Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра электронной техники и технологии

ЦИФРОВАЯ ОБРАБОТКА БИОМЕДИЦИНСКИХ СИГНАЛОВ И ИЗОБРАЖЕНИЙ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальности 1-39 02 03 «Медицинская электроника»

Минск БГУИР 2015

УДК 004.421:616-073
ББК 32.973.26-018.2+53.4
Ц75

А в т о р ы:
М. В. Давыдов, В. М. Бондарик, Н. С. Давыдова, А. С. Терех

Р е ц е н з е н т ы:
кафедра конструирования и производства приборов Белорусского национального технического университета (протокол №7 от 20.01.2014);

декан педиатрического факультета учреждения образования «Белорусский государственный медицинский университет», кандидат медицинских наук, доцент В. А. Кепеть

Цифровая обработка биомедицинских сигналов и изображений. Ц75 Лабораторный практикум : учеб.-метод. пособие / М. В. Давыдов [и др.]. – Минск : БГУИР, 2015. – 75 с. : ил.
ISBN 978-985-543-088-0.

Рассматриваются вопросы программирования в системе MATLAB, синтеза биологических сигналов с различной степенью зашумленности, разрабатывается программа свертки и изучается влияние различных импульсных характеристик на свойства данного сигнала. Выполняется разработка программы спектрального преобразования Фурье и исследование частотных свойств биологических сигналов. Изучаются возможности построения изображения биологических объектов, функции работы с биомедицинскими изображениями, полученными в результате МРТ.

Предназначено для закрепления и углубления теоретических знаний, приобретения практических навыков работы в области цифровой обработки биомедицинских сигналов и изображений.

УДК 004.421:616-073
ББК 32.973.26-018.2+53.4

ISBN 978-985-543-088-0

СОДЕРЖАНИЕ

Лабораторная работа №1

Исследование свойств свертки цифровых сигналов

1.1 Цель работы

Изучение основ работы среды *MATLAB,* приобретение навыков генерирования сигналов с заданными свойствами.

1.2 Задачи

1 Разработать функцию построения сигнала электрокардиограммы (ЭКГ) с указанными в задании параметрами.

2 Разработать функцию построения сигнала «белого» шума с указанными в задании параметрами.

3 Разработать функцию построения сигнала гармонического шума (сетевой наводки) с указанными в задании параметрами.

4 Разработать функцию построения зашумленного сигнала ЭКГ.

5 Разработать функцию свертки цифрового сигнала с заданной импульсной характеристикой.

1.3 Общая теория. Программирование в системе *MATLAB*

Программами в системе *MATLAB* являются *m*-файлы текстового формата, содержащие запись программ в виде программных кодов. Язык программирования системы *MATLAB* имеет следующие средства:

– данные различного типа;
– константы и переменные;
– операторы, включая операторы математических выражений;
– встроенные команды и функции;
– функции пользователя;
– управляющие структуры;
– системные операторы и функции;
– средства расширения языка.

1.3.1 Структура *m*-файла-функции

M-файл-функция является типичным объектом языка программирования системы *MATLAB* [1–3]. Одновременно он является полноценным модулем с точки зрения структурного программирования, поскольку содержит входные и выходные параметры и использует аппарат локальных переменных. Структура такого модуля с одним выходным параметром выглядит следующим образом:

function var=f_name(Список_параметров)

%Основной комментарий

%Дополнительный комментарий

Тело файла с любыми выражениями

var=выражение

M-файл-функция имеет следующие свойства:

– начинается с объявления *function*, после которого указывается имя переменной v*ar* – выходного параметра, имя самой функции и список ее входных параметров;

– функция возвращает свое значение и может использоваться в виде name (Список_параметров) в математических выражениях;

– все переменные, имеющиеся в теле файла-функции, являются локальными, т. е. действуют только в пределах тела функции;

– файл-функция является самостоятельным программным модулем, который общается с другими модулями через свои входные и выходные параметры;

– файл-функция служит средством расширения системы *MATLAB*;

– при своем обнаружении файл-функция компилируется и затем исполняется, а созданные машинные коды хранятся в рабочей области системы *MATLAB*.

Последняя конструкция *var*=выражение вводится, если требуется, чтобы функция возвращала результат вычислений.

Приведенная форма файла-функции характерна для функции с одним выходным параметром. Если выходных параметров больше, то они указываются в квадратных скобках после слова function. При этом структура модуля имеет следующий вид:

function [*var*1,*var2*...]=*f_name*(Список_параметров)

%Основной комментарий

%Дополнительный комментарий

Тело файла с любыми выражениями

*var*1=выражение

*var*2=выражение

Такая функция обладает свойствами процедуры. Ее нельзя слепо использовать непосредственно в математических выражениях, поскольку она возвращает не единственный результат, а множество результатов по числу выходных параметров. Если функция используется как имеющая единственный выходной параметр, но имеет ряд выходных параметров, то для возврата значения будет использоваться первый из них. Это зачастую ведет к ошибкам в математических вычислениях. Поэтому, как отмечалось ранее, данная функция используется как отдельный элемент программ вида

[*var*1,*var*2,...]=*f_name*(Список_параметров)

После его применения переменные выхода *var*1, *var*2 и т. д. становятся определенными и их можно использовать в последующих математических выражениях и иных сегментах программы.

1.3.2 Комментарии

Команда *help name*, где *name* – имя *m*-файла, обеспечивает чтение первой строки с текстовым комментарием и тех строк с комментариями, которые следуют непосредственно за первой строкой. Комментарий, расположенный за пределами этой области, не выводится. Это позволяет создавать невыводимый программный комментарий, например *Z*=*X*+*Y*, где массив *Z* является суммой массивов *X* и *Y*.

Пустая строка прерывает вывод комментария при исполнении команды *help name*. Команда *type name* выводит текст программы со всеми комментариями, в том числе и следующими после пустых строк.

1.3.3 Особенности выполнения *m*-файлов-функций

M-файлы-функции могут использоваться как в командном режиме, так и вызываться из других *m*-файлов. При этом необходимо указывать все входные и выходные параметры. Исключением является случай, когда выходной параметр единственный – в этом варианте функция возвращает единственный результат и может использоваться в математических выражениях.

1.3.4 Управляющие структуры

Помимо программ с линейной структурой, инструкции которых исполняются строго по порядку, существует множество программ, структура которых нелинейна. При этом ветви программ могут выполняться в зависимости от определенных условий, иногда с конечным числом повторений – циклов, иногда в виде циклов, завершаемых при выполнении заданного условия. Практически любая серьезная программа имеет нелинейную структуру. Для создания таких программ необходимы специальные управляющие структуры. Они имеются в любом языке программирования и, в частности, в *MATLAB*.

Циклы типа for...end

Циклы типа *for...end* обычно используются для организации вычислений с заданным числом повторяющихся циклов. Конструкция такого цикла имеет следующий вид:

for var=Выражение.
Инструкция.
...
Инструкция
end

Выражение чаще всего записывается в виде *s:d:e*, где *s* – начальное значение переменной цикла *var*; *d* – приращение этой переменной; *e* – конечное значение управляющей переменной, при достижении которого цикл завершается. Возможна и запись в виде *s:e* (в этом случае *d* = l). Список выполняемых в цикле инструкций завершается оператором *end*.

Пример 1 – Вычисление квадрата нечетных чисел от 1 до 99

```
% определение массива для хранения результата
Rez = [];
% задаем цикл для переменной i от 1 до 100 через 2
for i = 1:2:99
    % в каждой итерации цикла добавляем результат в массив
    Rez = [Rez  i^2];
end
```

Пример 2 – Оформление цикла в виде отдельного *m*-файла

```
function Out = OddSquare (Beg_Num, End_Num)
% определение массива для хранения результата
Rez = [];
% задаем цикл для i от 1 до 100 через 2
for i = Beg_Num:2:End_Num
    %добавляем текущий результат в массив
    Rez = [Rez  i^2];
end
% присваиваем выходной переменной значение
Out = Rez;
```

Для запуска данного примера необходимо в рабочем окне *MATLAB* выполнить следующие действия:

1 Вызов окна редактора *m*-файлов путем нажатия кнопки *New M-File* (Создать *m*-файл).

2 Ввод строки приведенного кода.

3 Ключевое слово *function* объявляет новую функцию, имя которой *OddSquare*, а ее параметры – *Beg_Num*, *End_Num* – начало и конец диапазона, в котором вычисляются квадраты нечетных чисел. Символ *Out* – выходная переменная функции.

4 Сохранение функции пользователя на диске. Для этого достаточно щелкнуть мышью по кнопке *Save* (Сохранить). **Название *m*-файла должно совпадать с названием функции** (в данном случае *OddSquare.m*).

5 Закрытие окна редактора *m*-файлов.

6 В командном окне *MATLAB* набрать >> *Arr* = *OddSquare* (3, 111).

При этом будет сформирован массив *Arr*, состоящий из 55 элементов, которые являются квадратами нечетных чисел в диапазоне от 3 до 111.

1.3.5 Визуализация вычислений

Основными функциями двумерной графики являются:
plot(*x*, *y*)
plot(*x*, *y*, *s*)
Здесь:
– *x* – аргумент функции, задаваемой в виде вектора;
– *y* – функция, представленная в аналитическом виде или в виде вектора или матрицы;
– *s* – вектор стилей графика; константа, определяющая цвет линий графика, тип точек и тип линий (таблица 1.1).

Таблица 1.1 – Стили графиков

Цвет линии		**Тип точки**		**Тип линии**	
Y	Желтый	.	Точка	-	Сплошная
M	Фиолетовый	O	Окружность	:	Двойной пунктир
C	Голубой	×	Крест	-.	Штрихпунктир
R	Красный	+	Плюс	--	Штриховая
G	Зеленый	*	Восьмиконечная снежинка		
B	Синий	S	Квадрат		
W	Белый	D	Ромб		
K	Черный	V, ^, <, >	Треугольник вверх, вниз, влево, вправо		
		P	Пятиконечная звезда		
		H	Шестиконечная звезда		

Пример 3 – Разработанная функция с добавлением вывода полученных значений

```
function Out = OddSquare (Beg_Num, End_Num)
% определение массива для хранения результата
Rez = [];
Num = [];
% задаем цикл для переменной i от 1 до 100 через 2
for i = Beg_Num:2:End_Num
    Rez = [Rez  i^2];           % добавляем результат в массив
    Num = [Num  i];
end
plot(Num, Rez,['R','*','-.']); % графич. построение результата
Out = Rez;                     % присваиваем выходное значение
```

1.3.6 Реализация алгоритма свертки

Просуммируем знания о преобразовании входного сигнала системы в выходной. Во-первых, входной сигнал может быть разложен на набор входных импульсов, каждый из которых может рассматриваться как промасштабированная и сдвинутая дельта-функция. Во-вторых, выход от каждого импульса есть промасштабированный и сдвинутый импульсный отклик. В-третьих, целиком выходной сигнал может быть найден суммированием этих промасштабированных и сдвинутых импульсных откликов. Другими словами, если задан импульсный отклик системы, то можно вычислить выходной сигнал для любого входного сигнала (рисунок 1.1) [4–8].

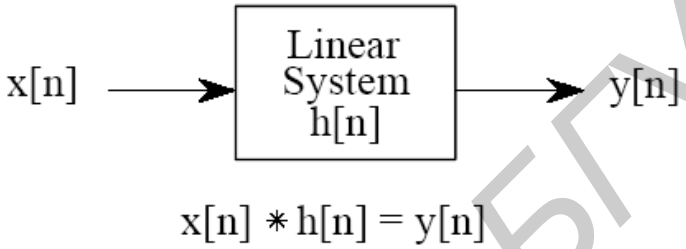


Рисунок 1.1 – Выходной сигнал линейной системы равен свертке входного сигнала с импульсным откликом системы

Импульсный отклик имеет различные названия в некоторых публикациях. Если рассматриваемая система есть фильтр, то импульсный отклик называется ядро фильтра, свертка ядра или просто ядро. При обработке изображений импульсный отклик называется функцией расширения точки. Хотя эти термины используются при немного различных подходах, все они означают то же самое – сигнал, вырабатываемый системой, когда на ее вход поступает дельта-функция.

Свертка есть формальная математическая операция, аналогичная умножению, сложению и интегрированию. Операция сложение берет два числа и производит третье число, свертка аналогично берет два сигнала и производит третий сигнал. Свертка используется во многих областях математики, таких, как теория вероятности и статистика. Свертка обозначается звездочкой, когда записывается в виде формулы (см. рисунок 1.1).

В линейных системах свертка используется для описания соотношения между тремя важными сигналами: входным сигналом, импульсным откликом и выходным сигналом.

Математическое описание механизма свертки. Если $x[n]$ есть N-точечный сигнал с номерами от 0 до $N - 1$, и $h[n]$ есть M-точечный сигнал с номерами от 0 до $M - 1$, свертка этих сигналов $y[n] = x[n] * h[n]$ будет сигналом из $N + M - 1$ точек с номерами от 0 до $N + M - 2$, определяемым следующей формулой:

$$y[i] = \sum_{j=0}^{M-1} h[j]x[i-j]. \qquad (1.1)$$

Операция свертки может быть реализована в виде алгоритма

```
function out = Svertka(Sign, h)
% Sign - фильтруемый сигнал
% h - импульсная характеристика фильтра

Numb_Sign = length(Sign);
Numb_h = length(h);
Sign = [Sign zeros(1,Numb_h-1)];

out = [];
for i = 1:Numb_Sign
% выделение части сигнала
    Sign_cut = Sign(i:i+Numb_h-1);
% умножение на имп. характ.
    Mult = Sign_cut.*h;
% нахождение выходного отсчета
    Filt = sum(Mult);
% добавление в выходной сигнал
    out = [out Filt];
end

out = out;
```

1.4 Практическая часть

1.4.1 Написание функции для построения кардиосигнала

Входными данными будут амплитуда кардиосигнала *A_ECG*, частота дискретизации сигналов *Fd* (Гц), частота сердечных сокращений *F_ECG* (уд/мин) и длительность требуемого сигнала *T_sign* (с).

Ключевые моменты:

1 При написании функции удобно использовать встроенную функцию *MATLAB ecg*(*n*). Данная функция строит один период ЭКГ-сигнала длиной *n* точек.

2 Предварительно необходимо вычислить количество кардиоциклов *N_ECG* в сигнале. Для этого вычисляется длительность одного кардиоцикла как *T_ECG* = 60 / *F_ECG* и затем *N_ECG* = *round*(*T_sign*/*T_ECG*). Функция *round* округляет результат деления.

3 Количество точек в одном кардиоцикле (*n*) определяется исходя из времени кардиоцикла частоты дискретизации: *n* = *round*(*T_ECG***Fd*).

4 После выполнения предварительных расчетов выполняется функция *ECG_1* = *ecg*(n), которая строит один кардиоцикл.

5 В цикле с индексом *i* = 1: *N_ECG* выполняется добавление в массив *Sign_ECG* значений *ECG*_1 (по аналогии с примерами 1–3).

6 Результат работы цикла необходимо умножить на заданную амплитуду кардиосигнала (*A_ECG*) и присвоить выходной переменной.

7 Полученный сигнал рекомендуется построить в графическом окне по аналогии с примером 3 (рисунок 1.2).

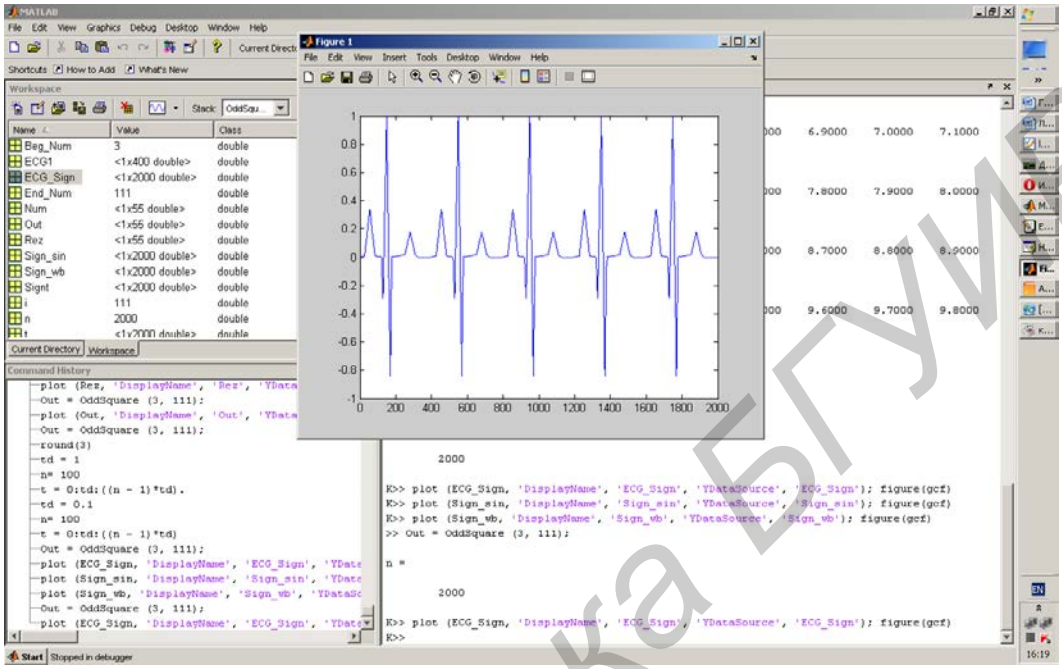


Рисунок 1.2 – Построение сигнала ЭКГ

1.4.2 Написание функции для построения сигнала белого шума

Входными данными будут амплитуда сигнала «белого» шума *A_WN* и количество точек требуемого сигнала *N_Wn*.

Ключевые моменты:

1 При написании функции удобно использовать встроенную функцию *MATLAB rand* (*m*, *n*). Данная функция используется для создания массивов случайных чисел с равномерным законом распределения. При этом *m* – количество строк, *n* – количество столбцов. Таким образом, для формирования сигнала необходимо сделать вектор (1 строка, *N_WN* столбцов) *Sign_WN* = *rand* (1, *N_WN*).

2 Значения полученного сигнала лежат в пределах от 0 до 1. Для того чтобы значения были от –1 до +1, необходимо умножить каждую точку сигнала на два (значения от 0 до 2) и отнять единицу (значения от –1 до +1): *Sign_WN* = 2∗*rand* (1, *N_WN*) – 1.

3 Результат работы функции необходимо умножить на заданную амплитуду сигнала «белого» шума (*A_WN*) и присвоить выходной переменной.

4 Полученный сигнал рекомендуется построить в графическом окне по аналогии с примером 3 (рисунок 1.3).

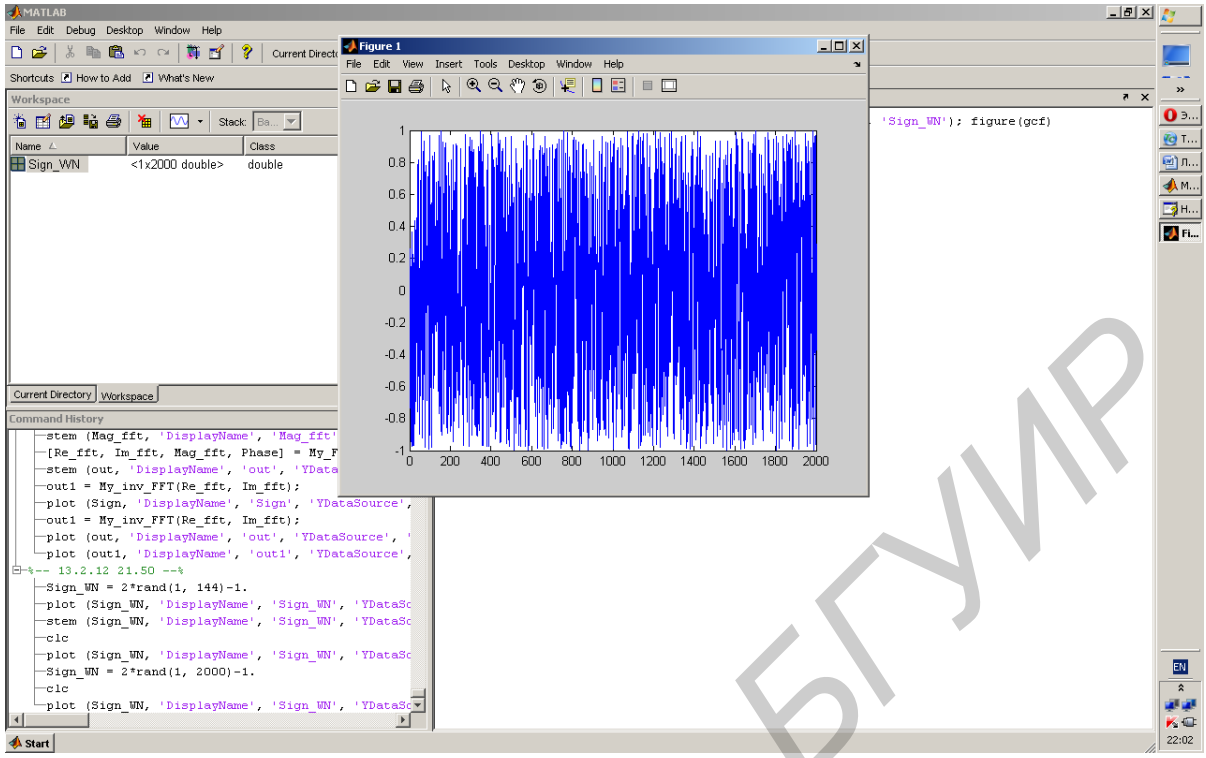


Рисунок 1.3 – Построение сигнала «белого» шума

1.4.3 Написание функции для построения сигнала гармонического шума

Входными данными будут амплитуда сигнала гармонического шума *A_sin*, частота дискретизации сигналов *Fd* (Гц), частота синусоидального сигнала *F_sin* (Гц) и количество точек требуемого сигнала *N_sin*.

Ключевые моменты:

1 При написании функции удобно использовать встроенную функцию *MATLAB sin* (2∗*pi*∗*F_sin*∗*t*). Данная функция используется для создания синусоидального сигнала.

2 Для формирования сигнала необходимо задать вектор *t*. Для этого необходимо вычислить дискрет времени *td*, обратно пропорциональный частоте дискретизации *td* = 1 / *Fd*. Далее вектор *t* задается как *t* = 0:*td*:((*N_sin* – 1)∗*td*).

3 Результат работы функции *sin* (2∗*pi*∗*F_sin*∗*t*) необходимо умножить на заданную амплитуду сигнала гармонического шума (*A_sin*) и присвоить выходной переменной.

4 Полученный сигнал рекомендуется построить в графическом окне по аналогии с примером 3 (рисунок 1.4).

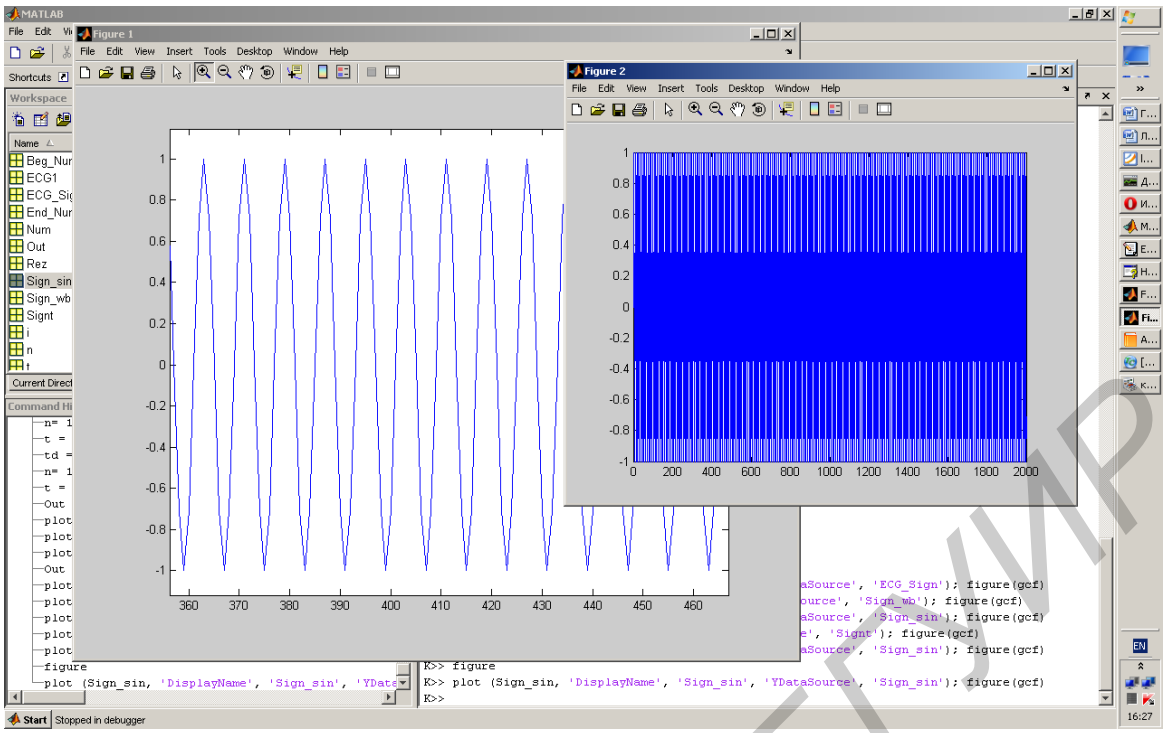


Рисунок 1.4 – Построение синусоидального сигнала

На выделенных фрагментах рисунка 1.4 построены соответственно часть сигнала и весь сигнал.

1.4.4 Написание функции для построения суммы всех полученных сигналов

Входными данными будут амплитуда кардиосигнала *A_ECG*, частота дискретизации сигналов *Fd* (Гц), частота сердечных сокращений *F_ECG* (уд/мин) и длительность требуемого сигнала *T_sign* (с), амплитуда сигнала «белого» шума *A_wn*, амплитуда сигнала гармонического шума *A_sin* и частота синусоидального сигнала *F_sin* (Гц).

Ключевые моменты:

1 В данной функции последовательно вызываются уже написанные функции для построения кардиосигнала, «белого» и гармонического шума.

2 При вызове функции построения кардиосигнала используются входные параметры функции.

3 При вызове функции построения «белого» шума параметр *A_wn* получается из входных параметров, параметр *N_wn* определяется как *length*(*Sign_ECG*). *Length* (вектор) – функция, которая выдает количество точек в векторе.

4 При вызове функции построения гармонического шума параметры *A_sin* и *Fd* получаются из входных параметров, параметр *N_sin* определяется как *length* (*Sign_ECG*). *Length* (вектор) – функция, которая выдает количество точек в векторе.

5 Итоговый сигнал получается как сумма трех полученных сигналов (рисунок 1.5). Необходимо обратить внимание на длину слагаемых сигналов. Она должна быть одинаковой, в противном случае при выполнении программы будет выдано сообщение об ошибке.

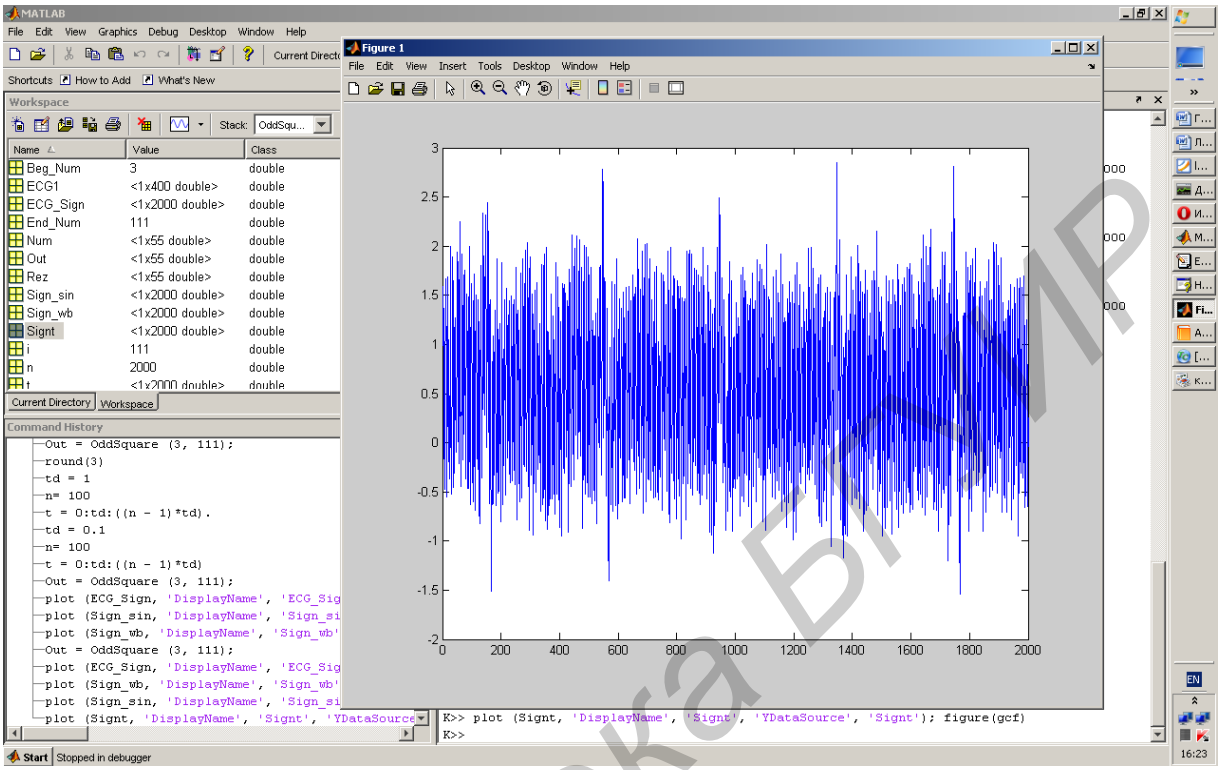


Рисунок 1.5 – Построение суммы полученных сигналов

1.4.5 Написание функции для свертки полученной суммы сигналов (зашумленного кардиосигнала)

Синтезируется зашумленный кардиосигнал с заданными импульсными характеристиками: дельта-функция, сдвинутая на *N* отсчетов, дельта-функция, ядро скользящего усредняющего фильтра и оконного *sinc*-фильтра.

Ключевые моменты:

1 Пример программы функции свертки приведен в теоретическом описании в пункте 1.3.6.

2 Импульсные характеристики необходимо получить у преподавателя в виде отдельного *m*-файла, параметры сигнала указаны в таблице 1.2.

3 В отчете необходимо построить полученные после свертки сигналы и проанализировать влияние импульсных характеристик на сигнал ЭКГ.

1.5 Содержание отчета

1 Цель работы.

2 Листинг программы генерации сигнала.

3 Распечатки экранов с результатами выполнения индивидуальных заданий.

4 Выводы по работе.

Таблица 1.2 – Варианты заданий к лабораторной работе №1

Вариант	*Fd*, Гц	*T_sign*, с	*F_ECG*, уд/мин	*A_ ECG*, мВ	*A_WB*, мВ	*A_sin*, мВ
1	2300	18	64	212	90	120
2	410	26	66	200	86	90
3	990	14	58	180	86	102
4	630	14	54	184	86	102
5	600	16	50	180	98	66
6	340	28	64	208	110	102
7	300	18	70	188	94	102
8	880	14	94	200	70	114
9	2300	14	62	204	70	114
10	940	28	68	200	82	102
11	830	22	62	204	74	66
12	760	28	56	192	98	66
13	400	26	66	196	90	72
14	1200	20	66	204	86	66
15	660	22	54	220	82	72
16	880	20	52	204	90	114
17	390	18	60	200	86	84
18	2000	22	52	184	86	96
19	500	16	62	200	86	72
20	340	26	64	196	94	60
21	1300	28	64	196	98	66
22	340	30	62	200	98	96
23	2200	22	74	200	94	108
24	910	14	64	188	90	96
25	750	18	78	204	102	102
26	1500	28	62	200	70	108
27	390	22	60	200	90	72
28	400	26	82	188	74	120
29	680	20	52	216	98	72
30	2900	28	68	208	94	102
31	540	22	58	220	70	72
32	840	14	70	200	94	90
33	2100	22	66	204	94	66
34	500	12	64	208	82	108
35	450	16	76	208	74	108

1.6 Контрольные вопросы

1 Какой вид имеет структура *m*-файла в системе *MATLAB*?
2 Для чего используется выходная переменная?
3 Как выполняются комментарии в программе?
4 Как используется управляющая структура *for...end*?

5 Как выполняется визуализация данных?

6 Какие существуют средства видоизменения графиков?

7 Каковы особенности применения встроенной функции *ecg*(*n*)?

8 Каковы особенности применения встроенной функции *rand*(1, *n*)?

9 Каковы особенности применения встроенной функции *sin*()?

Литература

1 Поршнев, С. В. *MATLAB* 7. Основы работы и программирования : учебник / С. В. Поршнев. – М. : Бином. Лаборатория знаний, 2006. – 320 с.

2 Иглин, С. П. Математические расчеты на базе *MATLAB* / С. П. Иглин. – СПб. : BHV-Санкт-Петербург, 2005. – 640 с.

3 Мартынов, Н. Н. *MATLAB* 7. Элементарное введение / Н. Н. Мартынов. – М. : Кудиц-Образ, 2005. – 416 с.

4 Сергиенко, А. Б. Цифровая обработка сигналов / А. Б. Сергиенко. – СПб. : Питер, 2005. – 604 с.

5 Лайонс, Р. Цифровая обработка сигналов / Р. Лайонс. – М. : ООО «Бином-пресс», 2006. – 656 с.

6 Основы цифровой обработки сигналов : курс лекций / А. И. Солонина [и др.]. – СПб. : БХВ-Петербург, 2005. – 768 с.

7 Васильев, В. П. Основы теории и расчета цифровых фильтров : учеб. пособие для высш. учеб. заведений / В. П. Васильев, Э. Л. Муро, С. М. Смольский. – М. : Издательский центр «Академия», 2007. – 272 с.

8 Айфичер, Э. С. Цифровая обработка сигналов : практический подход / Э. С. Айфичер, Б. У. Джервис. – М. : Издательский дом «Вильямс», 2008. – 992 с.

Лабораторная работа №2

Спектральный анализ биомедицинских сигналов

2.1 Цель работы

Изучение особенностей реализации алгоритмов преобразования Фурье в среде *MATLAB*.

2.2 Задачи

1 Разработать функцию прямого преобразования Фурье методом корреляции заданных сигналов с базисными функциями ДПФ.

2 Разработать функцию обратного преобразования Фурье.

3 Разработать функцию для преобразования спектральных коэффициентов из прямоугольной системы координат в полярную.

4 Изучить спектры кардиосигнала, гармонического и «белого» шумов, построенных при выполнении лабораторной работы №1.

5 Изучить пары преобразования Фурье.

6 Исследовать взаимосвязь заданных временных и спектральных параметров сигналов.

2.3 Общая теория. Программирование в системе *MATLAB*

2.3.1 Базовые функции ДПФ

Синусы и косинусы, используемые в ДПФ, обычно называются базовыми функциями. Другими словами, выход ДПФ есть набор чисел, которые представляют амплитуды. Базовые функции есть набор синусов и косинусов с единичной амплитудой. Если вы определили каждую амплитуду (в частотной области) соответствующего синуса или косинуса (базовые функции), то в результате вы получите масштабированные синусы и косинусы, сумма которых сможет сформировать сигнал во временной области.

Базовые функции ДПФ определяются следующими выражениями:

$$c_k[i] = \cos(2\pi ki/N),$$
$$s_k[i] = \sin(2\pi ki/N), \qquad (2.1)$$

где $c_k[\ ]$ и $s_k[\ ]$ – косинусные и синусные волны, каждая из N точек с номерами от 0 до N – 1. Параметр k определяет частоту волны. В N-точечном ДПФ k имеет номера от 0 до N/2.

Например, рисунок 2.1 показывает некоторые из 17 синусов и 17 косинусов, используемых в 32-точечном ДПФ. Поскольку эти синусоиды складываются для формирования входного сигнала, они должны быть той же самой длины, что и входной сигнал. В этом случае каждая синусоида имеет 32 точки с номерами от 0 до 31. Параметр k определяет частоту каждой синусоиды. Практически $c_1[\ ]$ –косинусная волна, которая совершает один полный цикл колебаний на

N точках, $c_5[\ ]$ – косинусная волна с пятью полными колебаниями на N точках и т. д. Это важная концепция в понимании базовых функций; частотный параметр k равен числу полных колебаний (циклов), приходящихся на N точек сигнала.

На рисунке 2.1 32-точечное ДПФ имеет 17 дискретных косинусных волн и 17 дискретных синусных волн в качестве базовых функций. Показаны гармоники под номерами 0, 2, 10, 16. Эти сигналы дискретны, непрерывные линии помогают отслеживать формы волны.

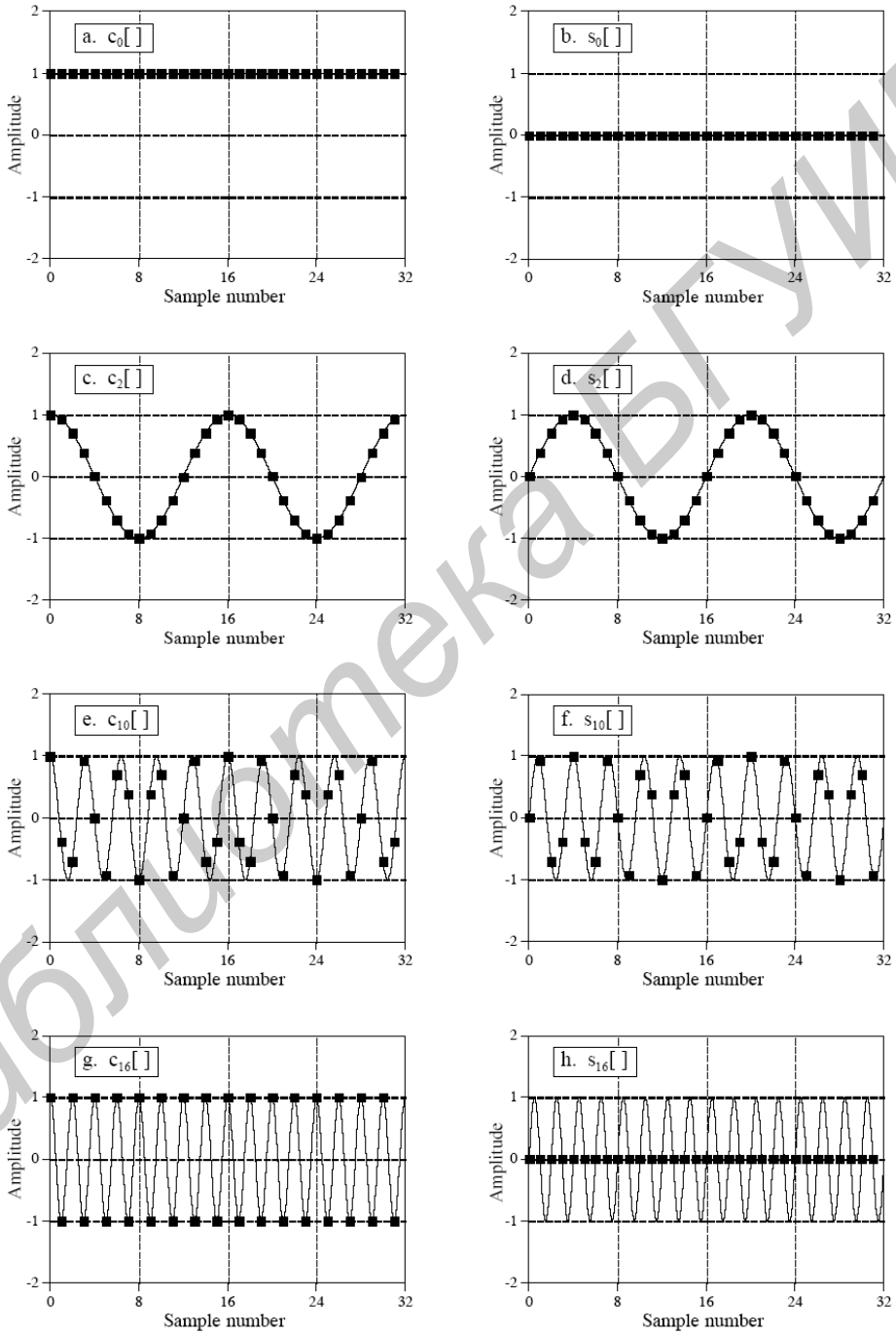


Рисунок 2.1 – Базовые функции ДПФ

2.3.2 Анализ, вычисление ДПФ с помощью корреляции

Данная процедура формализуется в следующем *уравнении анализа*, математических формулах для нахождения сигналов частотной области по сигналам временной области:

$$\begin{aligned} \operatorname{Re} X[k] &= \sum_{i=0}^{N-1} x[i]\cos(2\pi ki / N), \\ \operatorname{Im} X[k] &= -\sum_{i=0}^{N-1} x[i]\sin(2\pi ki / N), \end{aligned} \tag{2.2}$$

где Re $X[k]$ и Im $X[k]$ – вычисляемые сигналы частотной области;

$x[i]$ – анализируемый сигнал временной области;

i – изменяется от 0 до $N - 1$;

k – изменяется от 0 до $N/2$.

Каждый отсчет в частотной области находится перемножением сигнала временной области на рассматриваемые синусные и косинусные волны и суммированием результатов умножения. Таким образом находится корреляция входного сигнала с базовыми функциями. Коэффициенты корреляции показывают, насколько сигнал «похож» на ту или иную базовую функцию.

```
%============ Вычисление спектра сигнала ============
N = length(Sign);    %Определяем длину сигнала
Re_fft = [];         %Пустые массивы спектральных коэффициентов
Im_fft = [];
for k =0:N/2         %Цикл для прохода по всем базовым функциям
    Re_basis = [];   %Пустой массив k-й базовой функции
    Im_basis = [];
    for i = 0:(N-1) %Цикл расчета k-й базовой функции
        Re_basis = [Re_basis cos(2*pi*k*i/N)];
        Im_basis = [Im_basis sin(2*pi*k*i/N)];
    end
    %Расчет корреляции сигнала и k-й базовой функции
    Re_cor = sum(Re_basis.*Sign);
    Re_fft = [Re_fft Re_cor];
    Im_cor = sum(Im_basis.*Sign);
    Im_fft = [Im_fft Im_cor];
end
```

2.3.3 Синтез, вычисление инверсного (обратного) ДПФ

Вычисление инверсного ДПФ выполняется по формуле

$$x[i] = \sum_{k=0}^{N/2} \operatorname{Re}\overline{X}[k]\cos(2\pi ki / N) + \sum_{k=0}^{N/2} \operatorname{Im}\overline{X}[k]\sin(2\pi ki / N), \tag{2.3}$$

где $x[i]$ – синтезируемый сигнал;

i – изменяется от 0 до $N/2$;

$\mathrm{Re}\,\overline{X}\,[k]$ и $\mathrm{Im}\,\overline{X}\,[k]$ – амплитуды косинусов и синусов соответственно;

k – изменяется от 0 до $N/2$.

Иными словами, любой сигнал из N точек $x[i]$ может быть создан суммированием $N/2 + 1$ косинусных волн и $N/2 + 1$ синусных волн. Амплитуды косинусных и синусных волн содержатся в массивах $\mathrm{Re}\,\overline{X}\,[k]$ и $\mathrm{Im}\,\overline{X}\,[k]$ соответственно. В уравнении синтеза эти амплитуды перемножаются на базовые функции, создавая набор масштабированных синусных и косинусных волн. Сложение этих масштабированных синусных и косинусных волн формирует сигнал временной области $x[i]$.

В выражении (2.3) массивы называются $\mathrm{Im}\,\overline{X}\,[k]$ и $\mathrm{Re}\,\overline{X}\,[k]$ вместо $\mathrm{Im}\,X\,[k]$ и $\mathrm{Re}\,X\,[k]$. Это вызвано тем, что амплитуды, необходимые для синтеза ($\mathrm{Im}\,\overline{X}\,[k]$ и $\mathrm{Re}\,\overline{X}\,[k]$), слегка отличаются от сигналов частотной области ($\mathrm{Im}\,X\,[k]$ и $\mathrm{Re}\,X\,[k]$). В формульном выражении преобразование выглядит следующим образом:

$$\begin{aligned} \mathrm{Re}\,\overline{X}[k] &= \frac{\mathrm{Re}\,X[k]}{N/2}, \\ \mathrm{Im}\,\overline{X}[k] &= -\frac{\mathrm{Im}\,X[k]}{N/2}. \end{aligned} \qquad (2.4)$$

За исключением двух специальных случаев:

$$\mathrm{Re}\,\overline{X}[0] = \frac{\mathrm{Re}\,X[0]}{N},$$

$$\mathrm{Re}\,\overline{X}[N/2] = \frac{\mathrm{Re}\,X[N/2]}{N}.$$

Преобразование сигналов частотной области в амплитуды синусов, необходимых для синтеза сигнала временной области. В рассматриваемых примерах N – число точек в сигнале, k имеет нумерацию от 0 до $N/2$.

Предположим, вам представлена частотная область, и предлагается синтезировать сигнал временной области. Для начала необходимо найти амплитуды синусов и косинусов. Другими словами, взяв $\mathrm{Im}\,X\,[k]$ и $\mathrm{Re}\,X\,[k]$, необходимо найти $\mathrm{Im}\,\overline{X}\,[k]$ и $\mathrm{Re}\,\overline{X}\,[k]$. Формула (2.4) показывает, как это сделать математически. В компьютерной программе необходимо выполнить три действия: разделить все величины в частотной области на $N/2$; изменить знак перед всеми мнимыми величинами; разделить первую и последнюю величину в реальной части – $\mathrm{Re}\,X\,[0]$ и $\mathrm{Re}\,X\,[N/2]$ – на два. Это обеспечит величину амплитуд, необходимую для синтеза по формуле (2.3). Взятые вместе формулы (2.3) и (2.4), определяют инверсное ДПФ.

Пример программы для вычисления инверсного ДПФ

```
%=============== Подготовительные операции ===============
```

```
F = length(Re_fft); %Определяем длину спектра
N = 2*(F-1);        %Определяем длину сигнала
%Нормализация коэффициентов спектра
Re_fft_norm = Re_fft/(N/2);
Im_fft_norm = Im_fft/(N/2);
Re_fft_norm(1) = Re_fft_norm(1)/2;
Re_fft_norm(F) = Re_fft_norm(F)/2;
%=============== Синтез сигнала ===============
Sign = zeros(1,N);  %Создаем пустой сигнал
for k =0:F-1        %Цикл для прохода по всем базовым функциям
    Re_basis = [];  %Пустой массив k-й базовой функции
    Im_basis = [];
    for i = 0:(N-1) %Цикл расчета k-й базовой функции
        Re_basis = [Re_basis  cos(2*pi*k*i/N)];
        Im_basis = [Im_basis  sin(2*pi*k*i/N)];
    end
    %Расчет сигнала как суммы масштабированных базовых функций
    Re_temp = Re_basis.*Re_fft_norm(k+1);
    Im_temp = Im_basis.*Im_fft_norm(k+1);
    Sign = Sign + Re_temp + Im_temp;
end
Out = Sign;
```

2.3.4 Полярная система координат

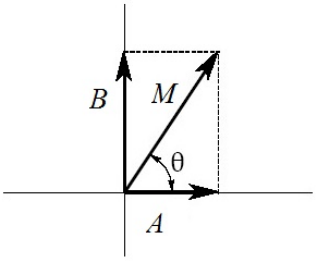
Частотная область есть группа амплитуд косинусных и синусных волн (с учетом масштабирования). Это называется **прямоугольной** системой координат. Альтернативным способом частотная область может быть представлена в **полярной** системе координат. В этой системе координат Re*X*[ ] и Im*X*[ ] преобразуются в два других массива, называемых **амплитуда *X*** (*magnitude*), которая записывается в формулах как **Mag *X*[ ]**, и **фаза *X***, которая записывается как **Phase *X*[ ]**. Амплитуда и фаза попарно вычисляются из реальной и мнимой частей. Например, Mag *X*[0] и Phase *X*[0] вычисляются с использованием только Re *X*[0] и Im *X*[0]. Подобным образом Mag *X*[14] и Phase *X*[14] вычисляются с использованием только Re *X*[14] и Im *X*[14] и т. д. Для понимания этих преобразований посмотрим, что происходит, когда вы складываете косинусную волну и синусную волну в частотной области. Результат есть косинусная волна той же самой частоты, но с новой амплитудой и новым фазовым сдвигом. В виде формулы это выглядит так:

$$A\cos(x) + B\sin(x) = M\cos(x + \theta). \qquad (2.5)$$

Сложение косинуса и синуса дает косинус с другими амплитудой и фазовым сдвигом. Информация, содержащаяся в A и B, преобразуется в две переменные M и θ.

Информация в процессе преобразования не теряется. Другими словами, информация, содержащаяся в амплитудах A и B, также содержится в величинах M и θ. Хотя формула содержит косинусы и синусы, она полностью соответ-

ствует преобразованию простых векторов. Рисунок 2.2 показывает векторную аналогию, переменные A и B могут рассматриваться в прямоугольной системе координат, в то время как переменные M и θ есть параметры полярной системы координат.



$$M = \left(A^2 + B^2\right)^{1/2}$$
$$\theta = \arctan\left(B/A\right)$$

Рисунок 2.2 – Преобразование прямоугольной системы координат в полярную

В полярной системе координат Mag $X[\ ]$ является амплитудой косинусной волны (M в формуле (2.5) и рисунке 2.2), Phase $X[\ ]$ – фазой угла косинусной волны (θ в формуле (2.5) и рисунке 2.2). Следующие формулы обеспечивают преобразование прямоугольной системы координат в полярную систему координат и наоборот.

Преобразование полярной системы координат (Mag $X[k]$, Phase $X[k]$) в прямоугольную (Re $X[k]$, Im $X[k]$) имеет следующий вид:

$$\begin{aligned} \text{Mag } X[k] &= (\text{Re } X[k]^2 + \text{Im } X[k]^2)^{1/2}, \\ \text{Phase } X[k] &= \arctan \frac{\text{Im } X[k]}{\text{Re } X[k]}. \end{aligned} \tag{2.6}$$

Преобразование прямоугольной системы координат (Re $X[k]$, Im $X[k]$) в полярную (Mag $X[k]$, Phase $X[k]$) имеет вид

$$\begin{aligned} \text{Re } X[k] &= \text{Mag } X[k] \cos(\text{Phase } X[k]), \\ \text{Im } X[k] &= \text{Mag } X[k] \sin(\text{Phase } X[k]). \end{aligned} \tag{2.7}$$

Прямоугольная система координат и полярная система координат позволяют рассматривать ДПФ с двух точек зрения. В полярной системе координат ДПФ раскладывает N-точечный сигнал в $N/2 + 1$ косинусных волн и $N/2 + 1$ синусных волн, каждая со своей *амплитудой*. В полярной системе координат ДПФ раскладывает N-точечный сигнал в $N/2 + 1$ косинусных волн, каждая из которых определяется *амплитудой* и *фазовым сдвигом*.

Прямоугольная система координат больше всего подходит для вычислений, как в формульном виде, так и в компьютерной программе. Графики почти всегда отображаются в полярной системе координат. Как показывает практика, это более наглядно для человеческого восприятия, чем реальная и мнимая части.

2.3.5 Особенности полярной системы координат

Особенность 1. Деление на нуль

Когда преобразуется прямоугольная система координат в полярную, очень часто встречаются частоты с реальной частью, равной нулю, и мнимой частью, не равной нулю. Это значит, что в таких случаях фаза точно равна 90° или минус 90°. Но когда программа пытается вычислить фазу по формуле Phase $X[k]$ = arctan(Im $X[k]$/Re $X[k]$), появляется ошибка *деления на нуль*. Даже если программа не остановится, фаза для этой частоты будет неверной. Чтобы избежать эту проблему, перед делением необходимо проверить реальную часть на наличие нулей. Если нуль обнаружен, то необходимо проверить мнимую часть на положительное и отрицательное значения для определения значения фазы $\pi/2$ или минус $\pi/2$ соответственно. Альтернативный путь решения этой проблемы – если реальная часть равна нулю, заменить ее очень маленькой величиной (например 10^{-20}).

Особенность 2. Некорректный арктангенс

Представьте отсчет частотной области, у которого Re $X[k] = 1$ и Im $X[k] = 1$. Формула (2.6) даст соответствующую полярную величину Mag $X[k] = 1{,}414$ и Phase $X[k] = 45°$. Теперь рассмотрим другой отсчет, у которого Re $X[k] = -1$ и Im $X[k] = -1$. Снова формула (2.6) даст Mag $X[k] = 1{,}414$ и Phase $X[k] = 45°$. Но фаза должна быть минус 135°. Это ошибка появляется, когда реальная часть отрицательная. Проблема может быть исправлена тестированием реальной и мнимой части после вычисления фазы. Если реальная и мнимая частей отрицательные, то надо вычесть 180° (или π радиан) из вычисленной фазы. Если реальная часть отрицательная, а мнимая положительная, то добавить 180° (или π радиан). Если не учесть эту проблему, то вычисленные фазы будут находиться между минус $\pi/2$ и $\pi/2$, вместо минус π и π.

```
%------------Переход в полярную систему координат------------
for k =0:N/2        %Цикл для прохода по всем базовым функциям
    %вычисление амплитуды
    Mag_fft(k+1) = sqrt(Re_fft(k+1)^2 + Im_fft(k+1)^2);

    %проверка реальной части на нуль (Особенность 1)
    if (Re_fft(k+1)==0)
```

```
        Re_fft(k+1)=10^-20;
    End
    %вычисление фазы
    Phase(k+1) = atan(Im_fft(k+1)/Re_fft(k+1));

    %проверка знаков реальной и мнимой частей (Особенность 2)
    if (Re_fft(k+1)<0)&(Im_fft(k+1)<0)
        Phase(k+1) = Phase(k+1) - pi;
    end
    if (Re_fft(k+1)<0)&(Im_fft(k+1)>=0)
        Phase(k+1) = Phase(k+1) + pi;
    end
end
```

Особенность 3. Фаза очень маленькой амплитуды

Иногда соответствующая величина амплитуды слишком мала, и она теряется в шуме округления. Для частот, у которых амплитуда падает до очень низкой величины, шум округления вызывает выбросы фаз. Пример показан на рисунке 2.3.

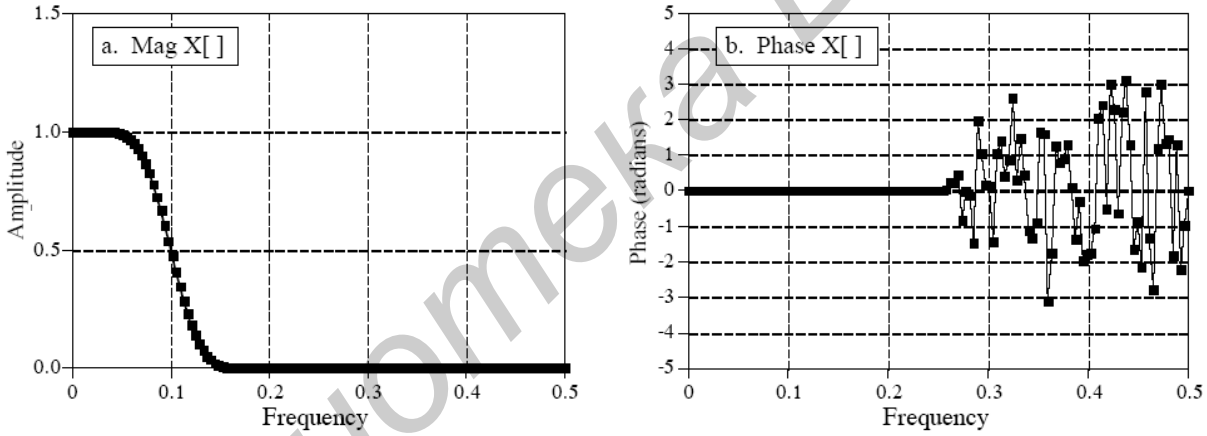


Рисунок 2.3 – Фаза сигнала с маленькой амплитудой

Особенность 4. Неоднозначность фазы 2π

На рисунке 2.4 наблюдается прерывистость данных. Каждый раз точка выглядит так, как будто она собирается «нырнуть» ниже минус 3,141592 и резко «прыгает» к 3,141592. Это есть результат периодической природы синусоиды. Например, фазовый сдвиг θ есть то же самое, что и фазовый сдвиг θ + 2π, θ + 4π, θ + 6π и т. д. Любая синусоида не изменяется, когда к фазе добавляется число, кратное 2π. Видимая прерывистость в сигнале есть результат компьютерных алгоритмов, осуществляющих предпочтительный отбор из бесконечного числа эквивалентных возможностей. Всегда выбирается самая маленькая величина, сохраняющая фазу между минус π и π.

Очень часто легче определить фазу, если нет этой прерывистости, даже если это означает, что фаза будет превышать π или буде ниже минус π. Это называется **разверткой фазы (*unwrapping the phase*)** (рисунок 2.4). Как показано в программе, 2π многократно добавляется или вычитается из каждой величины фазы. Точная величина фазы определяется алгоритмом, который минимизирует разницу между соседними отсчетами. Верхняя кривая показывает типичный сигнал фазы, полученный при преобразовании прямоугольной системы координат в полярную. Каждая величина сигнала должна находиться между минус π и π. Нижняя кривая показывает развернутую фазу.

```
%======== Устранение прерывистости фазы (Особенность 4)
Uw_Phase = 0*Phase;
for k =2:N/2+1     %Цикл для прохода по всем базовым функциям
    C = round((Uw_Phase(k-1)-Phase(k))/(2*pi));
    Uw_Phase(k) = Phase(k) + C*2*pi;
end
```

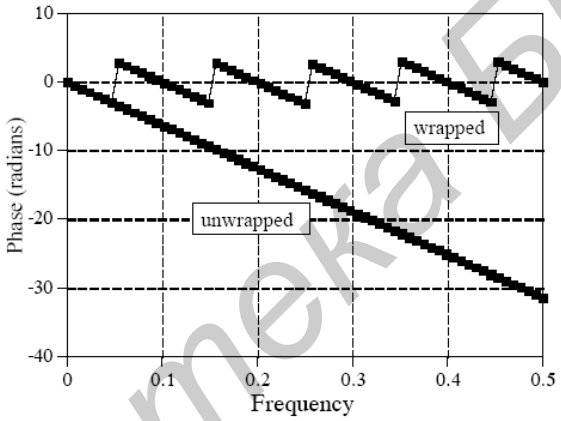


Рисунок 2.4 – Пример фазовой развертки

Особенность 5. Амплитуда всегда положительная (неоднозначность фазы π)

Рисунок 2.5 показывает сигнал частотной области в прямоугольной и полярной формах. Реальная часть гладкая и вполне легкая для понимания, мнимая часть целиком состоит из нулей. Сигнал в полярной форме содержит резкие изломы и острые углы. Это вызвано тем, что амплитуда в полярной форме (*magnitude*) должна быть положительной *по определению*. Хотя реальная часть падает ниже нуля, амплитуда в полярной форме остается положительной за счет изменения фазы на π (или минус π, что одно и то же). Это не проблема с точки зрения математики, но нерегулярная кривая может быть более трудной для понимания.

Особенность 6. Выбросы между π и минус π

Так как π и минус π представляют одну и ту же фазу, шум округления может быть причиной резкого скачка фазы между этими величинами у соседних

отсчетов. Как показано на рисунке 2.5, это может приводить к острым изломам и выбросам. Однако фаза в действительности непрерывна. При обработки сигнала необходимо помнить, что случайный шум может вызывать быстрые скачки фазы между π и минус π.

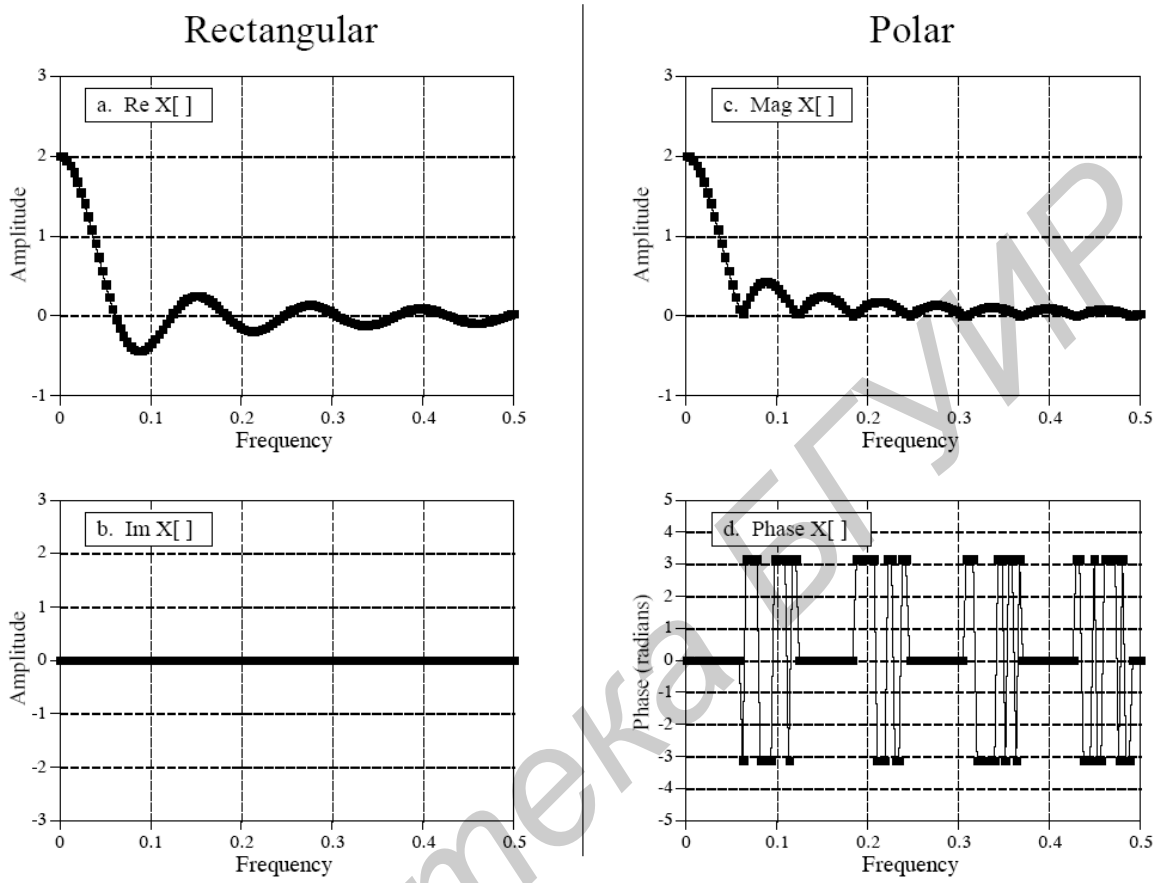


Рисунок 2.5 – Пример сигнала в прямоугольной и полярной формах

2.4 Практическая часть

2.4.1 Разработка функции прямого преобразования Фурье методом корреляции заданных сигналов с базисными функциями ДПФ

На основе примера, приведенного в данной лабораторной работе, разработать собственную функцию, входным параметром которой будет сигнал (во временной области), выходным параметром – спектр входного сигнала. Код функции приводится в отчете по лабораторной работе.

2.4.2 Разработка функции обратного преобразования Фурье

На основе примера, приведенного в данной лабораторной работе, разработать собственную функцию, входным параметром которой будет спектр сигнала, выходным параметром – сигнал во временной области. Выполнить проверку

функций: задать сигнал, выполнить прямое преобразование Фурье, выполнить обратное преобразование Фурье, сравнить исходный и полученный сигналы. Если сигналы идентичны – функции работают корректно. Код функции приводится в отчете по лабораторной работе.

2.4.3 Разработка функции для преобразования спектральных коэффициентов из прямоугольной системы координат в полярную

Учесть особенности полярной системы координат. Для разработки использовать приведенные в данной лабораторной работе примеры кода. Код функции приводится в отчете по лабораторной работе.

2.4.4 Изучение спектров кардиосигнала, гармонического и «белого» шумов, построенных при выполнении лабораторной работы №1

В отчете построить спектры указанных сигналов по отдельности, а также общий спектр суммы сигналов с выделением каждого из спектров.

2.4.5 Изучение пары преобразования Фурье

1 Построить дельта-функцию и исследовать ее спектр в прямоугольной и полярной системах координат. Для построения дельта-функции используется функция zeros (1, *n*), после этого самый первый отсчет приравнивается 1. Количество точек сигнала $n = 500$.

2 Построить дельта-функцию со сдвигом (согласно варианту) и исследовать ее спектр в прямоугольной и полярной системах координат. Построение аналогично предыдущему пункту.

3 Построить синусоидальные сигналы заданной частоты (согласно варианту) и исследовать их спектр в прямоугольной и полярной системах координат. Исследовать спектр суммы полученных синусоидальных сигналов в полярной системе координат. Количество точек сигнала $n = 500$.

4 Построить прямоугольные сигналы с заданными в варианте параметрами и исследовать их спектр в полярной системе координат Для построения прямоугольного сигнала используются функции one (1, *m*) и zeros (1, *n*). При этом *m* – количество единичных отсчетов, *n* – количество нулевых отсчетов.

5 Построить *sinc*-сигналы с заданными в варианте параметрами и исследовать их спектр в полярной системе координат. Для построения *sinc*-сигнала использовать формулу

$$x[i] = \frac{1}{N} \frac{\sin(2\pi i(M - 1/2)/N)}{\sin(\pi i / N)}.$$

В частотной области импульс имеет единичную амплитуду от 0 до $M - 1$. Параметр N есть длина ДПФ, $x[i]$ есть сигнал временной области с i, меняющимся от 0 до $N - 1$. Чтобы избежать деления на нуль, $x[0] = (2M - 1)/N$.

Для всех вариантов использовать значение $N = 500$.

Пример программы для построения sinc-сигнала

```
function out=My_sinc(M, N)
i = 1:N-1;                          %задаем точки сигнала
temp1 = sin(2*pi*i*(M-1/2)/N);%вычисляем делимое
temp2 = sin(pi*i/N);                %вычисляем делитель
out = (temp1/temp2)/N;
out = [(2*M-1)/N out];              %добавляем нулевой член массива
```

2.4.6 Исследование взаимосвязи заданных временных и спектральных параметров сигналов

Все полученные сигналы и спектры привести в отчете. Проанализировать взаимосвязь временных и спектральных параметров построенных сигналов. Результаты анализа отразить в выводах.

2.5 Содержание отчета

1 Цель работы.

2 Таблица по варианту (вариант определяется двумя последними цифрами номера студенческого билета) (таблица 2.1).

3 Листинги разработанных программ.

4 Результаты исследования спектров сигналов.

5 Выводы.

Таблица 2.1 – Варианты заданий к лабораторной работе №2

Вариант	Сдвиг дельта-функции	Частоты синусоидальных сигналов, F (Гц)		Параметры прямоугольного сигнала: m, n				Параметры *sinc*-функции, M	
	n	$F1$	$F2$	$m1$	$n1$	$m2$	$n2$	$M1$	$M2$
1	3	15	21	3	497	15	485	3	9
2	5	18	34	5	495	25	475	5	15
3	7	17	36	7	493	35	465	7	21
4	6	23	38	6	494	30	470	6	18
5	8	24	37	8	492	40	460	8	24
6	9	26	35	9	491	45	455	9	27
7	12	25	31	12	488	60	440	12	36
8	14	27	32	14	486	70	430	14	42
9	16	29	46	16	484	80	420	16	48
10	15	28	42	15	485	75	425	15	45
11	18	21	43	18	482	90	410	18	54

Продолжение таблицы 2.1

Вариант	Сдвиг дельта-функции	Частоты синусоидальных сигналов, F (Гц)		Параметры прямоугольного сигнала: m, n				Параметры *sinc*-функции, M	
	n	$F1$	$F2$	$m1$	$n1$	$m2$	$n2$	$M1$	$M2$
12	17	34	44	17	483	85	415	17	51
13	23	36	56	23	477	115	385	23	69
14	24	38	54	24	476	120	380	24	72
15	26	37	58	26	474	130	370	26	78
16	25	35	57	25	475	125	375	25	75
17	27	31	3	27	473	135	365	27	81
18	29	32	5	29	471	145	355	29	87
19	28	46	7	28	472	140	360	28	84
20	21	42	6	21	479	105	395	21	63
21	34	43	8	34	466	170	330	34	102
22	36	44	9	36	464	180	320	36	108
23	38	56	12	38	462	190	310	38	114
24	37	54	14	37	463	185	315	37	111
25	35	58	16	35	465	175	325	35	105
26	31	57	15	31	469	155	345	31	93
27	32	3	18	32	468	160	340	32	96
28	46	5	17	46	454	230	270	46	138
29	42	7	23	42	458	210	290	42	126
30	43	6	24	43	457	215	285	43	129
31	44	8	26	44	456	220	280	44	132
32	56	9	25	56	444	280	220	56	168
33	54	12	27	54	446	270	230	54	162
34	58	14	29	58	442	290	210	58	174
35	57	16	28	57	443	285	215	57	171

2.6 Контрольные вопросы

1 Какие базовые функции используются при ДПФ?
2 Каковы особенности базовых функций ДПФ?
3 Как выполняется вычисление ДПФ с помощью корреляции?
4 Как выполняется вычисление инверсного (обратного) ДПФ?
5 Как выполняется нормализация коэффициентов ДПФ ?
6 Каковы особенности полярной системы координат?
7 Каковы особенности пары ДПФ дельта-функция – синус?
8 Каковы особенности пары ДПФ прямоугольный импульс – *sinc*-функция?

Литература

1 Сергиенко, А. Б. Цифровая обработка сигналов / А. Б. Сергиенко. – СПб. : Питер, 2005. – 604 с.

2 Лайонс, Р. Цифровая обработка сигналов / Р. Лайонс. – М. : ООО «Бином-пресс», 2006. – 656 с.

3 Основы цифровой обработки сигналов : курс лекций / А. И. Солонина [и др.]. – СПб. : БХВ-Петербург, 2005. – 768 с.

4 Айфичер, Э. С. Цифровая обработка сигналов : практический подход / Э. С. Айфичер, Б. У. Джервис. – М. : Издательский дом «Вильямс», 2008. – 992 с.

Лабораторная работа №3

Проектирование цифрового фильтра и исследование его характеристик

3.1 Цель работы

Изучение особенностей реализации алгоритмов цифровых фильтров в среде *MATLAB*.

3.2 Задачи

1 Разработать функцию для скользящего усредняющего фильтра.

2 Изучить влияние скользящего усредняющего фильтра на различные виды шума.

3 Разработать функцию расчета ядра НЧ *sinc*-фильтра.

4 Разработать функцию свертки сигнала с импульсной характеристикой фильтра.

5 Изучить влияние *sinc*-фильтра на различные виды шума.

6 Создать набор рекурсивных коэффициентов с заданными параметрами в *fdatool*.

7 Разработать функцию для фильтрации с помощью БИХ фильтра.

3.3 Общая теория

3.3.1 Реализация скользящего усредняющего фильтра

Как следует из названия, скользящий усредняющий фильтр осуществляет усреднение некоторого числа точек из входного сигнала для вычисления каждой точки в выходном сигнале. В виде формулы это записывается так:

$$y[i] = \frac{1}{M} \sum_{j=0}^{M-1} x[i + j], \tag{3.1}$$

где $y[\ ]$ – выходной сигнал;

M – число усредняемых точек;

$x[\ ]$ – входной сигнал.

Например, в пяти усредняемых точках выходной сигнал под номером 80 будет равен

$$y[80] = \frac{x[80] + x[81] + x[82] + x[83] + x[84]}{5}.$$

Это соответствует изменению суммирования в формуле с $j = 0$ до $M - 1$ на $j = -(M - 1)/2$ до $(M - 1)/2$. Например, в 11-точечном скользящем усредняющем фильтре индекс j будет меняться от 0 до 11 (односторонне усреднение) или от минус 5 до 5 (симметричное усреднение). Симметричное усреднение требует, чтобы M было нечетным числом. Программирование проще производить при

одностороннем усреднении, но это приводит к относительному сдвигу между входным и выходным сигналом.

Таким образом, скользящий усредняющий фильтр – это свертка, использующая очень простое ядро фильтра. Например, 5-точечный фильтр имеет ядро …0, 0, 1/5, 1/5, 1/5, 1/5, 1/5, 0, 0… . То есть скользящий усредняющий фильтр есть свертка входного сигнала с прямоугольным импульсом, имеющего площадь, равную единице.

Программное выполнение скользящего усредняющего фильтра приведено ниже.

```
function out = AvrFilt(Sign, n)
% Sign - фильтруемый сигнал
% h - импульсная характеристика фильтра

Numb_Sign = length(Sign);

out = [];
for i = 1:Numb_Sign-n
    % выделение части сигнала
    Sign_cut = Sign(i:i+n);
    % нахождение выходного отсчета
    Filt = (1/n)*sum(Sign_cut);
    % добавление в выходной сигнал
    out = [out Filt];
end

out = out;
```

3.3.2 Параметры усредняющего фильтра

Уменьшение шума и ступенчатый отклик

Усредняющий скользящий фильтр *оптимален* для типичной проблемы уменьшения случайного «белого» шума при одновременном сохранении резкости ступенчатого отклика.

На рисунке 3.1 графически показана работа фильтра. Сигнал «а» (*Original signal*) есть всплеск импульса в случайном шуме. Показан результат сглаживающего действия скользящего усредняющего фильтра: уменьшается амплитуда случайного шума (хорошо) и уменьшается резкость краев (плохо). Из всех возможных фильтров скользящий усредняющий обеспечивает самый низкий шум для данного уменьшения резкости краев. Суммарное уменьшение шумов равно корню квадратному из числа точек усреднения. Например, при 100 точках усреднения шум уменьшится в 10 раз.

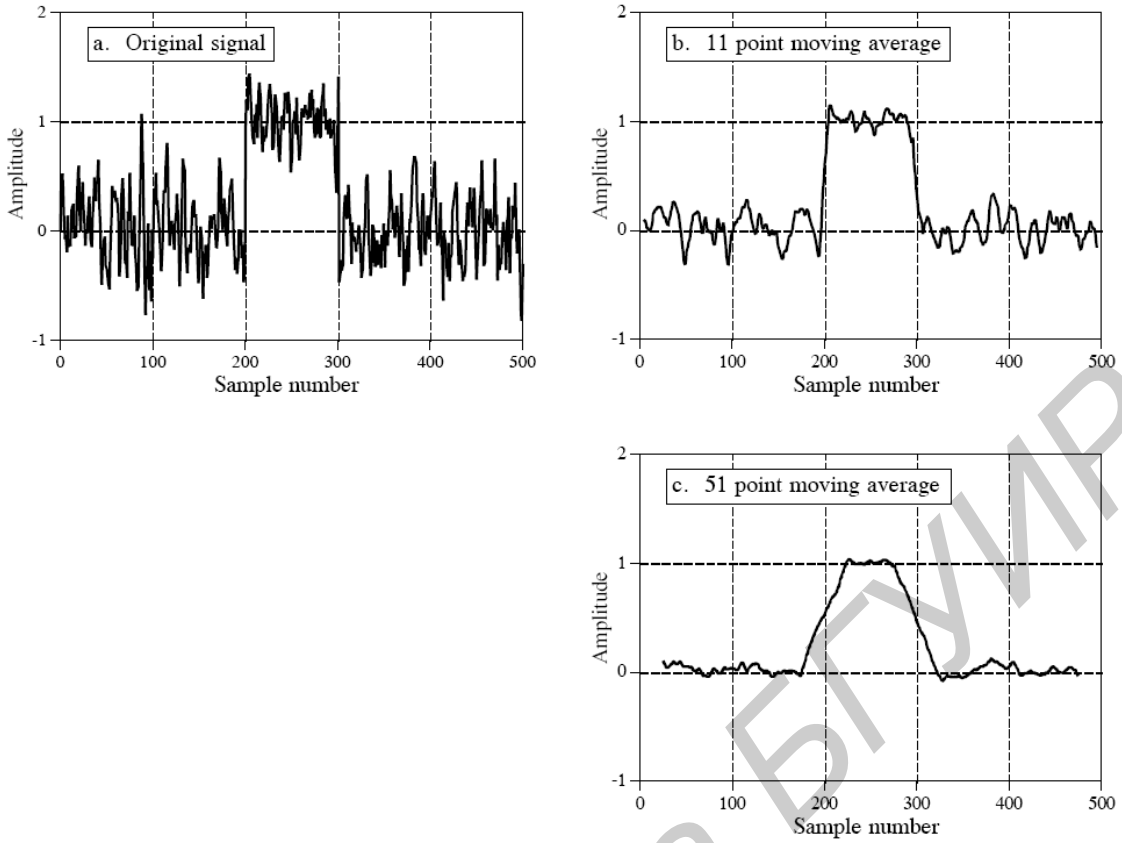


Рисунок 3.1 – Пример скользящего усредняющего фильтра

Чтобы понять, почему использование скользящего усредняющего фильтра является самым лучшим решением, представьте, что вы хотите создать фильтр с заданной резкостью краев. Например, допустим, что мы зафиксировали резкость краев, определив подъем ступенчатого отклика за 11 точек. Для этого необходимо, чтобы ядро фильтра имело 11 точек. Возникает вопрос оптимизации: как выбрать 11 величин в ядре фильтра, чтобы минимизировать шум в выходном сигнале? Поскольку шум, который мы пытаемся уменьшить, случайный, то во входном сигнале нет каких-то особенных точек, каждая создает точно такой же шум, как и соседняя. Следовательно, бесполезно отдавать предпочтение в обработке какой-либо одной точке из входного сигнала, назначая для нее более большой коэффициент в ядре фильтра. Самый низкий шум получается, когда все входные отсчеты обрабатываются одинаково, т. е. при использовании скользящего усредняющего фильтра.

Частотный отклик (частотная характеристика)

На рисунке 3.2 показан частотный отклик скользящего усредняющего фильтра. Математически он описывается преобразованием Фурье прямоугольного импульса:

$$H[f] = \frac{\sin(\pi f M)}{M \sin(\pi f)}. \qquad (3.2)$$

По формуле (3.2) рассчитывается частотный отклик M-точечного скользящего усредняющего фильтра. Частота f меняется от 0 до 0,5. Для $f = 0$ применяют $H[f] = 1$.

Такой фильтр затруднительно использовать как низкочастотный фильтр из-за медленного спада и плохого подавления в полосе блокировки. Очевидно, что скользящий усредняющий фильтр не может отделить одну полосу частот от другой. Известно, что «хорошие» характеристики во временной области преобразуются в «плохие» характеристики в частотной области и наоборот. Таким образом, скользящий усредняющий фильтр исключительно хорош как *сглаживающий фильтр* (действие во временной области), но исключительно плох как *низкочастотный фильтр* (действие в частотной области).

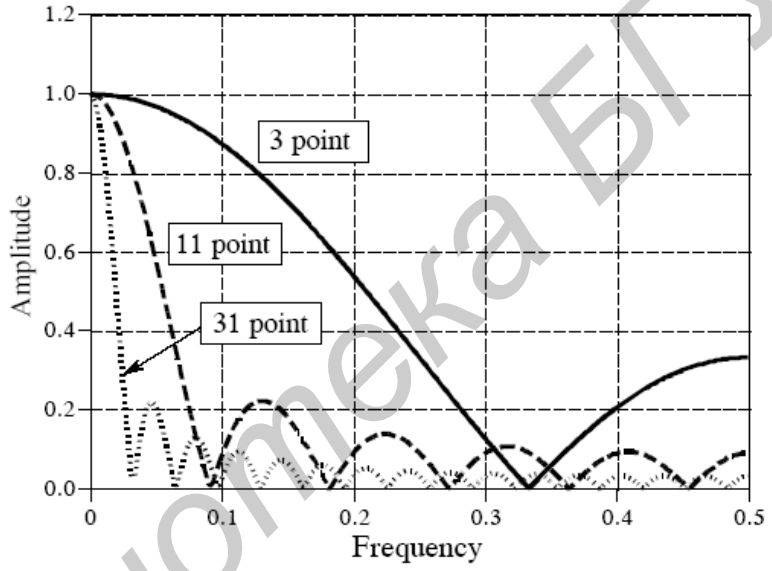


Рисунок 3.2 – Частотный отклик скользящего усредняющего фильтра

3.3.3 Оконный *sinc*-фильтр

Оконные *sinc*-фильтры (оконные гармонические фильтры) используются для отделения одной полосы частот от другой. Они очень стабильны, предсказуемы и позволяют получить очень высокий уровень характеристик. Их исключительно «хорошие» характеристики в частотной области получаются ценой «плохих» характеристик во временной области, включая звон и выбросы в шаговом отклике. Когда *sinc*-фильтры выполняются стандартной сверткой, они легко программируются, но медленно вычисляются.

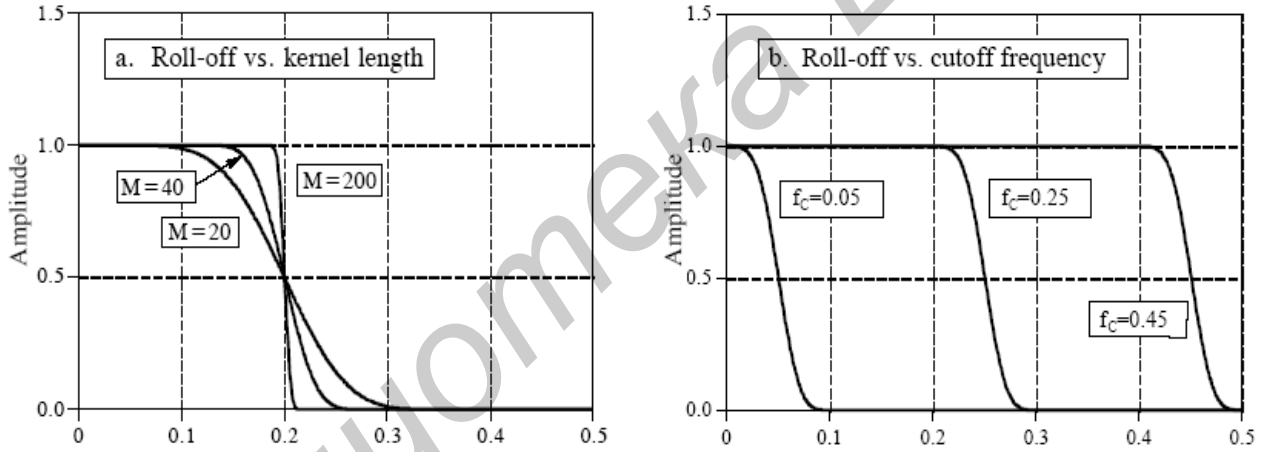
Для создания оконного *sinc*-фильтра необходимо выбрать два параметра: f_C – частоту среза (*cutoff frequency*) и M – длину ядра фильтра. Частота среза,

выраженная в долях частоты дискретизации, должна лежать в пределах от 0 до 0,5. Величина *M* определяется *крутизной спада* (*roll-off*) согласно следующему приближению:

$$M = \frac{4}{BW}. \tag{3.3}$$

Согласно формуле (3.3) длина ядра *M* определяется шириной полосы спада *BW*. Это только аппроксимация, поскольку крутизна спада зависит от конкретного окна.

Здесь *BW* – ширина спада, измеренная в точках, где кривая едва подходит к единице и к нулю (например, 99 % и 1 %). Ширина полосы перепада также выражается в долях частоты дискретизации и должна быть между 0 и 0,5. Рисунок 3.3, *a* показывает пример использования этой аппроксимации. Показаны три кривые для ядра фильтра с длиной *M*, равной 20, 40, 200. Из формулы (3.3) получаем ширину полосы перепада *BW*, равную 0,2, 0,1 и 0,02 соответственно. Из рисунка 3.3, *b* видно, что форма частотного отклика не зависит от выбранной частоты среза (*M* = 60).



a *b*

Рисунок 3.3 – Длина ядра фильтра (*a*) и крутизна спада оконного *sinc*-фильтра (*b*)

Поскольку время, требуемое для свертки, пропорционально длине сигнала, то формула (3.3) выражает компромисс между *временем вычисления* (зависит от величины *M*) и *резкостью фильтра* (величина *BW*). Например, меньшая на 20 % крутизна спада окна Блэкмана (по сравнению с окном Хэмминга) может быть скомпенсирована путем увеличения длины ядра на 20 %. Можно сказать, что время вычисления окна Блэкмана на 20 % больше, чем окна Хэмминга с эквивалентной крутизной спада. Это важно, так как скорость вычисления этих фильтров и так очень медленная.

Как видно из рисунка 3.3, *b*, частота среза оконного s*inc*-фильтра измеряется в точке равной *половине* амплитуды. Почему используется 0,5 вместо стандартной 0,707 (3 дБ), используемой в аналоговых и других цифровых фильтрах? Это вызвано тем, что частотный отклик оконного *sinc*-фильтра *симметричен* между полосой пропускания и полосой блокировки. Например, окно Хэмминга имеет рябь в полосе пропускания 0,2 % и *идентичную* в полосе подавления 0,2 %. Другие фильтры не имеют этой симметрии и поэтому не имеют преимущества в использовании точки в пол-амплитуды для обозначения частоты среза. Эта симметрия делает оконные *sinc*-фильтры идеальными для *спектральной инверсии*.

После выбора f_C и M ядро оконного *sinc*-фильтра вычисляется по формуле

$$h[i] = K\frac{\sin(2\pi f_C(i - M/2))}{i - M/2}\left[0{,}42 - 0{,}5\cos\left(\frac{2\pi i}{M}\right) + 0{,}08\cos\left(\frac{4\pi i}{M}\right)\right]. \quad (3.4)$$

Частота среза f_C выражена в долях частоты дискретизации от 0 до 0,5. Длина ядра фильтра определяется величиной M, которая должна быть четным целым числом. Номер отсчета i – целое число, меняющееся от 0 до M, что дает $M + 1$ точек в ядре. Константа K выбирается из условия получения единичного коэффициента усиления на нулевой частоте. Чтобы избежать деления на нуль, при $i = M/2$ берут $h[i] = 2\pi f_C K$.

3.3.4 Рекурсивные фильтры

При работе рекурсивного фильтра каждая точка выходного сигнала находится путем сложения результатов перемножения величин входного сигнала на коэффициент «*a*» и перемножения ранее вычисленных величин выходного сигнала на коэффициент «*b*»:

$$\begin{aligned} y[n] = a_0x[n] + a_1x[n-1] + a_2x[n-2] + a_3x[n-3] + \ldots \\ + b_1y[n-1] + b_2y[n-2] + b_3y[n-3] + \ \ldots\ . \end{aligned} \quad (3.5)$$

Формула (3.5) называется рекурсивной формулой, а фильтр, который использует ее, называется рекурсивным фильтром.

В формуле нет величины b_0, так как она соответствует вычисляемому отсчету. Величины «*a*» и «*b*», которые определяют фильтр, называются рекурсивными коэффициентами. На практике не используется более двенадцати рекурсивных коэффициентов, иначе фильтр становится неустойчивым.

3.3.5 Моделирование цифровых фильтров средствами программ *GUI MATLAB*: *GUI FDATool*

На основе программных средств в *MATLAB* разработаны программы *GUI* (*Graphic User Interface* – графический интерфейс пользователя), представляющие собой средства, предназначенные для моделирования путем интерактивного общения без прямого доступа к программным средствам с графическим выво-

дом результатов. В частности, для моделирования цифровых фильтров (ЦФ) разработаны две программы *GUI – FDATool* и *SPTool*.

Программа *GUI FDATool* (*Filter Design and Analysis Toolbox* – средства проектирования и анализа фильтров) разработана на основе пакетов расширения *Signal Processing Toolbox* и *Filter Design Toolbox* и предназначена для проектирования цифровых фильтров.

Основные задачи, связанные с проектированием ЦФ и решаемые средствами *GUI FDATool*:

– синтез ЦФ;
– выбор структуры ЦФ;
– анализ ЦФ;
– сохранение ЦФ на время сеанса в *GUI FDATool*;
– экспорт ЦФ как объектов *dfilf*;
– импорт ЦФ как объектов *dfilf*;
– моделирование структуры ЦФ с фиксированной точкой (ФТ).

Обращение к *GUI FDATool* происходит по команде ***fdatool***, после чего открывается окно ***Filter Design & Analysis Tool*** (рисунок 3.4) с привычным интерфейсом современных *Windows*-приложений.

Синтез цифровых фильтров

Синтез ЦФ производится при открытом окне ***Filter Design & Analysis Tool*** и нажатой кнопке ***Design Filter*** («Проектирование фильтра»), расположенной на панели инструментов в нижнем левом углу (см. рисунок 3.4).

Основные этапы синтеза ЦФ в *GUI FDATool* включают в себя:

I Выбор типа ЦФ. Тип фильтра выбирается в группе *Design Method* («Метод синтеза») с помощью переключателя ***IIR*** (БИХ-фильтр) или ***FIR*** (КИХ-фильтр).

II Выбор метода синтеза ЦФ. Метод синтеза выбирается в группе ***Design Method*** в раскрывающихся списках ***FIR*** или ***IIR***.

III Задание входных параметров. Входные параметры задаются в группе ***Options*** («Параметры»), и их набор зависит от типа ЦФ (КИХ или БИХ), метода синтеза и переключателя в группе ***Filter Order*** («Порядок фильтра»). Входные параметры обсуждаются далее.

IV Задание требований к АЧХ. Требования к АЧХ включают в себя:

1 Тип избирательности, задаваемый в группе ***Response Type*** («Тип характеристики») с помощью переключателей ***Lowpass*** (ФНЧ), ***Highpass*** (ФВЧ), ***Bandpass*** (ПФ) или ***Bandstop*** (РФ).

2 Частоту дискретизации и граничные частоты полос пропускания (ПП) и полос задерживания (ПЗ), задаваемые в группе ***Frequency Specifications*** («Требования к частотам»). Предварительно в раскрывающемся списке ***Units*** («Единицы измерения») указываются единицы измерения частот, после чего задаются частоты:

– в поле ввода ***Fs*** – частота дискретизации;

– в полях ввода ***Fpass***, ***Fstop*** – граничные частоты ПП и ПЗ.

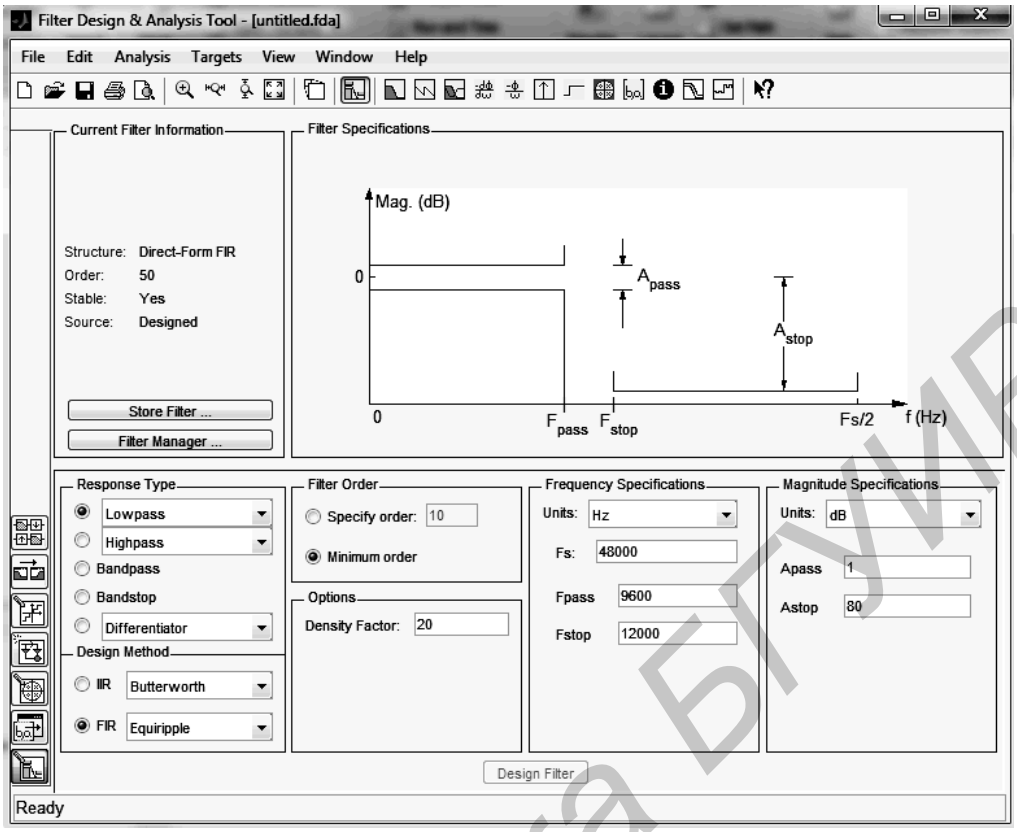


Рисунок 3.4 – Окно Filter Design & Analysis Tool

3 Допустимые отклонения АЧХ в ПП и ПЗ, задаваемые в группе ***Magnitude Specifications*** («Требования к АЧХ»). Предварительно в раскрывающемся списке *Units* указываются единицы измерения АЧХ:

– ***Linear*** («Безразмерный») – если требования задаются к нормированной АЧХ;

– ***dB*** (дБ) – если требования задаются к АЧХ (дБ) (характеристике затухания [6]).

После этого задаются допустимые отклонения АЧХ:

1 В полях ввода ***Dpass*** (в дБ – ***Apass***) – допустимые отклонения в ПП;

2 В полях ввода ***Dstop*** (в дБ – ***Astop***) – допустимые отклонения в ПЗ.

V Синтез ЦФ. Он производится при нажатии кнопки ***Design Filter***, расположенной внизу по центру окна (см. рисунок 3.4). В дальнейшем во избежание путаницы данной кнопки с кнопкой ***Design Filter***, расположенной на панели инструментов в нижнем левом углу, первую будем называть просто кнопкой ***Design Filter***, а вторую – кнопкой ***Design Filter*** на панели инструментов в нижнем левом углу. По завершении синтеза автоматически выдаются:

1 В группе ***Magnitude Response*** (***dB***) (АЧХ (дБ)) – график АЧХ (дБ) (характеристика ослабления [6]). После синтеза данная группа автоматически замещает группу ***Frequency Specifications***.

2 В группе ***Current Filter Information*** («Текущая информация о фильтре»):

– ***Structure*** – структура фильтра;
– ***Order*** – порядок фильтра;
– ***Stable*** – устойчивость фильтра (*Yes* или *No*);
– ***Source*** – способ загрузки фильтра – в результате синтеза в ***GUI FDATool*** (***Designed***) или импорта (***Imported***).

Входные параметры цифровых фильтров

Входные параметры ЦФ зависят от его типа (КИХ или БИХ) и метода синтеза. Особое внимание следует уделить входным параметрам КИХ-фильтров, так как итерационные методы синтеза данных фильтров связаны с проверкой выполнения требований к АЧХ.

Проверка выполнения требований к АЧХ КИХ-фильтров

В [6] рассмотрен синтез КИХ-фильтров итерационным методом наилучшей равномерной (чебышевской) аппроксимации.

В ***GUI FDATool*** минимальный порядок R [6] синтезированного КИХ-фильтра по умолчанию определяется с точностью до ±2. Точное определение минимального порядка $R_{\min}$ КИХ-фильтра предполагает проверку выполнения требований к АЧХ.

В зависимости от точности определения порядка выбирается переключатель в группе ***Filter Order***, а именно:

– если порядок R определяется с точностью до ±2, то устанавливается переключатель ***Minimum order*** («Минимальный порядок»);

– если определяется минимальный порядок $R_{\min}$, то сначала устанавливается переключатель ***Minimum order*** и выполняется синтез фильтра, а затем при уточнении порядка – переключатель ***Specify order*** («Произвольный порядок»).

В зависимости от выбора переключателя в группе ***Filter Order*** задаются разные входные параметры. Однако, прежде чем их обсуждать, остановимся на проверке выполнения требований к АЧХ, не зависящей от метода синтеза.

Проверка требований может выполняться:

1 К нормированной АЧХ. В этом случае предварительно выводится график нормированной АЧХ, для чего в контекстном меню (открывается щелчком правой кнопки мыши на свободном поле графика) следует выбрать команду ***Analysis Parameters*** («Параметры анализа»). Открывается одноименное окно, где в раскрывающемся списке ***Magnitude Display*** («Вывод АЧХ») нужно выбрать ***Magnitude*** (АЧХ) и нажать кнопку ***OK***. Аналогичные действия можно выполнить, выбирая в пункте меню ***Analysis*** команду ***Analysis Parameters***.

2 К АЧХ (дБ) (характеристика ослабления [6]).

Проверка выполнения требований к АЧХ производится следующим образом:

1 Поочередно выводится АЧХ в ПП и ПЗ. Для этого следует нажать кнопку ***Zoom in*** («Увеличить масштаб») на панели инструментов и, не отпуская левую кнопку мыши, поочередно выделить АЧХ в ПП и ПЗ.

2 В каждой из ПП и ПЗ определяется максимальное (по модулю) отклонение АЧХ и сравнивается с заданным. Для этого к соответствующей точке АЧХ нужно подвести указатель мыши и выполнить щелчок левой кнопкой, после чего выдаются координаты точки – значения частоты и АЧХ. Расставленные точки удаляются с помощью команды контекстного меню ***Delete*** или ***Delete all***. Возврат к исходному масштабу АЧХ выполняется нажатием кнопки ***Zoom to full view*** («Исходный масштаб») или щелчком правой кнопки мыши на свободном поле графика.

Возможны две ситуации:

1 Требования не выполняются. В этом случае следует увеличить порядок фильтра R и, повторяя процедуру синтеза, определить минимальный порядок $R_{\min}$, при котором требования будут выполняться.

2 Требования выполняются. В этом случае можно уменьшить порядок фильтра R и, повторяя процедуру синтеза, определить минимальный порядок $R_{\min}$, при котором требования будут выполняться.

В обоих случаях изменение порядка производится при выборе в группе ***Filter Order*** переключателя ***Specify order***.

Рассмотрим, как задаются входные параметры при выборе разных переключателей – ***Minimum order*** или ***Specify order***.

Входные параметры при синтезе КИХ-фильтров методом наилучшей равномерной (чебышевской) аппроксимации

При синтезе КИХ-фильтров методом чебышевской аппроксимации (в раскрывающемся списке ***FIR*** выбирается значение ***Equiripple***) входные параметры зависят от переключателя в группе ***Filter Order***, а именно:

При выборе переключателя ***Minimum order*** в группе ***Options*** в поле ввода ***Density Factor*** («Коэффициент плотности сетки частот») необходимо задать значение этого параметра. Параметр **Density Factor** тождественен параметру ***lgrid*** в функциях ***firpm*** и ***firgr*** [6].

При выборе переключателя ***Specify order*** в его поле ввода необходимо указать порядок КИХ-фильтра и в группе ***Magnitude Specifications*** задать:

– в поле ввода ***Wpass*** – вес (веса) в ПП;

– в поле ввода ***Wstop*** – вес (веса) в ПЗ.

Веса определяются по методике, изложенной в [6]: вес, равный единице, присваивается полосе с наибольшим максимально допустимым отклонением, а веса в остальных полосах рассчитываются как отношение наибольшего максимально допустимого отклонения к максимально допустимому отклонению в данной полосе. Поэтому веса – всегда числа, большие или равные единице.

В ***GUI FDATool*** для синтеза частотно-избирательных фильтров методом чебышевской аппроксимации по умолчанию используются КИХ-фильтры толь-

ко 1-го и 2-го типов, а КИХ-фильтры 3-го и 4-го типов – для цифровых преобразователей Гильберта и дифференциаторов (таблица 1 в [6]). Поэтому вид импульсной характеристики – симметричная/антисимметричная – не задается.

Входные параметры БИХ-фильтров при синтезе методом билинейного Z-преобразования

Метод билинейного Z-преобразования [7] автоматически обеспечивает минимальный порядок синтезированного БИХ-фильтра, поэтому проверка выполнения требований к АЧХ не производится.

Соответственно, в группе ***Filter Order*** активизирован только переключатель ***Minimum order***, и в группе ***Options*** в раскрывающемся списке ***Match Exactly*** («Согласование точно») следует задать значение одноименного параметра.

Значения в списке ***Match Exactly*** тождественны значениям параметра MATCH в функциях синтеза БИХ-фильтров [7].

Примеры синтеза цифровых фильтров

Для лучшего понимания синтеза ЦФ средствами ***GUI FDATool*** рекомендуется предварительно познакомиться с синтезом ЦФ программными средствами *MATLAB* [6, 7].

Пример 1

В примере 2 [6] был синтезирован программными средствами MATLAB оптимальный КИХ-фильтр ФНЧ по заданным требованиям к АЧХ (таблица 3 в [6]) методом чебышевской аппроксимации. Необходимо синтезировать тот же фильтр средствами GUI FDATool.

Основные этапы синтеза, описанные ранее, включают в себя:

I Выбор типа ЦФ. В группе ***Design Method*** установить переключатель ***FIR***.

II Выбор метода синтеза. В группе ***Design Method*** в раскрывающемся списке ***FIR – Equiripple*** («Метод наилучшей равномерной (чебышевской) аппроксимации»).

III Задание входных параметров. В группе ***Filter Order*** – переключатель ***Minimum order***. В группе ***Options*** в поле ввода ***Density Factor*** – 20.

IV Задание требований к АЧХ (см. таблицу 3 в [6]):

1 Тип избирательности задается в группе ***Response Type*** с помощью переключателя ***Lowpass***.

2 Частоты дискретизации и граничные частоты ПП и ПЗ задаются в группе ***Frequency Specifications***:

– в раскрывающемся списке ***Units*** – ***Hz***;
– в поле ввода ***Fs*** – 8000;
– в поле ввода ***Fpass*** – 1000;
– в поле ввода ***Fstop*** – 1500.

3 Максимально допустимые отклонения АЧХ указываются в группе ***Magnitude Specifications***:

– в раскрывающемся списке ***Units*** – ***Linear***;
– в поле ввода ***Dpass*** – 0,05;
– в поле ввода ***Dstop*** – 0,01.

V Синтез фильтра производится после нажатия кнопки ***Design Filter***, и по его завершении автоматически выдаются:

1 В группе ***Magnitude Response (dB)*** – график АЧХ (дБ) (характеристика ослабления [6]).

2 В группе ***Current Filter Information***:
– ***Structure*** – *Direct-Form FIR*;
– ***Order*** – 23;
– ***Stable*** – *Yes*;
– ***Source*** – *Designed*.

Определим, является ли КИХ-фильтр оптимальным. Для этого проверим, выполняются ли требования к АЧХ по изложенной методике, и увидим, что они не выполняются ни в ПП, ни в ПЗ.

Для изменения порядка КИХ-фильтра в группе ***Filter Order*** выберем переключатель ***Specify order*** и в его поле ввода укажем порядок КИХ-фильтра – 24.

Зададим новые входные параметры в группе ***Magnitude Specifications***:
– в поле ввода ***Wpass*** – 1;
– в поле ввода ***Wstop*** – 5.

Вес, равный единице, присвоен ПП с наибольшим максимально допустимым отклонением 0,05, а вес в ПЗ рассчитан как отношение 0,05/0,01 = 5.

Нажмем кнопку ***Design Filter*** (она активизируется при нажатии кнопки ***Design Filter*** на панели инструментов в нижнем левом углу). Проверим выполнение требований к АЧХ и убедимся, что они вновь не выполняются.

Повторим процедуру, задавая порядок $R = 25$, не меняя входные параметры, и убедимся, что требования к АЧХ выполняются.

Таким образом, синтезирован оптимальный ФНЧ с ЛФЧХ порядка $R_{opt} = 25$ на базе КИХ-фильтра 2-го типа.

Пример 2

В примере 2 [7] программными средствами *MATLAB* был синтезирован оптимальный БИХ-фильтр ФНЧ Золотарева – Кауэра (эллиптический) по заданным требованиям к АЧХ (см. таблицу 3 в [6]) методом билинейного *Z*-преобразования. Необходимо синтезировать тот же фильтр средствами ***GUI FDATool***.

Основные этапы синтеза, описанные ранее, включают в себя:

I Выбор типа ЦФ. В группе ***Design Method*** – переключатель ***IIR***.

II Выбор метода синтеза. В группе ***Design Method*** в раскрывающемся списке ***IIR*** – ***Elleptic*** («Эллиптический фильтр: метод билинейного *Z*-преобразования»).

III Задание входных параметров. В группе ***Filter Order*** – переключатель ***Minimum order***. В группе ***Options*** в раскрывающемся списке ***Match Exactly*** – ***both***.

IV Задание требований к АЧХ (см. таблицу 3 в [6]):

1 Тип избирательности указывается в группе ***Response Type*** – переключатель ***Lowpass***.

2 Частота дискретизации и граничные частоты ПП и ПЗ задаются в группе ***Frequency Specifications***:

– в раскрывающемся списке ***Units*** – ***Hz***;
– в поле ввода ***Fs*** – 8000;
– в поле ввода ***Fpass*** – 1000;
– в поле ввода ***Fstop*** – 1500.

3 Допустимые отклонения АЧХ (дБ) указываются в группе ***Magnitude Specifications***:

– в раскрывающемся списке ***Units*** – ***dB***;
– в поле ввода ***Apass*** – 0.4455;
– в поле ввода ***Astop*** – 40.

Значения ***Apass*** (дБ) и ***Astop*** (дБ) вычислены в примере 1 [6].

Для БИХ-фильтров не предусмотрено задание требований к нормированной АЧХ.

V Синтез фильтра производится после нажатия кнопки ***Design Filter***, и по его завершении автоматически выдаются:

1 В группе ***Magnitude Response (dB)*** – график АЧХ (дБ) (характеристика ослабления [6]).

2 В группе ***Current Filter Information***:

– ***Structure*** – *Direct-Form* II, *Second-Order Sections*;
– ***Order*** – 5;
– ***Sections*** – 3;
– ***Stable*** – *Yes*;
– ***Source*** – *Designed*.

Таким образом, синтезирован оптимальный БИХ-фильтр ФНЧ Золотарева – Кауэра порядка $R_{\min} = 5$.

Выбор структуры цифрового фильтра

По завершении синтеза ЦФ в ***GUI FDATool*** автоматически создается его описание в виде объекта ***dfilt*** со структурой, выбираемой по умолчанию: она указывается в группе ***Current Filter Information***.

Для преобразования структуры в пункте меню ***Edit*** («Редактирование») следует выбрать команду ***Convert Structure*** («Преобразование структуры»). Откроется одноименное окно со списком рекомендуемых для данного ЦФ структур. Из списка необходимо выбрать требуемую структуру и нажать кнопку ***OK***. Новая структура фиксируется в группе ***Current Filter Information***.

Для БИХ-фильтров с каскадными структурами из звеньев 2-го порядка предусмотрена возможность:

– расстановки звеньев – с целью минимизации собственных шумов;
– масштабирования – с целью минимизации вероятности переполнения на выходах сумматоров при реализации ЦФ с ФТ.

Для расстановки звеньев следует в пункте меню ***Edit*** выбрать команду ***Reorder and Scale Second-Order Sections*** («Расстановка и масштабирование звеньев 2-го порядка»). Откроется окно ***Reordering and Scaling of Second-Order Sections*** (рисунок 3.5), в котором:

1 Расстановка звеньев фиксируется с помощью переключателей в группе ***Reordering*** («Расстановка»). Переключатели отображают параметры функции ***reorder*** (см. таблицу 2 в [8]). По умолчанию для минимизации собственных шумов звенья расставлены в порядке возрастания радиусов (добротностей) полюсов, чему соответствует переключатель ***Auto*** («По умолчанию»).

2 Норма, на основе которой выполняется масштабирование, выбирается в группе ***Scaling*** («Масштабирование») при установленном флаге ***Scale*** («Масштаб»). Эта операция эквивалентна выполняемой с помощью функции ***scale*** (см. таблицу 2 в [8]). Одна из шести норм выбирается с помощью бегунка. Кроме этого, в группу ***Scaling*** включены поля ввода и раскрывающиеся списки, используемые при моделировании ЦФ с ФТ [9, 10].

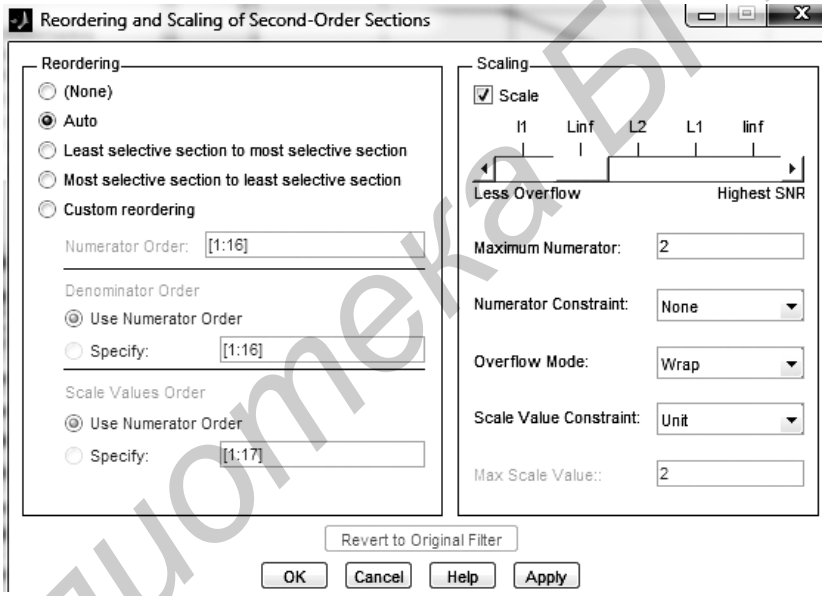


Рисунок 3.5 – Окно *Reordering and Scaling of Second-Order Sections*

После установки в группе ***Reordering*** переключателя ***Auto*** и указания в группе ***Scaling*** нормы для масштабирования нажимается кнопка ***OK***.

Анализ цифровых фильтров

Анализ ЦФ, синтезированного в ***GUI FDATool*** или импортированного из ***Workspace***, производится при открытом окне ***Filter Design & Analysis Tool*** и нажатой кнопке ***Design Filter*** на панели инструментов в нижнем левом углу с помощью команд пункта меню ***Analysis*** (или дублирующих их кнопок на панели инструментов):

– ***Magnitude Response*** (АЧХ);
– ***Phase Response*** (ФЧХ);
– ***Magnitude and Phase Responses*** (АЧХ и ФЧХ);

– ***Group Delay Response*** («ГВЗ – групповое время задержки»);
– ***Phase Delay*** («Фазовая задержка»);
– ***Impulse Response*** («Импульсная характеристика»);
– ***Step Response*** («Переходная характеристика»);
– ***Pole/Zero Plot*** («Карта нулей и полюсов»);
– ***Filter Coefficients*** («Коэффициенты передаточной функции»);
– ***Filter Information*** («Свойства фильтра – объекта ***dfilt***»);
– ***Magnitude Response Estimate*** («Оценка АЧХ – для ЦФ с ФТ»);
– ***Round-off Noise Power Spectrum*** («Энергетический спектр шума округления – для ЦФ с ФТ»).

Помимо этого, в пункт меню ***Analysis*** включены дополнительные команды:

1 ***Filter Specifications*** («Требования к АЧХ») – по этой команде в поле графика выводятся требования к АЧХ.

2 ***Overlay Analysis*** («Наложение характеристик») – по этой команде выводится список флагов, имена которых дублируют вышеперечисленные команды анализа. Установив соответствующий флаг, можно добавить в поле графика еще одну характеристику, поверх имеющейся. По умолчанию установлен флаг ***None*** («Добавляемой характеристики нет»).

3 ***Analysis Parameters*** («Параметры анализа») – по этой команде открывается окно с тем же именем, в котором можно изменить параметры анализируемой характеристики (список параметров зависит от характеристики).

4 ***Sampling Frequency*** («Частота дискретизации») – по этой команде открывается одноименное окно, в котором можно изменить единицы измерения частоты дискретизации.

Сохранение цифровых фильтров на время сеанса в GUI FDATool

При работе в *GUI FDATool* в течение одного сеанса может синтезироваться (либо импортироваться) несколько ЦФ. Для их сохранения на время сеанса удобно воспользоваться кнопкой ***Store Filter*** («Сохранение фильтра») в группе ***Current Filter Information*** (эта кнопка дублирует команду ***Store Filter*** в пункте меню ***File***). При этом открывается окно ***Store Filter***, в котором указывается имя фильтра, и нажимается кнопка ***OK.***

Сохраняемые на время сеанса ЦФ размещаются в буфере ***Filter Manager*** («Диспетчер фильтров»). Для загрузки ЦФ из буфера в *GUI FDATool* следует нажать кнопку ***Filter Manager*** в группе ***Current Filter Information***. Открывается окно ***Filter Manager***, в котором достаточно курсором указать имя ЦФ.

Для того чтобы редактирование сохраненного ранее ЦФ автоматически сохранялось, следует в окне ***Filter Manager*** установить флаг ***Edit.***

Для анализа характеристик сохраненного ЦФ с помощью *GUI FVTool* следует в окне ***Filter Manager*** нажать кнопку ***FVTool,*** для переименования ЦФ – кнопку ***Rename*** («Переименование»), а для удаления ЦФ – кнопку ***Remove*** («Удаление»).

Экспорт цифровых фильтров как объектов dfilt

При выходе из *GUI FDATool* после окончания сеанса все ЦФ, сохраненные в буфере ***Filter Manager***, автоматически удаляются.

Сохранение синтезированного ЦФ осуществляется путем его экспорта из *GUI FDATool*.

Можно экспортировать коэффициенты передаточной функции синтезированного ЦФ, но удобнее экспортировать ЦФ в виде объекта ***dfilt*** [8].

Для экспорта ЦФ (объекта ***dfilt***) на диск необходимо выполнить следующие действия:

1 Загрузить экспортируемый ЦФ из буфера ***Filter Manager*** (или экспортировать ЦФ непосредственно после его синтеза).

2 В пункте меню ***File*** выбрать команду ***Export*** («Экспорт»), после чего откроется одноименное окно (рисунок 3.6).

3 Сбросить флаг ***Overwrite Variables*** («Замена переменных») во избежание замены переменных, имеющихся в ***Workspace***, переменными объекта ***dfilt*** с теми же именами.

4 В раскрывающемся списке ***Export To*** («Экспортировать в…») выбрать *MAT-File*.

5 В раскрывающемся списке ***Export As*** («Экспортировать как…») выбрать ***Objects*** («Объекты»).

6 В группе ***Variable Names*** («Имена переменных») в поле ввода ***Discrete Filter*** («Дискретный фильтр») указать имя ЦФ (имя объекта ***dfilt***).

7 Нажать кнопку ***OK***, после чего откроется окно ***Export to a MAT-file*** с содержимым папки ***work*** (рисунок 3.7).

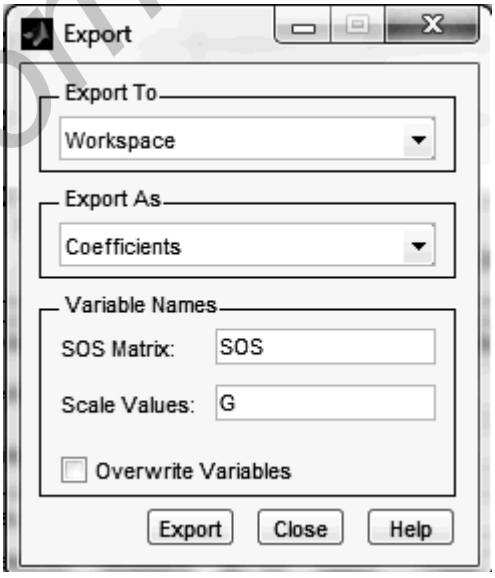


Рисунок 3.6 – Окно *Export*

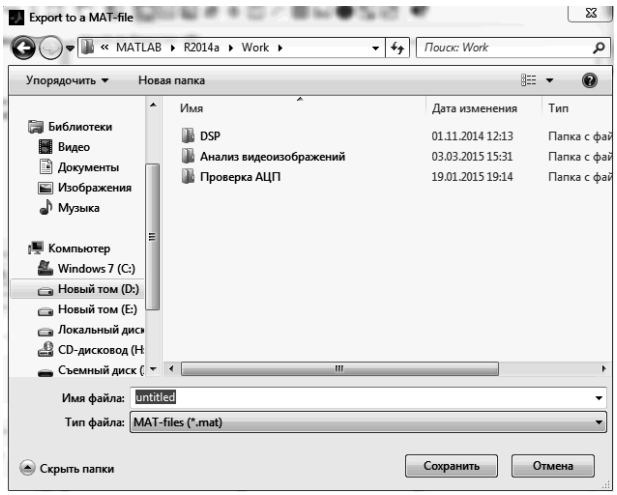


Рисунок 3.7 – Окно *Export to a MAT-file*

Если ЦФ (объект ***dfilt***) экспортируется в папку ***work***, то в поле ввода **Имя файла** дублируется имя ЦФ (объекта ***dfilt***) и нажимается кнопка **Сохранить**.

Если ЦФ экспортируется в ранее созданную папку пользователя, то предварительно следует ее открыть.

Если ЦФ экспортируется в новую папку пользователя, то предварительно следует создать ее и открыть. Для дальнейшего автоматического доступа к новой папке необходимо сохранить путь к ней.

Для экспорта ЦФ (объекта ***dfilt***) в ***Workspace*** следует:

1 Загрузить экспортируемый ЦФ из буфера ***Filter Manager*** (или экспортировать ЦФ непосредственно после его синтеза).

2 В пункте меню ***File*** выбрать команду ***Export***, после чего откроется одноименное окно.

3 Сбросить флаг ***Overwrite Variables*** во избежание замены переменных, имеющихся в ***Workspace***, переменными объекта ***dfilt*** с теми же именами.

4 В раскрывающемся списке ***Export To*** выбрать ***Workspace***.

5 В раскрывающемся списке ***Export As*** выбрать ***Objects***.

6 В группе ***Variable Names*** в поле ввода ***Discrete Filter*** указать имя ЦФ (имя объекта ***dfilt***).

7 Нажать кнопку ***OK***.

8 Проверить содержимое ***Workspace***.

Для экспорта ЦФ (объекта ***dfilt***) в *GUI SPTool* следует:

1 Загрузить экспортируемый ЦФ из буфера ***Filter Manager*** (или экспортировать ЦФ непосредственно после его синтеза).

2 Обратиться к *GUI SPTool* по команде: ***sptool***, после чего откроется окно ***SPTool: startup.spt***.

3 В *GUI FDATool* в пункте меню ***File*** выбрать команду **Export**, после чего откроется одноименное окно (см. рисунок 3.6).

4 В раскрывающемся списке ***Export To*** выбрать ***SPTool***.

5 В группе ***SPTool Options*** в поле ввода ***Discrete Filter*** указать имя ЦФ (имя объекта ***dfilt***).

6 Нажать кнопку ***OK***.

7 Убедиться, что имя экспортируемого ЦФ появилось в окне ***SPTool: startup.spt*** в списке ***Filters***.

Импорт цифровых фильтров как объектов dfilt

Синтезированные программными средствами *MATLAB* или средствами *GUI FDATool* и сохраненные на диске (в папке ***work*** или в папке пользователя) в виде объектов ***dfilt*** КИХ- и БИХ-фильтры могут быть импортированы в *GUI FDATool*. Для импорта объекта ***dfilt*** в *GUI FDATool* с диска необходимо выполнить следующие действия:

1 Загрузить импортируемый ЦФ (объект ***dfilt***) с диска в ***Workspace*** по команде ***load <имя объекта dfilt>***.

2 В пункте меню ***File*** выбрать команду ***Import Filter from Workspace*** («Импорт из рабочей области»), после чего в окне ***Filter Design & Analysis Tool*** появится группа ***Filter Coefficients*** («Коэффициенты фильтра»).

3 В группе ***Filter Coefficients*** в раскрывающемся списке ***Filter Structure*** («Структура фильтра») выбрать ***Filter object*** (фильтр как объект ***dfilt***).

4 В поле ввода ***Discrete filter*** («Дискретный фильтр») указать имя объекта ***dfilt***.

5 В раскрывающемся списке ***Units*** выбрать ***Noralized*** **(0 *to* 1)** (по умолчанию). Выбор других единиц измерения частот в раскрывающемся списке ***Units*** целесообразно производить после импорта ЦФ, в противном случае последует запрос о частоте дискретизации ***Fs*** из ***Workspace***.

6 Нажать кнопку ***Import Filter*** («Импорт фильтра»).

При импорте нескольких объектов ***dfilt*** в течение одного сеанса следует на время сессии сохранять их в буфере ***Filter Manager***.

После импорта объектов ***dfilt*** (одного или нескольких) для редактирования и/или анализа импортируемого ЦФ следует нажать кнопку ***Design Filter*** на панели инструментов в нижнем левом углу.

Моделирование структуры цифровых фильтров с фиксированной точкой

Моделирование ЦФ с ФТ средствами *GUI FDATool* предполагает обязательное знакомство с моделированием ЦФ с ФТ программными средствами *MATLAB* [9].

Согласно принятой в [9] терминологии, объект ***dfilt*** со свойством ***Arithmetic***: ***'double'*** назван исходным ЦФ (исходным КИХ- и БИХ-фильтром), а объект ***dfilt*** со свойством ***Arithmetic: 'fixed'*** – ЦФ с ФТ (КИХ- и БИХ-фильтром с ФТ).

Моделирование ЦФ с ФТ производится непосредственно после синтеза исходного ЦФ в *GUI FDATool* или его импорта из ***Workspace*** следующим образом:

1 В окне ***Filter Design & Analysis Tool*** нажать кнопку ***Set quatization parameters*** («Установка параметров квантования») на панели инструментов в нижнем левом углу. При этом нижняя половина окна ***Filter Design & Analysis***

Tool автоматически изменит свой вид: она будет содержать только раскрывающийся список ***Filter Arifmetic*** («Тип арифметики»).

2 В раскрывающемся списке ***Filter Arifmetic*** выбрать значение ***Fixed-point*** («Фиксированная точка»). Тем самым переходим от исходного ЦФ к ЦФ с ФТ, свойства которого установлены по умолчанию. Список свойств ЦФ с ФТ и их расшифровка приводятся в [5]. При выборе значения ***Fixed-point*** нижняя половина окна ***Filter Design & Analysis Tool*** вновь автоматически изменится: в правой части появятся три вкладки, контекстно связанные со свойствами ЦФ с ФТ (рисунок 3.8):

а) ***Coefficients*** («Коэффициенты») – для выбора форматов коэффициентов передаточной функции;

б) ***Input/Output*** («Вход/Выход») – для выбора форматов входного и выходного сигналов;

в) ***Filter Internals*** («Внутреннее состояние фильтра») – для выбора форматов при выполнении арифметических операций.

Для понимания содержимого вкладок рекомендуется обратиться к расшифровке соответствующих свойств ЦФ с ФТ [5].

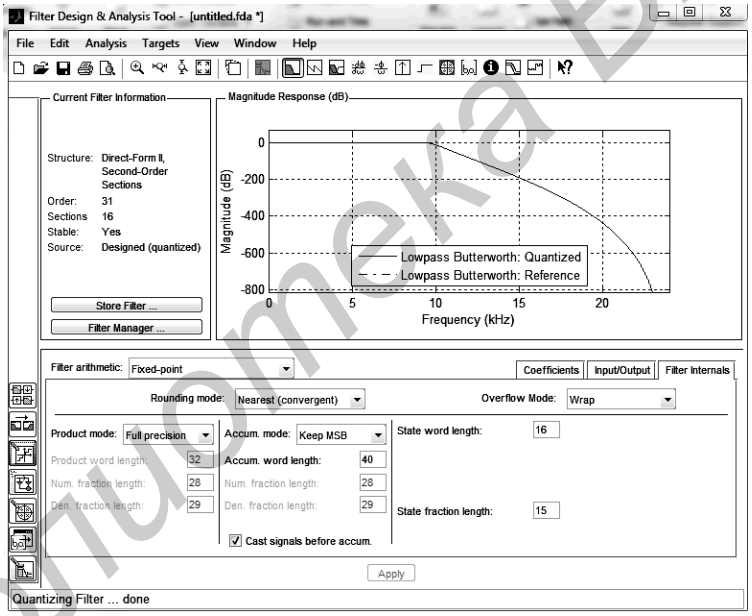


Рисунок 3.8 – Окно *Filter Design & Analysis Tool* после нажатия кнопки *Set quatizion parameters* и выбора в раскрывающемся списке *Filter Arifmetic* значения *Fixed-point*

3 Устанавливаются требуемые форматы ЦФ с ФТ в группах ***Coefficients***, ***Input/Output*** и ***Filter Internals***. Свойства ЦФ с ФТ, относящиеся к группам ***Input/Output*** и ***Filter Internals***, представляют интерес при моделировании фильтрации (расчете реакции ЦФ) программными средствами *MATLAB* или в *GUI SPTool*.

4 Для каскадных структур БИХ-фильтров с ФТ при необходимости выполняется масштабирование коэффициентов числителей в передаточных функциях звеньев и/или коэффициентов усиления звеньев. Для этого в пункте

меню ***Edit*** выбирается команда ***Reorder and Scale Second-Order Sections***, после чего откроется одноименное окно. При установленном флаге ***Scale*** в раскрывающемся списке ***Numerator Constant*** («Константа для коэффициентов числителя») выбирается способ масштабирования для коэффициентов числителей в передаточных функциях звеньев; значение *normalize* эквивалентно функции ***normalize*** (таблица 2 в [9]). В раскрывающемся списке ***Scale Value Constant*** («Константа для коэффициентов усиления») выбирается способ масштабирования для коэффициентов усиления звеньев.

5 Анализируются характеристики КИХ-фильтра с ФТ. Команды ***Magnitude Response Estimate*** и ***Round-off Noise Power Spectrum*** в пункте меню ***Analysis***, предназначенные для анализа ЦФ с ФТ, рассматривались ранее.

После моделирования ЦФ с ФТ он может экспортироваться как объект ***dfilt***.

3.4 Практическая часть

3.4.1 Разработка функции для скользящего усредняющего фильтра

На основе приведенного примера разработать собственную функцию, входным параметром которой будет сигнал и количество точек усреднения, выходным параметром будет отфильтрованный сигнала. Код функции приводится в отчете по лабораторной работе.

3.4.2 Изучение влияния скользящего усредняющего фильтра на различные виды шума

1 Построить и отфильтровать кардиосигнал с «белым» шумом. Оценить эффективность работы фильтра.

2 Построить и отфильтровать кардиосигнал с гармоническим шумом. Оценить эффективность работы фильтра.

3.4.3 Разработка функции расчета ядра НЧ *sinc*-фильтра

На основе приведенных данных разработать собственную функцию расчета ядра НЧ *sinc*-фильтра, входным параметром которой будет частота среза и ширина полосы спада, выходным параметром – ядро *sinc*-фильтра.

3.4.4 Изучение влияния *sinc*-фильтра на различные виды шума

1 Построить и отфильтровать кардиосигнал с «белым» шумом. Оценить эффективность работы фильтра.

2 Построить и отфильтровать кардиосигнал с гармоническим шумом ниже частоты среза. Оценить эффективность работы фильтра.

3 Построить и отфильтровать кардиосигнал с гармоническим шумом выше частоты среза. Оценить эффективность работы фильтра.

3.4.5 Создание набора рекурсивных коэффициентов с заданными параметрами в *fdatool* и разработка функции для фильтрации с помощью БИХ-фильтра

С помощью пакета ***fdatool*** создать набор рекурсивных коэффициентов фильтра с заданными характеристиками, импортировать в рабочее пространство *MATLAB* и разработать функцию для реализации цифрового рекурсивного фильтра.

3.5 Содержание отчета

1 Цель работы.
2 Таблица по варианту.
3 Листинги разработанных программ.
4 Результаты исследования указанных цифровых фильтров.
5 Выводы.

3.6 Контрольные вопросы и задания

1 Какова классификация цифровых фильтров?
2 Каковы особенности КИХ-фильтров?
3 Нарисуйте импульсную и частотную характеристику скользящего усредняющего фильтра и поясните их особенности.
4 Нарисуйте импульсную и частотную характеристику оконного *sinc*-фильтра и поясните их особенности.
5 Каковы особенности БИХ-фильтров?

Литература

1 Ingle, V. Digital Signal Processing Using MATLAB / V. Ingle, J. Proakis // Second Edition. – Thomson, 2006.
2 Оппенгейм, А. Цифровая обработка сигналов / А. Оппенгейм, Р. Шафер. – М. : Техносфера, 2006.
3 Сергиенко, А. Б. Цифровая обработка сигналов / А. Б. Сергиенко. – 2-е изд. – СПб. : ПИТЕР, 2006.
4 Солонина, А. И. Основы цифровой обработки сигналов / А. И. Солонина [и др.]. – 2-е изд. – СПб. : БХВ-Петербург, 2005.
5 Солонина, А. И. Цифровая обработка сигналов. Моделирование в *MATLAB* / А. И. Солонина, С. М. Арбузов. – СПб. : БХВ-Петербург, 2008.
6 Солонина, А. И. Моделирование цифровой обработки сигналов в *MATLAB*. Ч. 1. Синтез оптимальных (по Чебышеву) КИХ-фильтров программными средствами *MATLAB* / А. И. Солонина // Компоненты и технологии. – 2008. – №11.

7 Солонина, А. И. Моделирование цифровой обработки сигналов в *MATLAB*. Ч. 2. Синтез оптимальных БИХ-фильтров программными средствами *MATLAB* / А. И. Солонина // Компоненты и технологии. – 2008. – №12.

8 Солонина, А. И. Моделирование цифровой обработки сигналов в *MATLAB*. Ч. 3. Описание структур КИХ- и БИХ-фильтров в *MATLAB* / А. И. Солонина // Компоненты и технологии. – 2009. – №1.

9 Солонина, А. И. Моделирование цифровой обработки сигналов в *MATLAB*. Ч. 4. Моделирование структур цифровых фильтров с фиксированной точкой программными средствами *MATLAB*: анализ характеристик КИХ-фильтров / А. И. Солонина // Компоненты и технологии. – 2009. – №2.

Лабораторная работа №4

Обработка биомедицинского изображения средствами *MATLAB*

4.1 Цель работы

Изучение принципов построения томографических изображений.

4.2 Задачи

1 Изучить общие вопросы построения цифровых изображений.

2 Создать фантом головы.

3 Выполнить вычисление синтезированных проекций с помощью параллельных лучей.

4 Выполнить вычисление синтезированных проекций с помощью параллельных лучей.

5 Выполнить реконструкцию фантома головы на основании проекционных данных, полученных с помощью веерных лучей.

6 Выполнить вычисление синтезированных проекций с помощью веерных лучей.

4.3 Общая теория

4.3.1 Основы компьютерной томографии

Основная проблема рентгеновского изображения (или другого проникающего излучения) заключается в том, что двумерное изображение получается из трехмерного объекта. Это означает, что различные структуры (детали) перекрываются в финальном изображении, хотя они совершенно разделены в объекте. Это главная проблема в медицинской диагностике, где многие анатомические структуры могут накладываться на те, которые пытается рассмотреть врач. В 1930-х гг. эта проблема была решена за счет согласованного движения источника излучения и детектора в процессе получения изображения. Из геометрии этого движения одна проекция внутри пациента оставалась в фокусе, а структуры вне этой проекции становились расплывчатыми. Это аналогично камере, сфокусированной на объекте в 5 метрах, когда объекты на 1 и 50 метрах получаются расплывчатыми. Эта родственная техника, базирующаяся на размывании границ объекта при движении, теперь по совокупности называется классической томографией. Слово томография означает «чертеж проекции».

Несмотря на хорошее развитие более чем за 50 лет, классическая томография очень редко используется. Это из-за того, что она имеет существенное ограничение: накладывающиеся объекты не удаляются из объекта, а только делаются расплывчатыми. В результате качество изображения получается слишком низким для практического применения. Длительный поиск решения привел

к системе, которая может создавать изображение двумерного среза трехмерного объекта без наложения других структур этого объекта.

Эта проблема была решена в начале 1970-х гг. посредством разработки метода компьютерной томографии (КТ), который позволяет наблюдать анатомические структуры внутри тела. Рисунок 4.1 показывает типичное изображение, полученное КТ. Компьютерная томография была введена под торговыми названиями компьютерная осевая томография (*Computed Axial Tomography*) и *CAT*-сканирование. Эти термины теперь не одобряются в медицинской области, хотя и часто встречаются в публикациях.

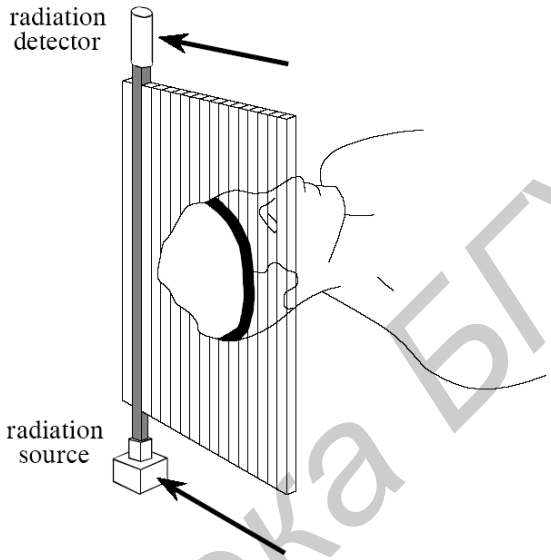


Рисунок 4.1 – Получение данных компьютерной томографией

Рисунок 4.1 иллюстрирует простую геометрию получения КТ-среза через центр головы. Узкий рентгеновский луч идет от источника к детектору. Это означает, что величина, измеренная детектором, связана с общей суммой материала, *через который* проходит луч. Кости и зубы больше ослабляют сигнал, чем мягкие ткани и жир. Как показано на этом рисунке, источник и детектор согласованно перемещаются для получения вида под этим специфическим углом. Эта фигура показывает только один получаемый вид, полное сканирование требует от 300 до 1000 видов, получаемых вращением с угловым шагом $0{,}3\ldots1{,}0^{o}$. Это выполняется установкой источника и детектора на поворотное устройство, вращающееся вокруг пациента. Преимуществом данных компьютерной томографии является то, что рентгеновское излучение проходит *только* через исследуемый срез тела. Это не похоже на классическую томографию, где рентгеновские лучи проходят через структуры, которые вы хотите отсечь в финальном изображении. Компьютерная томография исключает поступление ненужной информации в получаемые данные.

Перед реконструкцией изображения нужно произвести несколько предварительных шагов. Необходимо взять логарифм от каждого измерения.

Это вызвано экспоненциальным уменьшением интенсивности рентгеновского излучения при прохождении через материал. Взятие логарифма обеспечивает линейную зависимость сигнала от характеристик измеряемого материала. Другой предварительный шаг используется для компенсации *полихроматичности* (более одной энергии) рентгеновского излучения и *многоэлементности* детекторов (в отличие от одного элемента, показанного на рисунке 4.1). Хотя имеются и другие ключевые шаги в этой технике, но они не относятся к алгоритмам реконструкции, и мы не будем обсуждать их.

Рисунок 4.2 иллюстрирует соотношение между измеренным видом и соответствующим изображением. Таким образом, каждый отчет, получаемый в системе, равен сумме величин изображений вдоль луча этого отсчета. Например, вид 1 находится сложением всех пикселов ряда. Аналогично вид 3 находится сложением всех пикселов в каждой колонке. Другой вид (например вид 2) есть сумма пикселов вдоль строки, идущей под углом.

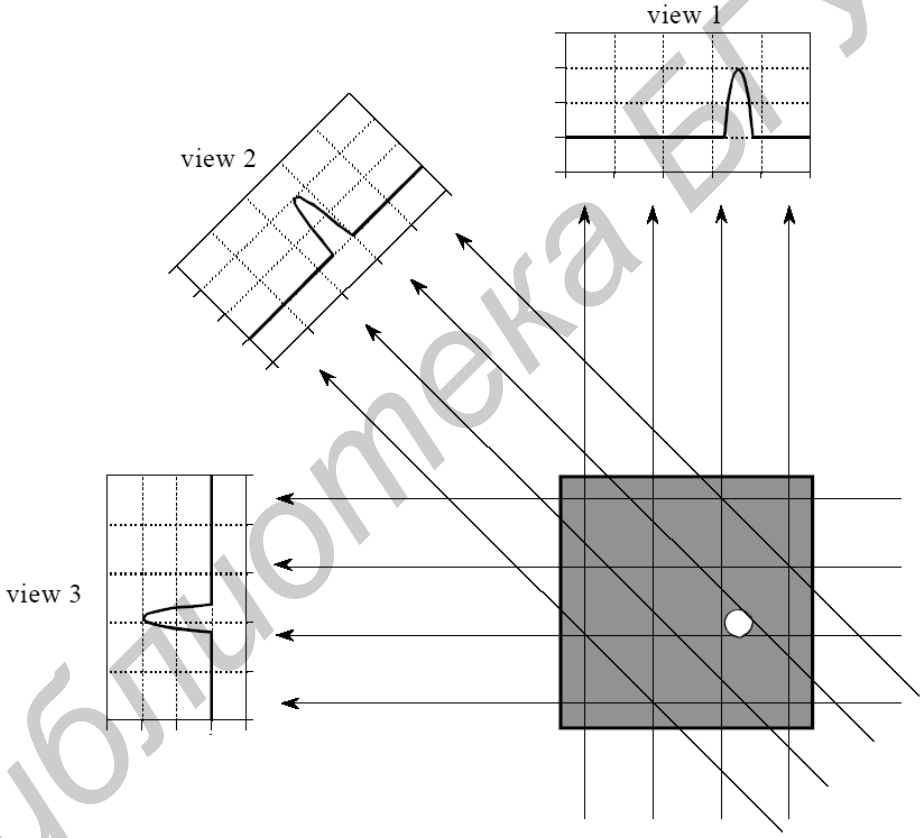


Рисунок 4.2 – Вид компьютерной томографии

Компьютерная томография получает набор видов и затем реконструирует соответствующее изображение. Каждый отсчет в виде равен сумме величин изображения вдоль луча, указывающего на отсчет. В рассматриваемом примере (см. рисунок 4.2) изображение есть маленькая коробочка (*pillbox*), окруженная нулями. Хотя здесь показано только три вида, обычно сканирование КТ использует десятки и сотни видов под слегка различными углами.

Существует четыре главных подхода для вычисления среза изображения из набора видов. Они называются реконструкционные алгоритмы КТ.

Первый метод основывается на решении большой системы линейных уравнений. Одно уравнение может быть записано для каждого измерения. То есть частный отсчет в частном профиле есть сумма частных групп пикселов в изображении. Для вычисления N^2 неизвестных переменных (т. е. величин пикселов в изображении) должно быть N^2 независимых уравнений и, следовательно, N^2 измерений. Большинство сканеров КТ требуют примерно на 50 % больше отсчетов, чем строгое требование этого анализа. Например, для реконструкции изображения 512×512 система должна сделать 700 видов из 600 отсчетов. Делая таким образом эту проблему *сверхдетерминированной*, финальное изображение имеет уменьшенные шумы и артефакты. Проблема этого первого метода реконструкции заключается в компьютерном времени. Решение системы из нескольких сотен тысяч линейных уравнений – непростая задача.

Второй метод реконструкции использует итеративный способ для вычисления финального изображения малыми шагами. Имеется несколько вариаций этого метода: способ алгебраической реконструкции (*Algebraic Reconstruction Technique – ART*), способ одновременной итеративной реконструкции (*Simultaneous Iterative Reconstruction – SIRT*) и итеративный способ наименьших квадратов (*Iterative Least Squares Technique – LIST*). Разница между этими методами заключается в методе выполнения последующей корректировки: луч за лучом, пиксел за пикселом, или одновременной коррекции целиком всех данных соответственно. В качестве примера реализации этих способов рассмотрим *ART*.

Все пикселы в массиве изображения установим в произвольную величину. Затем используется итеративная процедура для постепенного преобразования массива изображения к соответствующему профилю. Итерационный цикл состоит из циклов через каждую точку измеренных данных. Для каждой измеренной величины возникает вопрос: *как эти величины пикселов в массиве могут быть изменены, чтобы сделать их совместимыми с этим частным измерением?* Другими словами, измеренный отсчет сравнивается с суммой пикселов изображения вдоль луча, указывающего на отсчет. Если сумма вдоль луча ниже измеренного отсчета, все пикселы вдоль луча увеличиваются по величине. Аналогично, если сумма выше измеренного отсчета, величины всех пикселов вдоль луча уменьшаются. После завершения первого итерационного цикла еще имеются ошибки между суммой вдоль луча и измеренной величиной. Это вызвано тем, что изменение, сделанное для одного какого-либо измерения, нарушает все предыдущие корректировки. Идея заключается в том, что ошибки становятся меньше с повторением итераций, пока изображение не сойдется к правильному решению.

Итеративные способы обычно медленны, но они эффективны, когда лучшие алгоритмы не доступны. Действительно, *ART* использовался в первом коммерческом медицинском сканере КТ *EMI Marc I*, созданном в 1972 г. Развитие третьего

и четвертого метода почти целиком заменило итеративные способы в коммерческих продуктах КТ.

Два последних алгоритма реконструкции основываются на формальных математических решениях этой проблемы. Это элегантные примеры ЦОС.

Третий метод называется фильтрованное восстановление сцены по проекциям (*filtered backprojection*). Это модификация старого метода, называемого восстановлением сцены по проекциям или простым восстановлением сцены по проекциям. Рисунок 4.3 показывает, что простое восстановление сцены по проекциям является обычным подходом, но очень простым. Индивидуальный отсчет восстанавливается по проекциям путем установки всех пикселов изображения вдоль луча, указывающего на этот отсчет, в ту же самую величину. В более простых терминах, восстановление по проекциям формируется *размазыванием* каждого вида обратно через изображение в направлении, в котором этот вид был получен. Финальное изображение, восстанавливаемое по проекциям, получается как сумма всех видов, восстановленных по проекциям.

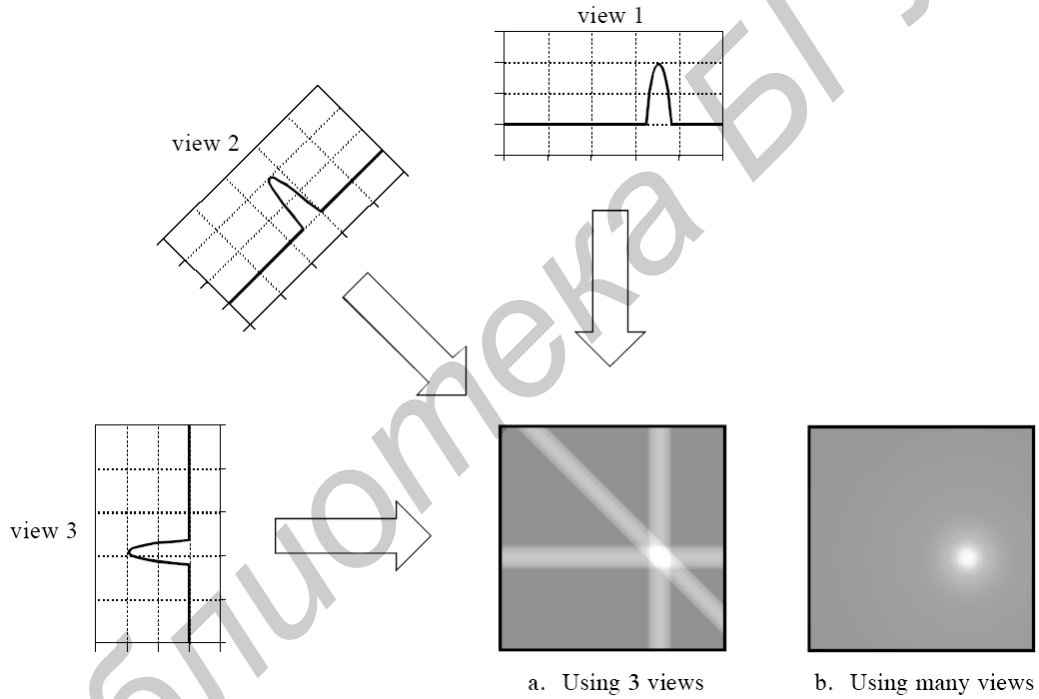


Рисунок 4.3 – Восстановление сцены по проекциям

Хотя восстановление сцены по проекциям концептуально просто, оно не корректно решает проблему. Как показано на рисунке 4.3, изображение, восстановленное по проекциям, очень *расплывчатое*. Одиночная точка в *правильном* изображении реконструируется как круглая область с уменьшением интенсивности к ее краям. В более формальных терминах функция расширения точки (*PSF*) для восстановления сцены по проекциям обладает круговой симметрией и затуханием соответственно ее радиусу.

Фильтрованное восстановление сцены по проекциям есть способ для корректировки расплывчатости, присущей простому восстановлению сцены

по проекциям. Как показано на рисунке 4.4, каждый вид *фильтруется* перед восстановлением для нейтрализации расплывчатости *PSF*. То есть каждый одномерный вид свертывается с одномерным ядром фильтра для создания набора *фильтрованных видов*. Затем фильтрованные виды восстанавливаются для реконструкции изображения, близкого к «правильному» изображению. Изображение, полученное этим методом, идентично «правильному» изображению, когда имеется *конечное* число видов и *конечное* число точек на вид. Данный метод является наиболее используемым алгоритмом в системах компьютерной томографии.

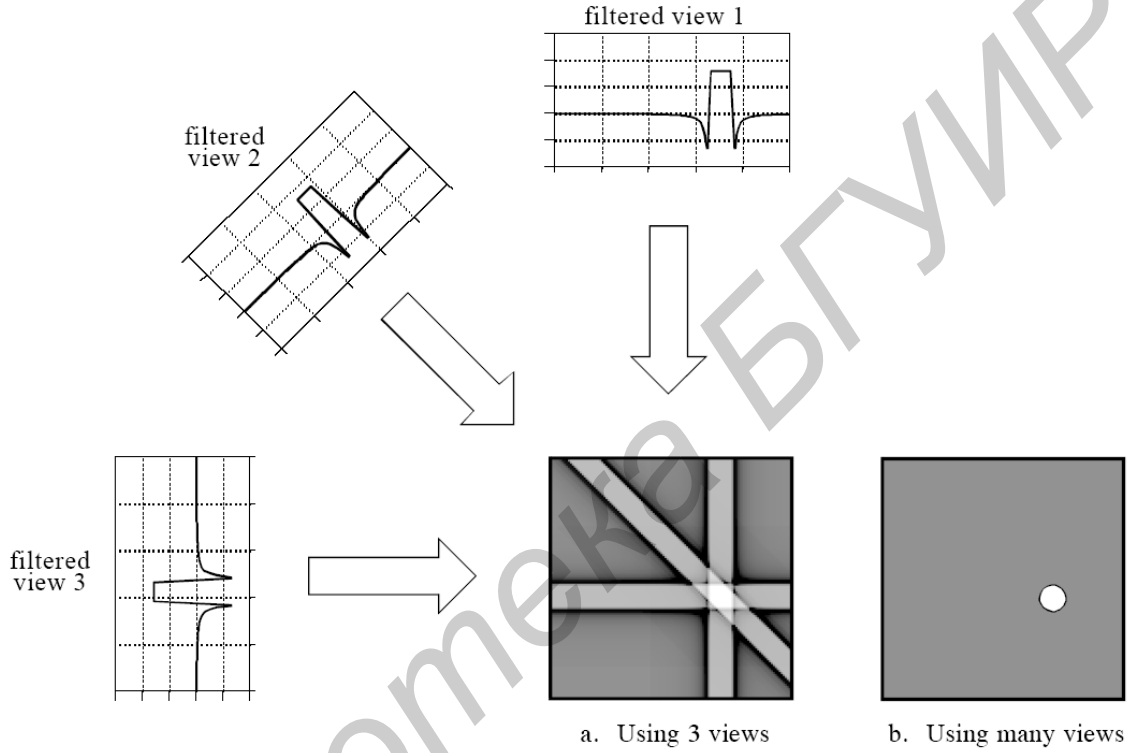


Рисунок 4.4 – Фильтрованное восстановление сцены по проекциям

Рассмотрим, как меняется изображение в результате фильтрации. Изображение в этом примере есть однородный белый круг, окруженный черным фоном (*pillbox*). Изображение полученных видов имеет плоский фон с округлой областью, представляющей белый круг. Фильтрация меняет вид в двух важных отношениях. Первое, вершина импульса делается плоской, в результате финальное восстановление создает *однородный* уровень сигнала внутри круга. Второе, по бокам импульса образуются отрицательные выбросы. При восстановлении эти отрицательные области нейтрализуют расплывчатость.

Четвертый метод называется реконструкцией Фурье. В пространственной области реконструкция включает в себя соотношение между двумерным изображением и одномерными видами. Путем взятия двумерного преобразования Фурье от изображения и одномерного преобразования Фурье от каждого вида проблема может быть переведена в частотную область. Как оказывается,

соотношение между изображением и его видами более простое в частотной области, чем в пространственной. Анализ этой проблемы в частотной области является вехой в технологии компьютерной томографии, называемой теоремой сечений Фурье.

Рисунок 4.5 показывает, как эта проблема выглядит в пространственной и частотной областях. В пространственной области каждый вид находится интегрированием изображения вдоль луча под определенным углом. В частотной области спектр изображения представлен двумерной решеткой. Спектр каждого вида (одномерный сигнал) представлен черной линией, наложенной на решетку.

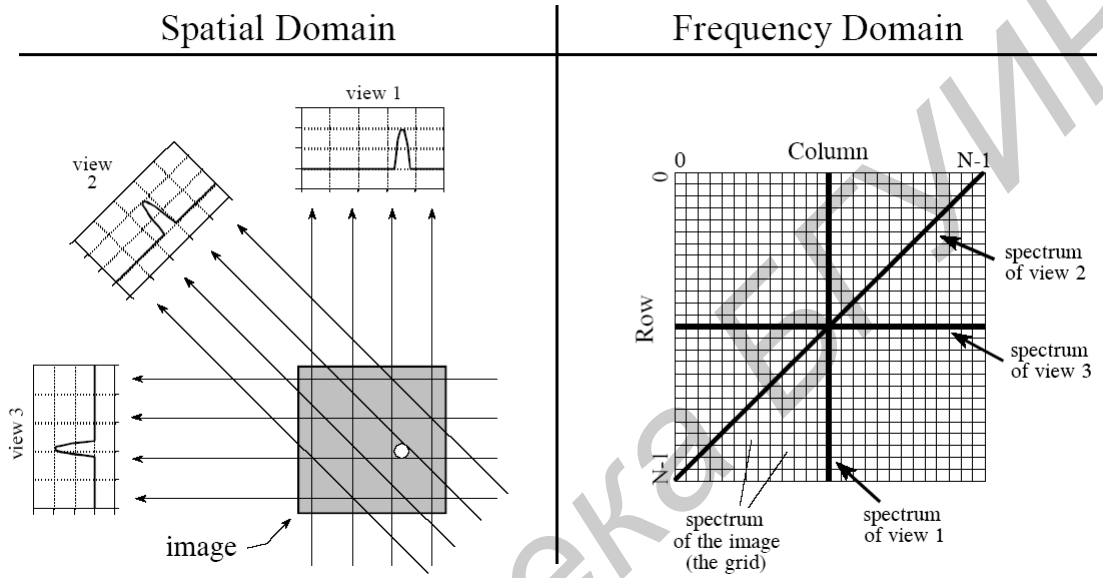


Рисунок 4.5 – Теорема сечений Фурье

Теорема сечений Фурье описывает соотношение между изображением и видом в частотной области. Она утверждает, что одномерный фурье-образ проекции представляет собой центральное сечение двумерного фурье-образа изображения. В пространственной области каждый вид находится интегрированием изображения вдоль луча под определенным углом. В частотной области спектр каждого вида есть одномерное «сечение» двумерного спектра изображения. Например, спектр вида 1 (*spectrum of view* 1) есть то же самое, что и спектр центральной колонки изображения. Отметим, что спектр каждого вида расположен на решетке под тем же самым углом, под которым был получен этот вид. Все частотные спектры включают отрицательные частоты и располагаются так, чтобы нулевая частота была в центре.

Реконструкция изображения требует трех шагов. Во-первых, берется одномерное преобразование Фурье от каждого вида. Во-вторых, эти спектры видов используются для вычисления двумерного частотного спектра изображения в соответствии с теоремой сечений Фурье. Так как спектр видов организован *радиально*, а правильный спектр изображения организован *прямоугольно*, то

требуется интерполяционное вращение для преобразования. В-третьих, берется обратное БПФ для получения изображения.

Преобразование «радиального к прямоугольному» также является ключом для понимания фильтрованного восстановления сцены по проекциям. Радиальное расположение есть спектр восстановленного изображения, в то время как прямоугольная решетка есть спектр правильного изображения. Если мы сравним одну маленькую область радиального спектра с соответствующей областью прямоугольной решетки, то мы обнаружим, что величины отсчетов идентичны. Однако они имеют разную *плотность отсчетов*. Правильный спектр имеет равномерные промежутки между точками во всех направлениях, как показывает равномерное расположение прямоугольной решетки. Для сравнения, восстановленный спектр имеет более высокую плотность отсчетов вблизи центра из-за своей радиальной природы. Например, как спицы в колесе сходятся ближе около ступицы. Это явление не влияет на реконструкцию Фурье, так как интерполируются *величины* самых ближайших соседей, а не их *плотность*. Фильтр при фильтрованном восстановлении по проекциям аннулирует эту неравномерную плотность отсчетов.

В частности, частотный отклик фильтра должен быть *обратным* по отношению к плотности отсчетов. Поскольку восстановленный спектр имеет плотность $1/f$, соответствующий фильтр имеет частотный отклик $H[f] = f$, который показан на рисунке 4.6 (*Frequency respone*).

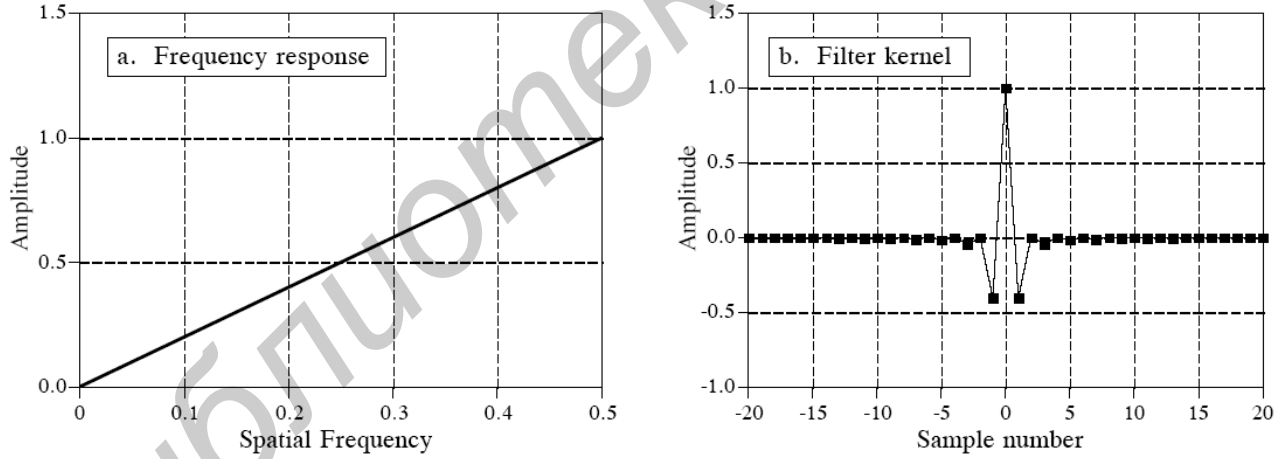


Рисунок 4.6 – Фильтр для восстановления сцены по проекциям

В математическом виде ядро фильтра описывается следующей формулой:

$$\begin{cases} h[0] = 1, \\ h[k] = 0 \end{cases} \quad \text{для четных величин } k,$$

$$h[k] = \frac{-4/\pi^2}{k^2} \quad \text{для нечетных величин } k.$$

Ядро фильтра для фильтрованного восстановления сцены по проекциям показано на рисунке 4.6 (*Filter kernel*).

4.4 Практическая часть

Реконструкция изображений по их проекционным данным

В данной лабораторной работе рассмотрим применение функций *radon*, *iradon*, *fanbeam* и *ifanbeam* для получения проекций изображений и обратную реконструкцию данных на основе их проекций. В функциях *radon* и *iradon* для получения проекций используется параллельно-лучевая геометрия, а в функциях *fanbeam* и *ifanbeam* – веерно-лучевая геометрия. Для того чтобы сравнить параллельно-лучевую и веерно-лучевую геометрии, создадим некоторые синтезированные проекции для каждой геометрии и затем применим их для восстановления исходного изображения.

Практической задачей применения реконструкции изображений является использование рентгеновской томографии, где проекции формируются на основании измерения ослабления радиации при прохождении через разные физические среды. В результате на изображении отображаются данные о структуре среза исследуемого объекта. Проекционные данные формируются и хранятся в специальной медицинской аппаратуре. На основании этих данных существует возможность реконструкции изображения исследуемых органов. В среде *MATLAB* моделирование восстановления вида объекта по его томографическим проекциям можно реализовать с помощью функций *iradon* и *ifanbeam*.

Функция *iradon* применяется для реконструкции изображений из параллельно-лучевых проекций. В параллельно-лучевой геометрии каждая проекция формируется в результате линейной комбинации изображений, представляющих некоторый исследуемый орган под разными углами. Функция *ifanbeam* применяется для реконструкции данных на основании их веерно-лучевых проекций, которые получены из одного источника излучения, но зафиксированы несколькими датчиками.

Рассмотрим применение пакета ***Image Processing Toolbox*** для моделирования описанных выше процедур при использовании обеих геометрий.

4.4.1 Создание фантома головы

Тестовое изображение фантома головы Шеппа – Логана может быть сгенерировано с помощью функции *phantom*. Изображение фантома иллюстрирует много особенностей, которые в действительности свойственны реальным томографическим изображениям головы человека. Размещенные в центре фантома объекты в виде эллипса моделируют реальные внутренние особенности строения головы человека (рисунок 4.7).

```
P=phantom(256);
```

```
imshow(P)P = phantom(256);
imshow(P)
```

Рисунок 4.7 – Созданный фантом

4.4.2 Параллельные лучи – вычисление синтезированных проекций

Рассмотрим вычисление синтезированных проекций на основе параллельно-лучевой геометрии с разным числом углов. Для обработки этих данных используется функция radon, а результат представляется в виде матрицы, в которой каждый столбец представляет собой преобразование Радона для одного значения угла.

```
% Получение набора проекций при изменении угла поворота
% сканирующего устройства от 0 до 170 градусов.
% Измерения проводятся через каждые 10 градусов:
theta1 = 0:10:170;
[R1, xp] = radon(P, theta1);
num_angles_R1 = size(R1, 2)
num_angles_R1 = 18
% В итоге получено 18 проекций фантома

% Получение набора проекций при изменении угла поворота
% сканирующего устройства от 0 до 175 градусов.
% Измерения проводятся через каждые 5 градусов:

theta2 = 0:5:175;
[R2, xp] = radon(P, theta2);
num_angles_R2 = size(R2, 2)
num_angles_R2 = 36
% В итоге получено 36 проекций фантома

% Получение набора проекций при изменении угла поворота
% сканирующего устройства от 0 до 178 градусов.
```

```
% Измерения проводятся через каждые 2 градуса:

theta3=0:2:178;
[R3, xp] = radon(P, theta3);
num_angles_R3 = size(R3, 2)
num_angles_R3 = 90
% В итоге получено 90 проекций фантома
```

Отметим, что для каждого угла проекция вычисляется на основании *N* точек вдоль оси *xp*, где *N* является константой, которая зависит от расстояния до всех точек изображения, спроектированных под разными углами.

```
N_R1=size(R1, 1)
N_R2=size(R2, 1)
N_R3=size(R3, 1)
N_R1=
   367
N_R2=
   367
N_R3=
   367
```

Таким образом, когда мы используем меньший фантом головы, при вычислении проекций требуется меньшее число точек вдоль оси *xp*.

```
P_128=phantom(128);
[R_128, xp_128]=radon(P_128, theta1);
N_128=size(R_128, 1)
N_128=
   185
```

Отобразим проекционные данные *R*3. Некоторые свойства изображения исходного фантома видны на изображении *R*3.

Первый столбец *R*3 соответствует проекциям нулевого угла, которые интегрированы в вертикальном направлении.

Последний столбец соответствует проекциям под 90°, которые просуммированы в горизонтальном направлении.

```
figure, imagesc(theta3, xp, R3)
colormap(hot)
colorbar
xlabel('Параллельный поворот угла - \theta');
ylabel(' Параллельное расположение датчиков x\ ');
```

На изображении представлена зависимость местонахождения сенсоров от поворота угла.

4.4.3 Параллельные лучи – реконструкция фантома головы на основании проекционных данных

Для каждого параллельного поворота-приращения при реконструкции используется соответствующая синтезированная проекция. При практической реализации реконструкции всегда известна геометрия источников и сенсоров.

Следующие три примера реконструкции (*I*1, *I*2 и *I*3) демонстрируют эффект изменения числа углов, по которым сделаны проекции. Для *I*1 (рисунок 4.8) и *I*2 (рисунок 4.9) некоторые особенности, которые четко заметны на исходном фантоме, видны не очень качественно. Что касается, эллипсообразных предметов, то они достаточно хорошо заметны на каждом изображении. Результирующее изображение *I*3 (рисунок 4.10) наиболее точно соответствует исходному изображению.

```
% Построение результирующей размерности
% при реконструкции с использованием
% размеров исходного изображения P.
output_size=max(size(P));
dtheta1=theta1(2)-theta1(1);
I1=iradon(R1, dtheta1, output_size);
figure, imshow(I1)
```

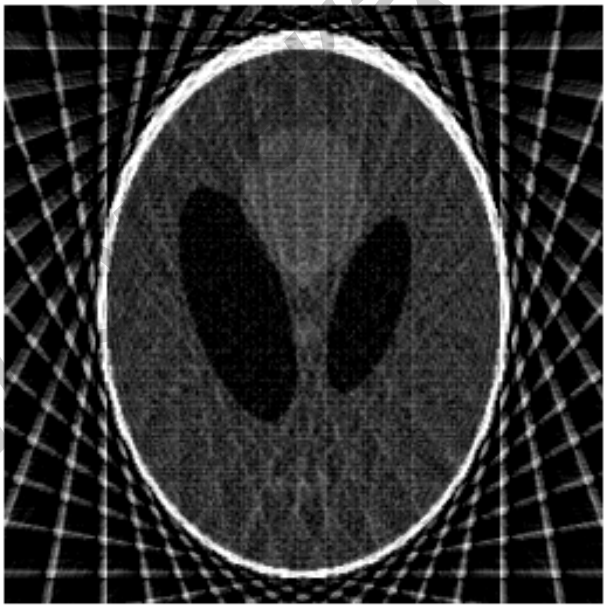


Рисунок 4.8 – Реконструкция фантома по 18 проекциям

```
dtheta2 = theta2(2) - theta2(1);
I2 = iradon(R2,dtheta2,output_size);
figure, imshow(I2)
```

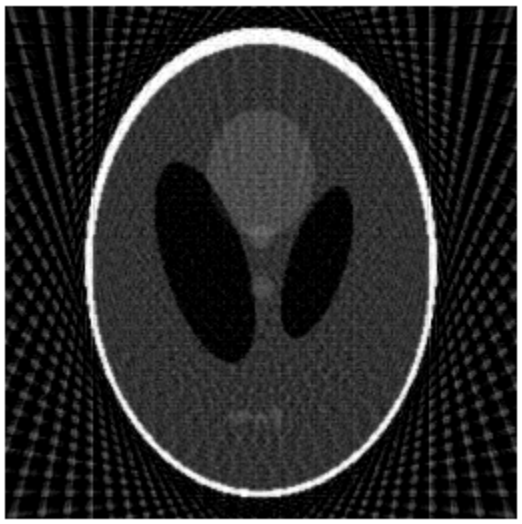


Рисунок 4.9 – Реконструкция фантома по 36 проекциям

Как уже отмечалось выше, на изображениях *I*1 и *I*2 есть некоторые артефакты, которые искажают действительную информацию. Для устранения этих артефактов используют большее число углов.

```
dtheta3 = theta3(2) - theta3(1);
I3 = iradon(R3,dtheta3,output_size);
figure, imshow(I3)
```

Рисунок 4.10 – Реконструкция фантома по 90 проекциям

4.4.4 Веерные лучи – вычисление синтезированных проекций

Рассмотрим вычисление синтезированных проекций с использованием веерно-лучевой геометрии и переменной '***FanSensorSpacing***'.

```
D=250;
dsensor1=2;
F1=fanbeam(P, D, 'FanSensorSpacing', dsensor1);
dsensor2=1;
F2=fanbeam(P, D, 'FanSensorSpacing', dsensor2);
dsensor3=0.25;
[F3, sensor_pos3, fan_rot_angles3]=fanbeam(P, D, 'FanSensor-
Spacing', dsensor3);
```

Отобразим проекционные данные *F3*. Отметим, что веерный угловой поворот в диапазоне от 0 до 360° может быть преобразован в модель с использованием 180°, поскольку некоторые свойства будут идентичны с обеих сторон.

Некоторые свойства на изображении при веерно-лучевых проекциях могут быть связаны с некоторыми свойствами на изображениях параллельно-лучевых проекций и наоборот.

```
figure, imagesc(fan_rot_angles3, sensor_pos3, F3)
colormap(hot)
colorbar
xlabel('Веерный поворот угла (в градусах) ')
ylabel('Веерное расположение датчиков (в градусах)')
```

4.4.5 Веерные лучи – реконструкция фантома головы на основании проекционных данных

Каждому веерному расположению сенсоров в пространстве соответствует своя синтезированная проекция. При практической реализации реконструкции всегда известна геометрия источников и сенсоров изображения *P*.

Изменение значения '*FanSensorSpacing*' приводит к изменению числа сенсоров, которые используются при изменении угла. При каждой веерно-лучевой реконструкции используется поворот на все значения угла. Использование данных при различных поворотах угла влияет на качество результата обработки (рисунки 4.11, 4.12, 4.13).

Отметим, что '*FanSensorSpacing*' является только одним параметром из нескольких, которые можно контролировать при использовании функций *fanbeam* и *ifanbeam*. Также существует возможность конвертации между параллельными и веерно-параллельными данными. Для этого используют функции *fan*2*para* и *para*2*fan*.

```
Ifan1=ifanbeam(F1, D, 'FanSensorSpacing', dsensor1, 'Out-
putSize', output_size);
figure, imshow(Ifan1)
```

Рисунок 4.11 – Реконструкция фантома по 18 проекциям с помощью веерных лучей

```
    Ifan2=ifanbeam(F2, D, 'FanSensorSpacing', dsensor2, 'Out-
putSize', output_size);
    figure, imshow(Ifan2)
```

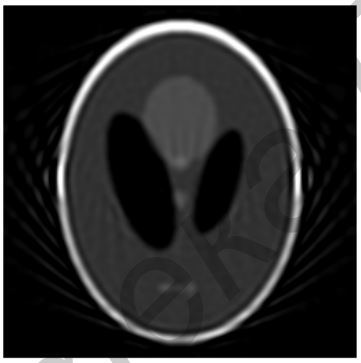


Рисунок 4.12 – Реконструкция фантома по 36 проекциям с помощью веерных лучей

```
    Ifan3=ifanbeam(F3, D, 'FanSensorSpacing', dsensor3, 'Out-
putSize', output_size);
    figure, imshow(Ifan3)
```

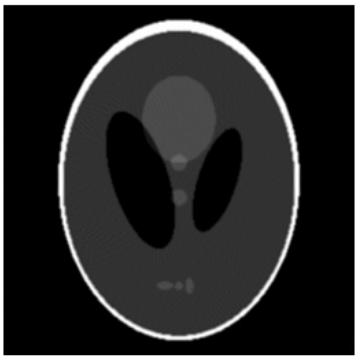


Рисунок 4.13 – Реконструкция фантома по 90 проекциям с помощью веерных лучей

4.4.6 Извлечение данных из трехмерных магниторезонансных изображений

Функции *imtransform* и *tformarray* могут быть использованы при интерполяции и других преобразованиях трехмерных магниторезонансных изображений, особенно при получении числовых характеристик.

Просмотр демонстрационного примера

Демонстрация включает следующие шаги.

Шаг 1: считывание и просмотр магниторезонансных изображений (горизонтальные срезы).

Шаг 2: получение сагиттальных данных на основе горизонтальных срезов с использованием функции *imtransform*.

Шаг 3: получение сагиттальных данных на основе горизонтальных срезов с использованием функции *tformarray*.

Шаг 4: получение сагиттальных данных и отображение их в виде последовательности.

Шаг 5: получение корональных данных и отображение их в виде последовательности.

Шаг 1: считывание и просмотр магниторезонансных изображений (горизонтальные срезы)

В этом примере используются магниторезонансные данные, которые представлены в *MATLAB* и используются при описании функций *montage* и *immovie*. Считывание магниторезонансных изображений добавляет две переменные в рабочее пространство *D* (128×128×1×27, в формате *unit*8) и полутоновую палитру map (89×3, в формате *double*).

Массив *D* включает 27 горизонтальных срезов магниторезонансных данных сканирования человеческого черепа размерностью 128×128. Значения элементов массива *D* находятся в диапазоне от 0 до 88. Таким образом, палитра обеспечивает генерацию изображения в диапазоне, пригодном для визуального анализа. Размерность данных в массиве *D* должна быть согласована с функцией *immovie*. Первые две размерности являются пространственными. Третья размерность представляет собой размерность цвета. Числа третьей размерности указывают на индексы в палитре, например *size* (*D*, 3), и используются для описания *RGB*-составляющих цвета. Четвертая размерность является временной и только в очень редких случаях используется в качестве пространственной координаты. Приведенные выше пространственные размерности в массиве *D* используются в функциях *imtransform* или *tformarray* для преобразования слайдов горизонтальных срезов в сагиттальные данные. Пространственные размерности массива *D* размещены в следующей последовательности (рисунок 4.14).

Размерность 1: от передней до задней части головы (ростральная (rostral) и каудальная (caudal) части).

Размерность 2: с левой части головы на правую.

Размерность 4: от верхней части головы к нижней (*inferior* и *superior*).

Данные, которые представляют собой набор из 27 горизонтальных срезов, подвергаются последующей обработке. Для предотвращения непредсказуемых ситуаций при обработке данных в примере используется функция *iptgetpref*.

```
truesizewarning=iptgetpref('TruesizeWarning');
iptsetpref('TruesizeWarning', 'off');
load mri;
figure;
immovie(D, map);
montage(D, map);
title('Horizontal Slices');
```

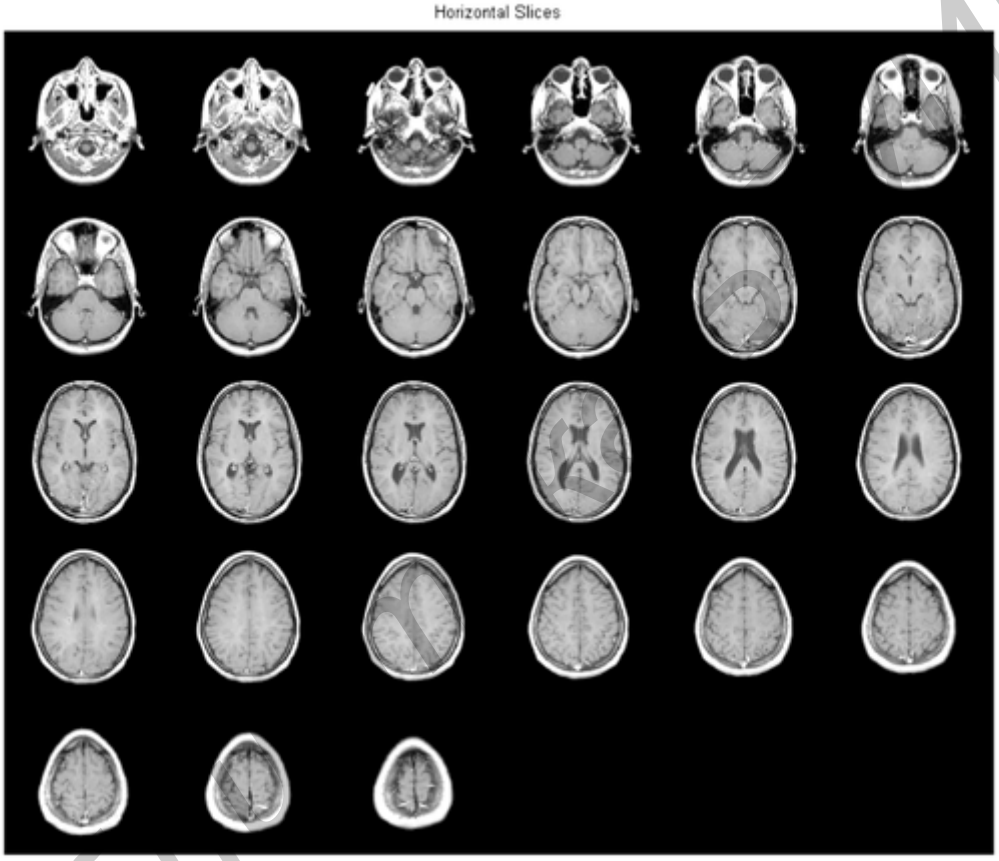


Рисунок 4.14 – Горизонтальные срезы магниторезонансной томографии

Шаг 2: получение сагиттальных данных на основе горизонтальных срезов с использованием функции imtransform

Существует возможность создания среднесагиттальных срезов, полученных на основе магниторезонансных данных, взятых из массива *D* и преобразованных с учетом анализа разных интервалов среза и пространственной ориентации.

Согласно сказанному выше, получение среднесагиттальных срезов проводят следующим образом.

```
M1=D(:, 64, :, :); size(M1)
```

Также существует возможность просмотра *M*1 как изображения размерностью 128×1×1×27. С использованием функции *reshape* данные *M*1 можно преобразовать в изображение размерностью 128×27 с дальнейшим просмотром с помощью *imshow*.

```
M2 = reshape(M1,[128 27]); size(M2)
figure, imshow(M2,map);
title('Sagittal - Raw Data');
ans =
   128    27
```

Размерности массива *M*2 следующие.

Размерность 1: от передней до задней части головы (ростральная (*rostral*) и каудальная (*caudal*) части).

Размерность 2: от верхней части головы к нижней (*inferior* и *superior*).

Качество визуального просмотра можно улучшить, изменив пространственное расположение (ориентацию) изображения и изменив масштаб вдоль одной из осей с коэффициентом 2,5. Для улучшения визуального просмотра также используются аффинные преобразования.

```
T0=maketform('affine', [0 -2,5; 1 0; 0 0])
```

Параметр, который применяется при реализации функции *maketform* и представляет собой блок 2×2:

```
[ 0 -2,5
1 0 ]
```

характеризует описанные выше преобразования поворота и масштабирования.

После преобразований получим следующее.

Размерность 1: от верхней части головы к нижней (*inferior* и *superior*).

Размерность 2: от передней до задней части головы (ростральная (*rostral*) и каудальная (*caudal*) части).

Выражение *imtransform(M2, T0, 'cubic')* применяется для преобразования данных из массива *T* в *M*2 и обеспечения приемлемой разрешительной способности при интерполяции вдоль направления сверху вниз. Однако при такого рода преобразованиях отпадает необходимость в проведении кубической интерполяции в направлении от передней части головы к задней. Поэтому окрестности в этом направлении определяются с большей степенью эффективности.

```
R2 = makeresampler({'cubic','nearest'},'fill');
M3 = imtransform(M2,T0,R2);
figure, imshow(M3,map);
title('Sagittal - IMTRANSFORM')
```

Шаг 3: получение сагиттальных данных на основе горизонтальных срезов с использованием функции tformarray

На данном этапе мы получим результаты, аналогичные результатам, полученным при выполнении второго шага алгоритма. Только теперь для преобразования трехмерных данных в двумерные, а также для других операций будем использовать функцию *tformarray*. В шаге 2 начальные данные представляют собой трехмерный массив, а результат обработки представлен в виде двумерного массива. При преобразованиях используется функция *imtransform*, которая создает массив *M*3, а также промежуточные двумерные массивы *M*1 и *M*2.

При использовании аргумента типа *TDIMS_A* в функции *tformarray* существует возможность проведения определенных преобразований для исходного массива. В результате этих преобразований получится изображение с такими размерностями.

Размерность 1: сверху вниз (исходная размерность).

Размерность 2: от передней до задней части головы (ростральная (*rostral*) и каудальная (*caudal*) части – исходная размерность).

Таким образом, получим примерный сагиттальный вид исходной размерностью 2 при параметре *tdims_a* = [4 1 2]. Параметр *tform* можно получить, проводя двумерные аффинные преобразования, которые заключаются в масштабировании размерности 1 с коэффициентом 2,5 и прибавлении к массиву значения 68,5. Вторая часть преобразований, которые заключаются в получении 64-го сагиттального плана, описывается функцией *INVERSE_FCN*.

Определение коэффициентов *T*2 и *Tc*.

```
T1 = maketform('affine',[-2,5 0; 0 1; 68,5 0]);
inverseFcn = @(X,t) [X repmat(t.tdata,[size(X,1) 1])];
T2 = maketform('custom',3,2,[],inverseFcn,64);
Tc = maketform('composite',T1,T2)
```

Проведем некоторые преобразования для определения третьей размерности.

```
R3=makeresampler({'cubic', 'nearest', 'nearest'}, 'fill');
```

Функция *tformarray* преобразовывает третью пространственную размерность, и массив *D* трансформируется в двумерные данные. Таким образом, исходное изображение размерами 66×128 с 27 видов растягивается до 66 в вертикальном направлении.

```
M4=tformarray(D, Tc, R3, [4 1 2], [1 2], [66 128], [], 0)
```

Полученный результат идентичен результату, полученному при обработке функцией *imtransform*.

```
figure, imshow(M4, map);
title('Sagittal - TFORMARRAY')
```

Шаг 4: получение сагиттальных данных и отображение их в виде последовательности

Создадим четырехмерный массив (трехмерные данные и размерность цветности), который используется для генерации движений слева направо с использованием более 30 снимков. Массив преобразований имеет следующую структуру.

Размерность 1: сверху вниз.

Размерность 2: от передней до задней части головы (ростральная (*rostral*) и каудальная (*caudal*) части).

Размерность 4: слева направо.

Сначала проведем следующие преобразования – трансформируем исходный массив с использованием параметра *TDIMS_A* = [4 1 2]. Преобразования будут заключаться в перестановке и масштабировании элементов массива в вертикальной размерности.

```
T3=maketform('affine', [-2,5 0 0; 0 1 0; 0 0 0,5; 68,5 0 -14])
```

Теперь в массиве преобразований *tformarray* параметр *TSIZE_B*=[66 128 35] включает 35 четырехразмерных слайдов.

```
S=tformarray(D, T3, R3, [4 1 2], [1 2 4], [66 128 35], [], 0)
```

Для просмотра движения сагиттальных слайдов они соответствующим образом монтируются.

```
figure;
 immovie(S, map);
 S2=padarray(S, [6 0 0 0], 0, 'both');
 montage(S2, map);
 title('Sagittal Slices');
```

Шаг 5: получение корональных данных и отображение их в виде последовательности

Создание корональных слайдов почти аналогично созданию сагиттальных. Для этого изменим параметр *TDIMS_A* из [4 1 2] на [4 2 1]. Создадим серию из 45 слайдов и смонтируем из них движения (рисунок 4.15). Размерности результирующего массива следующие.

Размерность 1: сверху вниз.

Размерность 2: слева направо.

Размерность 4: от задней до передней части головы (ростральная (*rostral*) и каудальная (*caudal*) части).

```
T4=maketform('affine', [-2,5 0 0; 0 1 0; 0 0 -0,5; 68,5 0 61]);
```

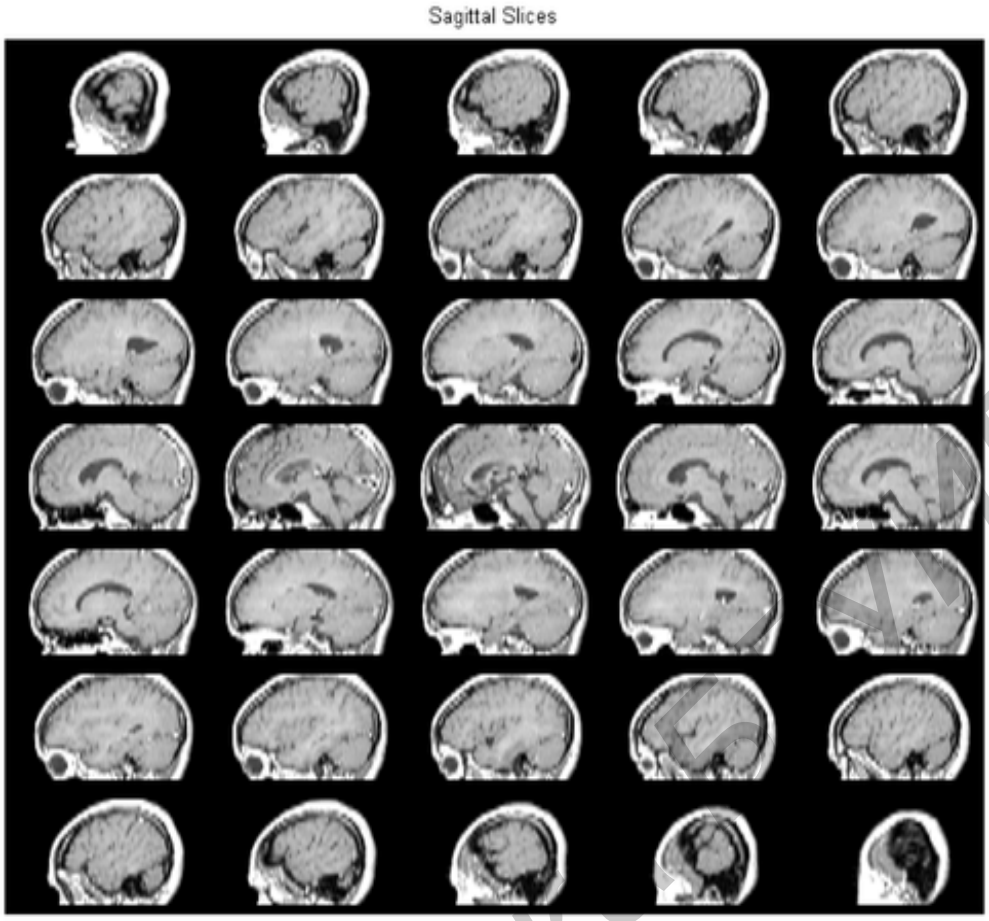


Рисунок 4.15 – Сагиттальные срезы магниторезонансной томограммы

В функции *tformarray* параметр *TSIZE_B* = [66 128 48] описывает размерности изображений.

```
C=tformarray(D, T4, R3, [4 2 1], [1 2 4], [66 128 45], [], 0)
```

Для просмотра движения кадры соответствующим образом монтируются (рисунок 4.16).

```
figure;
immovie(C, map);
C2=padarray(C, [6 0 0 0], 0, 'both');
montage(C2, map);
title('Coronal Slices')
```

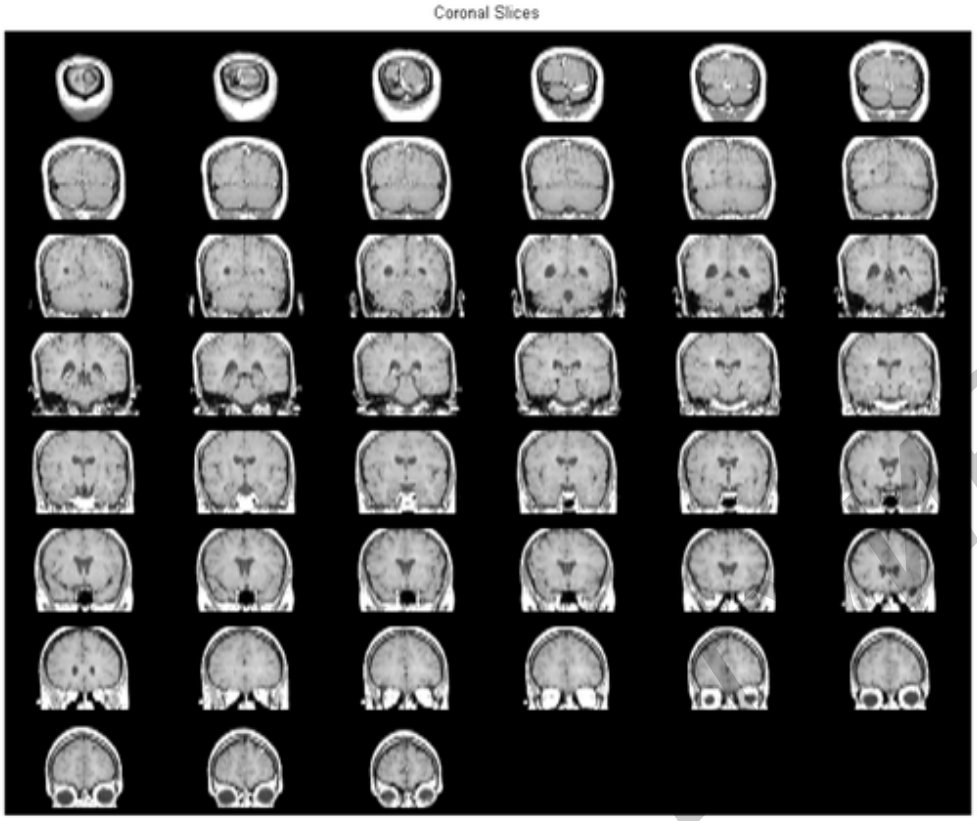


Рисунок 4.16 – Корональные срезы магниторезонансной томограммы

Отметим, что большинство операций преобразования описанных ранее массивов возможно корректировать в ручном режиме.

```
iptsetpref('TruesizeWarning', truesizewarning)
```

4.5 Содержание отчета

1 Цель работы.
2 Таблица по варианту.
3 Листинги программ восстановления сцены по проекциям.
4 Листинг программ и результаты исследования МРТ-изображений.
5 Выводы.

4.6 Контрольные вопросы

1 Каковы принципы получения проекций томографического изображения?
2 Как происходит реконструкция фантома головы на основании проекционных данных?
3 Каковы особенности синтеза проекций при использовании параллельных лучей?
4 Каковы особенности синтеза проекций при использовании веерных лучей?

Св. план 2014, поз. 15

Учебное издание

Давыдов Максим Викторович
Бондарик Василий Михайлович
Давыдова Надежда Сергеевна
Терех Александр Сергеевич

ЦИФРОВАЯ ОБРАБОТКА БИОМЕДИЦИНСКИХ СИГНАЛОВ И ИЗОБРАЖЕНИЙ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *Е. И. Герман*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *М. В. Гуртатовская*

Подписано в печать 19.06.2015. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс». Отпечатано на ризографе. Усл. печ. л. 4,53. Уч.-изд. л. 4,5. Тираж 100 экз. Заказ 118.

Издатель и полиграфическое исполнение: учреждение образования «Белорусский государственный университет информатики и радиоэлектроники». Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий №1/238 от 24.03.2014, №2/113 от 07.04.2014, №3/615 от 07.04.2014. ЛП №02330/264 от 14.04.2014. 220013, Минск, П. Бровки, 6