ImpShum()+ImpShum(); //Для
получения ОСШ на входе = 2 требуется 5 раз использовать функцию генерирования
импульсного шума. Для ОСШ = 10 - только 1 раз
      }

//Обработка согласованным фильтром
for (int i=0; i<25000; i++)
      {
       double sum=0;
       for (int j=0; j<100; j++)
             if (i-j>=0)
                  sum+=SSh[i-j]; //При i-j<0 нет сигнала, поэтому принимаем
его значение равным 0, следовательно, сумма не изменится при его сложении
       Soglas[i]=sum/100;
      }

//Обработка медианными фильтрами
double N[25099];
double P[100];
double q;
```

```
for (int i=0; i<25099; i++) //Заполняем нулями для корректной работы фильтра
на краях
      N[i]=0;

for (int i=50; i<25050; i++)
      N[i]=SSh[i-50]; //Забиваем отсчеты сигнала, оставляя нули по краям

for (int i=0; i<25000; i++)
    {
       double sum=0;
       for (int j=0; j<100; j++)
         P[j]=N[i+j]; //Задаем массив из 100 значений (окно фильтра)

       for (int k=0; k<99; k++)
             {
          for (int h=0; h<99-k; h++)
                 {
                  if (P[h]>P[h+1])
                     {
                  //Упорядочиваем отсчеты
                  q=P[h];
                          P[h]=P[h+1];
                  P[h+1]=q;
                       }
                 }
             }
      Med[i]=(P[49]+P[50])/2; //После обычного медианного фильтра

      for (int w=37; w<62; w++)
            sum+=P[w];
      Mod25pr[i]=sum/25; //После обр модиф мед фильтром с окном 25%
      sum=0;

      for (int w=25; w<75; w++)
            sum+=P[w];
      Mod50pr[i]=sum/50; //После обр модиф мед фильтром с окном 50%
      sum=0;

      for (int w=13; w<88; w++)
            sum+=P[w];
      Mod75pr[i]=sum/75; //После обр модиф мед фильтром с окном 75%
      }

//Расчет ОСШ
cout<<"Сигнал с шумом:"<<endl;
double SNRSSh=SNRSh(SSh);
cout<<"После обработки медианным фильтром:"<<endl;
double SNRMed=SNR(Med);
cout<<"После обработки модифицированным медианным фильтром с окном
25%:"<<endl;
double SNRMod25pr=SNR(Mod25pr);
cout<<"После обработки модифицированным медианным фильтром с окном
50%:"<<endl;
double SNRMod50pr=SNR(Mod50pr);
cout<<"После обработки модифицированным медианным фильтром с окном
75%:"<<endl;
double SNRMod75pr=SNR(Mod75pr);
cout<<"После обработки согласованным фильтром:"<<endl;
double SNRSoglas=SNRSogl(Soglas);

//Заносим результаты в файлы
fp=fopen("BezSh.txt","w");
for (int i=0; i<25000; i++)
     {
```

```
        fputs(" ",fp);
      fprintf(fp,"%lf",BezSh[i]);
       }
fclose(fp);

fp=fopen("SSh.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",SSh[i]);
       }
fclose(fp);

fp=fopen("Med.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Med[i]);
       }
fclose(fp);

fp=fopen("Mod25pr.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Mod25pr[i]);
       }
fclose(fp);

fp=fopen("Mod50pr.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Mod50pr[i]);
       }
fclose(fp);

fp=fopen("Mod75pr.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Mod75pr[i]);
       }
fclose(fp);

fp=fopen("Soglas.txt","w");
for (int i=0; i<25000; i++)
     {
        fputs(" ",fp);
      fprintf(fp,"%lf",Soglas[i]);
       }
fclose(fp);

delete []BezSh;
delete []SSh;
delete []Med;
delete []Mod25pr;
delete []Mod50pr;
delete []Mod75pr;
delete []Soglas;
}

double ImpShum() //Получение ХИП
{
```

```
    double u=((double) rand()/(RAND_MAX))*2-1;
    double v=((double) rand()/(RAND_MAX))*2-1;
    double r=u*u+v*v;
    if (r==0 || r>1)
            return ImpShum();
    double c=sqrt(-2*log(r)/r);
    if (u*c>2.5 || u*c<-2.5)
            return u*c;
      else return 0;
}

double SNRSh(double *Signal) //Функция нахождения ОСШ входного сигнала
{
double Psign=1;
double Pshum=0;

for (int i=0; i<25000; i++)
      if (i<=5000 || i>5100&&i<=10000 || i>10100&&i<=15000 ||
i>15100&&i<=20000 || i>20100)
            Pshum+=Signal[i]*Signal[i]/24600;
double S=Psign/Pshum;
cout<<"ОСШ = ";
cout<<S;
cout<<' ';
cout<<endl;
return S;
}

double SNR(double *Signal) //Функция нахождения ОСШ нелинейных фильтров
{
double Psign=0;
double Pshum=0;

for (int i=0; i<25000; i++)
      if (i>=4950&&i<=5150 || i>=9950&&i<=10150 || i>=14950&&i<=15150 ||
i>=19950&&i<=20150)
            Psign+=Signal[i]*Signal[i]/800;
      else Pshum+=Signal[i]*Signal[i]/24200;
double S=Psign/Pshum;
cout<<"ОСШ = ";
cout<<S;
cout<<' ';
cout<<endl;
return S;
}

double SNRSogl(double *Signal) //Функция нахождения ОСШ согласованного
фильтра
{
double Psign=0;
double Pshum=0;

for (int i=0; i<25000; i++)
      if (i>=5000&&i<5200 || i>=10000&&i<10200 || i>=15000&&i<15200 ||
i>=20000&&i<20200)
            Psign+=Signal[i]*Signal[i]/800;
      else Pshum+=Signal[i]*Signal[i]/24200;
double S=Psign/Pshum;
cout<<"ОСШ = ";
cout<<S;
cout<<' ';
cout<<endl;
return S;
     }
```

Приложение 4

Код программы определения ОСШ для случая смеси гауссовского и импульсного шума

```
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <fstream>
using namespace std;

void Everything(double *BezSh, double *SSh, double *Med, double *Mod25pr,
double *Mod50pr, double *Mod75pr, double *Soglas); //Прототип функции задания
сигнала, шума и фильтрации
double NormalShum(); //Прототип функции нормального распределения
double ImpShum(); //Прототип функции ХИП
double SNRSh(double *Signal); //Прототип функции нахождения ОСШ входного
сигнала
double SNR(double *Signal); //Прототип функции нахождения ОСШ нелинейных
фильтров
double SNRSogl(double *Signal); //Протип функции нахождения ОСШ согласованног
фильтра

void main()
{
setlocale(LC_ALL,"rus_rus.1251");
srand(time(NULL));

FILE *fp;   //указатель на файл
double *BezSh=new double[25000]; //задаем динамические массивы
double *SSh=new double[25000];
double *Med=new double[25000];
double *Mod25pr=new double[25000];
double *Mod50pr=new double[25000];
double *Mod75pr=new double[25000];
double *Soglas=new double[25000];

double ResSh=0;
double ResMed=0;
double ResMod25=0;
double ResMod50=0;
double ResMod75=0;
double ResSogl=0;


for (int i=0; i<100; i++) //Будем усреднять результаты по 100 выборкам
      {
      Everything(BezSh, SSh, Med, Mod25pr, Mod50pr, Mod75pr, Soglas);
      ResSh+=SNRSh(SSh)/100;
      ResMed+=SNR(Med)/100;
      ResMod25+=SNR(Mod25pr)/100;
      ResMod50+=SNR(Mod50pr)/100;
      ResMod75+=SNR(Mod75pr)/100;
      ResSogl+=SNRSogl(Soglas)/100;
      }

cout<<"Сигнал с шумом: "<<ResSh;
cout<<endl;
cout<<"После обработки медианным фильтром: "<<ResMed;
cout<<endl;
```

```
cout<<"После обработки модифицированным медианным фильтром с окном 25%
:"<<ResMod25;
cout<<endl;
cout<<"После обработки модифицированным медианным фильтром с окном 50%
:"<<ResMod50;
cout<<endl;
cout<<"После обработки модифицированным медианным фильтром с окном 75%:
"<<ResMod75;
cout<<endl;
cout<<"После обработки согласованным фильтром: "<<ResSogl;
cout<<endl;

delete []BezSh;
delete []SSh;
delete []Med;
delete []Mod25pr;
delete []Mod50pr;
delete []Mod75pr;
delete []Soglas;
}


void Everything(double *BezSh, double *SSh, double *Med, double *Mod25pr,
double *Mod50pr, double *Mod75pr, double *Soglas) //Функция задания сигнала,
шума и фильтрации
{
double *ShumI=new double[25000];
double *ShumG=new double[25000];
double Impuls=0;
double Gaus=0;

for (int i=0; i<25000; i++) //Задаем нормально распределенный шум
      {
       ShumG[i]=NormalShum()/3.15; //Изменяя знаменатель, регулируем мощность
нормального шума
       ShumI[i]=ImpShum();
       Impuls+=ShumI[i]*ShumI[i];
       Gaus+=ShumG[i]*ShumG[i];
      }

for (int i=0; i<25000; i++)
      {
      if (i>=5000&&i<5100 || i>=10000&&i<10100 || i>=15000&&i<15100 ||
i>=20000&&i<20100)
            BezSh[i]=1;
      else BezSh[i]=0; //Задаем прямоугольные импульсы
      SSh[i]=BezSh[i]+ShumG[i]+ShumI[i];
      }
delete []ShumI; //Освобождаем память
delete []ShumG;

//Обработка согласованным фильтром
for (int i=0; i<25000; i++)
      {
       double sum=0;
       for (int j=0; j<100; j++)
             if (i-j>=0)
                  sum+=SSh[i-j]; //При i-j<0 нет сигнала, поэтому принимаем
его значение равным 0, следовательно, сумма не изменится при его сложении
       Soglas[i]=sum/100;
      }

//Обработка медианными фильтрами
double N[25099];
```

```
double P[100];
double q;

for (int i=0; i<25099; i++) //заполняем нулями для корректной работы фильтра
на краях
      N[i]=0;

for (int i=50; i<25050; i++)
      N[i]=SSh[i-50]; //Забиваем отсчеты сигнала, оставляя нули по краям

for (int i=0; i<25000; i++)
    {
       double sum=0;
       for (int j=0; j<100; j++)
         P[j]=N[i+j]; //Задаем массив из 100 значений (окно фильтра)

       for (int k=0; k<99; k++)
             {
          for (int h=0; h<99-k; h++)
                 {
                  if (P[h]>P[h+1])
                     {
                  //Упорядочиваем отсчеты
                  q=P[h];
                          P[h]=P[h+1];
                  P[h+1]=q;
                       }
                 }
           }

      Med[i]=(P[49]+P[50])/2; //После обычного медианного фильтра

      for (int w=37; w<62; w++)
            sum+=P[w];
      Mod25pr[i]=sum/25; //После модиф мед фильтром с окном 25%
      sum=0;

      for (int w=25; w<75; w++)
            sum+=P[w];
      Mod50pr[i]=sum/50; //После модиф мед фильтром с окном 50%
      sum=0;

      for (int w=13; w<88; w++)
            sum+=P[w];
      Mod75pr[i]=sum/75; //После модиф мед фильтром с окном 75%
      }
}

double NormalShum() //Получение нормального распределенния согласно
преобразованию Бокса-Мюллера
{
    double u=((double) rand()/(RAND_MAX))*2-1;
    double v=((double) rand()/(RAND_MAX))*2-1;
    double r=u*u+v*v;
    if (r==0 || r>1)
           return NormalShum();
    double c=sqrt(-2*log(r)/r);
    return u*c;
}

double ImpShum() //Получение ХИП
{
    double u=((double) rand()/(RAND_MAX))*2-1;
    double v=((double) rand()/(RAND_MAX))*2-1;
```

```
    double r=u*u+v*v;
    if (r==0 || r>1)
            return ImpShum();
    double c=sqrt(-2*log(r)/r);
    if (u*c>3.7 || u*c<-3.7) //Регулируя эти значения, изменяем мощность
импульсного шума
            return u*c;
      else return 0;
}

double SNRSh(double *Signal) //Функция нахождения ОСШ входного сигнала
{
double Psign=1;
double Pshum=0;

for (int i=0; i<25000; i++)
      if (i<=5000 || i>5100&&i<=10000 || i>10100&&i<=15000 ||
i>15100&&i<=20000 || i>20100)
            Pshum+=Signal[i]*Signal[i]/24600;
double S=Psign/Pshum;
return S;
}

double SNR(double *Signal) //Функция нахождения ОСШ нелинейного фильтра
{
double Psign=0;
double Pshum=0;

for (int i=0; i<25000; i++)
      if (i>=4950&&i<5150 || i>=9950&&i<10150 || i>=14950&&i<15150 ||
i>=19950&&i<20150)
            Psign+=Signal[i]*Signal[i]/800;
      else Pshum+=Signal[i]*Signal[i]/24200;
double S=Psign/Pshum;
return S;
}

double SNRSogl(double *Signal) //Функция нахождения ОСШ согласованного
фильтра
{
double Psign=0;
double Pshum=0;

for (int i=0; i<25000; i++)
      if (i>=5000&&i<5200 || i>=10000&&i<10200 || i>=15000&&i<15200 ||
i>=20000&&i<20200)
            Psign+=Signal[i]*Signal[i]/800;
      else Pshum+=Signal[i]*Signal[i]/24200;
double S=Psign/Pshum;
return S;
     }
```