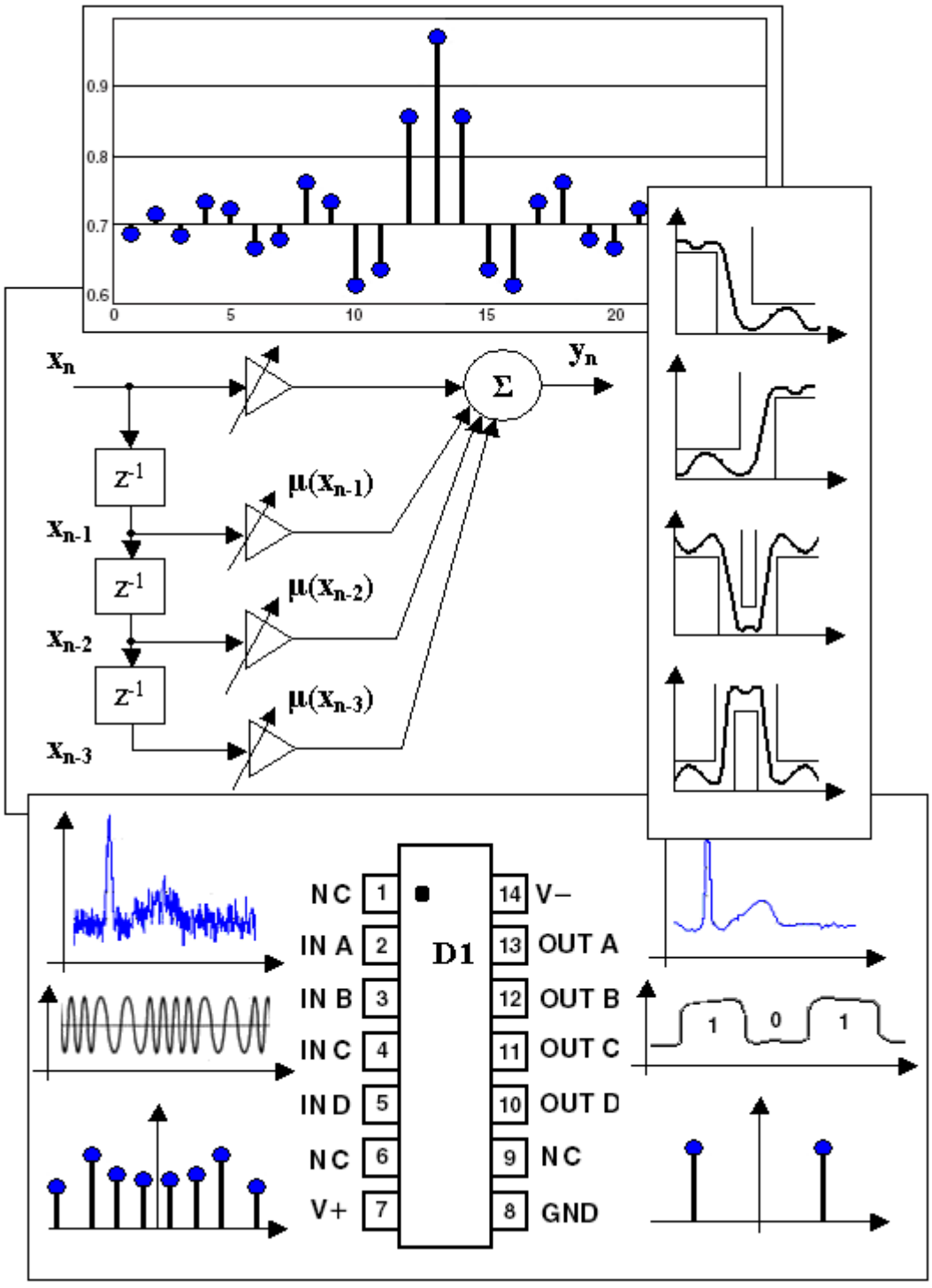
Титов Д. А., Василевский В. В., Косых А. В.

ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ

Методические указания к лабораторным работам



Омск-2011

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Омский государственный технический университет

Титов Д. А., Василевский В. В., Косых А. В.

ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ

Методические указания к лабораторным работам

Омск 2011

Составители: Д. А. Титов, к. т. н.,
В. В. Василевский, к. т. н.,
А. В. Косых, д. т. н.

Рецензент:

Данные методические указания обеспечивают выполнение шести лабораторных работ, охватывающих важнейшие аспекты цифровой обработки сигналов (проектирование цифровых фильтров, дискретное преобразование Фурье, адаптивная фильтрация сигналов, эффекты, обусловленные ограниченной разрядностью вычислителя). Лабораторные работы выполняются в программной среде Matlab 6.5.

Методические указания предназначены для студентов электро- и радиотехнических специальностей очной и заочной форм обучения.

Печатается по решению редакционно-издательского совета Омского государственного технического университета.

Лабораторная работа № 1

Проектирование цифровых фильтров.

1. Цель работы

Целью работы является изучение методов синтеза цифровых фильтров и основ проектирования цифровых фильтров в специализированных графических средах системы MATLAB.

2. Пояснения к работе

Под проектированием (или синтезом) цифрового фильтра понимается выбор таких наборов коэффициентов $\{a_i\}$ и $\{b_i\}$, при которых характеристики получающегося фильтра удовлетворяют заданным требованиям.

Методы синтеза цифровых фильтров можно классифицировать по различным признакам:

- По типу получаемого фильтра:
  - методы синтеза *рекурсивных* фильтров;
  - методы синтеза *нерекурсивных* фильтров;
- По наличию аналогового прототипа:
  - методы синтеза *с использованием* аналогового прототипа;
  - *прямые* (без использования аналогового прототипа) методы синтеза.

Название «прямые методы» означает, что в данном случае не используется аналоговый прототип. Исходными данными для синтеза служат какие-либо параметры фильтра (чаще всего амплитудно-частотная характеристика (АЧХ)), которые могут задаваться произвольно.

При использовании метода инвариантной импульсной характеристики происходит дискретизация импульсной характеристики аналогового прототипа. Частотная характеристика получающегося дискретного фильтра, соответственно, представляет собой периодически повторенную частотную характеристику аналогового прототипа. По этой причине данный метод непригоден для синтеза фильтров верхних частот (ФВЧ) и вообще фильтров, коэффициент передачи которых не стремится к нулю с ростом частоты. Метод инвариантной импульсной характеристики реализован в пакете Signal Processing с помощью функции `impinvar`.

При использовании метода билинейного z-преобразования происходит искажение характеристики аналогового прототипа только вдоль частотной оси. При этом частотный диапазон аналогового фильтра (от нуля до бесконечности) преобразуется к рабочему диапазону частот дискретного фильтра (от нуля до половины частоты дискретизации). Данный метод реализуется с помощью функции `bilinear` для произвольного аналогового прототипа. Кроме того, для расчета фильтров нижних частот (ФНЧ), ФВЧ полосовых и режекторных фильтров методом билинейного z-преобразования имеются следующие готовые функции:

`butter(n, w0, type)` – расчет фильтров Баттерворта;

`cheby1(n, Rp, w0, type)` – расчет фильтров Чебышева первого рода;

`cheby2(n, Rs, w0, type)` – расчет фильтров Чебышева второго рода;

`ellip(n, Rp, Rs, w0, type)` – расчет эллиптических фильтров (фильтров Золотарева -Кауэра).

В представленных записях `n` – порядок рассчитываемого фильтра, `w0` – частота среза (нормированная относительно `Fs*2`), `Rp` – уровень пульсаций АЧХ в полосе пропускания (дБ), `Rs` – уровень пульсаций АЧХ в полосе задерживания (дБ), `type` – параметр, определяющий тип фильтра. Для ФНЧ параметр `type` отсутствует, для ФВЧ `type = 'high'`. В случаях полосового и режекторного фильтров - `w0` – двухэлементный вектор частот среза. Для полосового фильтра параметр `type` отсутствует, для режекторного `type = 'stop'`.

Имеются также функции определения требуемого порядка этих фильтров по заданным параметрам АЧХ (граничным частотам полос пропускания и задерживания, а также допустимым пульсациям в этих полосах). Это соответственно функции `buttord`, `cheb1ord`, `cheb2ord`, `ellipord`.

В качестве примера синтезируем эллиптический ФНЧ четвертого порядка с частотой среза 3 кГц, пульсациями АЧХ в полосе пропускания 1 дБ, и подавлением сигнала в полосе задерживания 20 дБ. Частоту дискретизации примем равной 12 кГц. После синтеза построим графики АЧХ и фазочастотной характеристики (ФЧХ) полученного фильтра с помощью функции `freqz`.

```
>> Fs = 12000;                          % частота дискретизации
>> F0 = 3000;                           % частота среза
>> [b, a] = ellip(4, 1, 20, F0/Fs*2);   % Вычисление коэффициентов
>> freqz(b, a, Fs);                     % вывод графиков
```

Результаты выполнения команд представлены на рис. 1.1

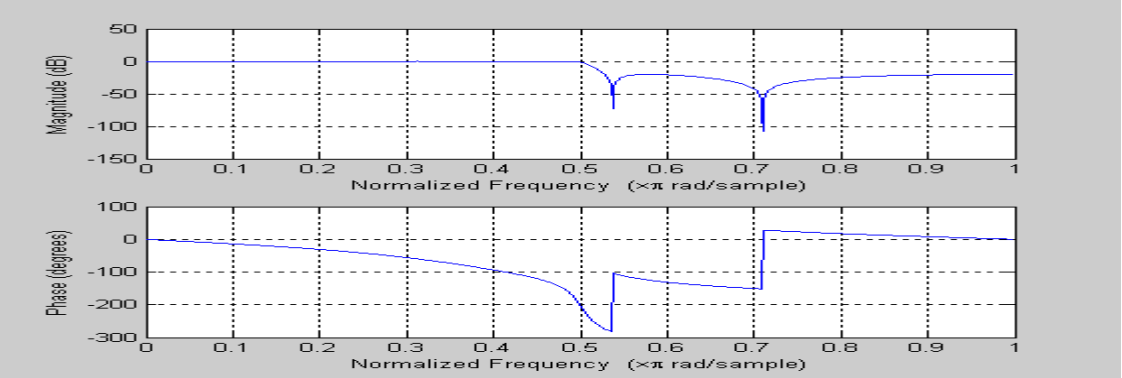


Рис. 1.1

Методы синтеза, не использующие аналоговый прототип, называются прямыми. К функциям прямого синтеза нерекурсивных фильтров относятся следующие:

Функции, реализующие синтез фильтров путем обратного преобразования Фурье от желаемой АЧХ с последующим умножением получившейся импульсной характеристики на некоторую весовую функцию (окно) для ослабления пульсаций АЧХ, появляющихся из-за эффекта Гиббса. Это функции `fir1` и `fir2`.

```
b=fir1(n, Wn, 'ftype', window)
```

Здесь `n` – порядок рассчитываемого фильтра, `wn` и `'ftype'` определяют частоту (частоты) среза и тип синтезируемого фильтра. Частоты среза задаются нормированными к частоте Найквиста (то есть их значения должны лежать в диапазоне 0…1, величина 1 соответствует

половине частоты дискретизации). Значения параметров зависят от типа фильтра. К примеру для ФНЧ параметр 'ftype' отсутствует, в то время как для ФВЧ 'ftype' = 'high'.

Сюда же можно отнести функцию синтеза ФНЧ с косинусоидальным сглаживанием АЧХ - firrcos. Кроме того, функция kaiserord позволяет по заданным параметрам АЧХ оценить требуемый порядок фильтра при синтезе с использованием окна Кайзера.

Функции, реализующие минимизацию среднеквадратического отклонения АЧХ получающегося фильтра от заданной. Это функции firls, fircls и fircls1.

Функции, реализующие минимаксную оптимизацию, то есть минимизацию пикового отклонения АЧХ получающегося фильтра от заданной. К данной группе относятся функции remez (стандартный вариант метода Ремеза, в fdatool этот метод расчета называется Equiripple) и cremez (расширенный вариант, поддерживающий синтез фильтров с нелинейной ФЧХ и с комплексными коэффициентами).

В качестве примера синтезируем методом Ремеза нерекурсивный ФНЧ 32-го порядка с частотой среза 3 кГц, частотой дискретизации 12 кГц. Начало полосы задерживания зададим равным 3,5 кГц. После синтеза построим графики импульсной характеристики, а также АЧХ полученного фильтра (ФЧХ фильтра линейна).

```
>> Fs = 12000;                          % частота дискретизации
>> F0 = 3000;                           % частота среза
>> F1 = 3500;                           % начало полосы задерживания
>> b = remez(32, [0 F0/Fs*2 F1/Fs*2 1], [1 1 0 0]);
>> impz(b)                              % график импульсной характеристики
>> [h, f] = freqz(b, 1, [], Fs);        % комплексный коэффициент передачи
>> figure
>> plot(f, abs(h))                      % график АЧХ
>> grid
```

Графики импульсной характеристики и АЧХ представлены на рис.1.2

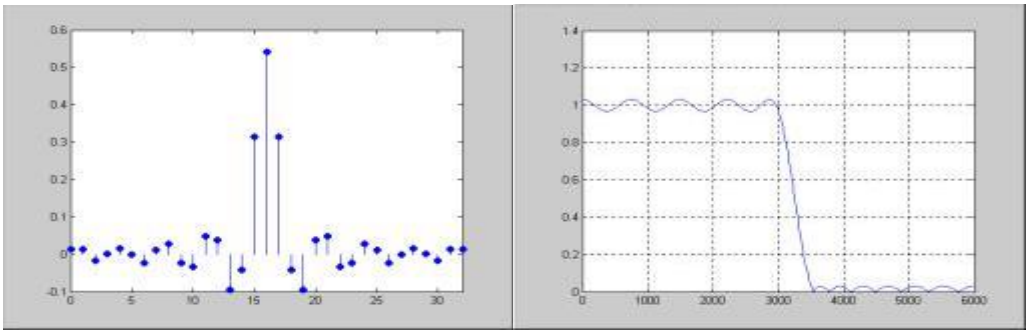


Рис. 1.2

К функциям прямого синтеза рекурсивных фильтров относятся следующие:

yulewalk - синтез рекурсивного фильтра с произвольной кусочно-линейной АЧХ методом Юла-Уолкера.

invfreqz - данная функция предназначена для решения задачи идентификации систем, она позволяет определить коэффициенты числителя и знаменателя функции передачи дискретной системы по набору значений этой функции передачи на различных частотах.

Программа синтеза и анализа фильтров — FDATool (Filter Design & Analysis Tool)

Мощные средства для интерактивного конструирования фильтров дает проектировщик – анализатор фильтров, вызываемый из командной строки командой

```
>> fdatool
```

Он открывает окно (рис. 1.3) с множеством полей и иных деталей интерфейса для выбора типа и его параметров, а также для выбора средств визуализации проектирования фильтров.

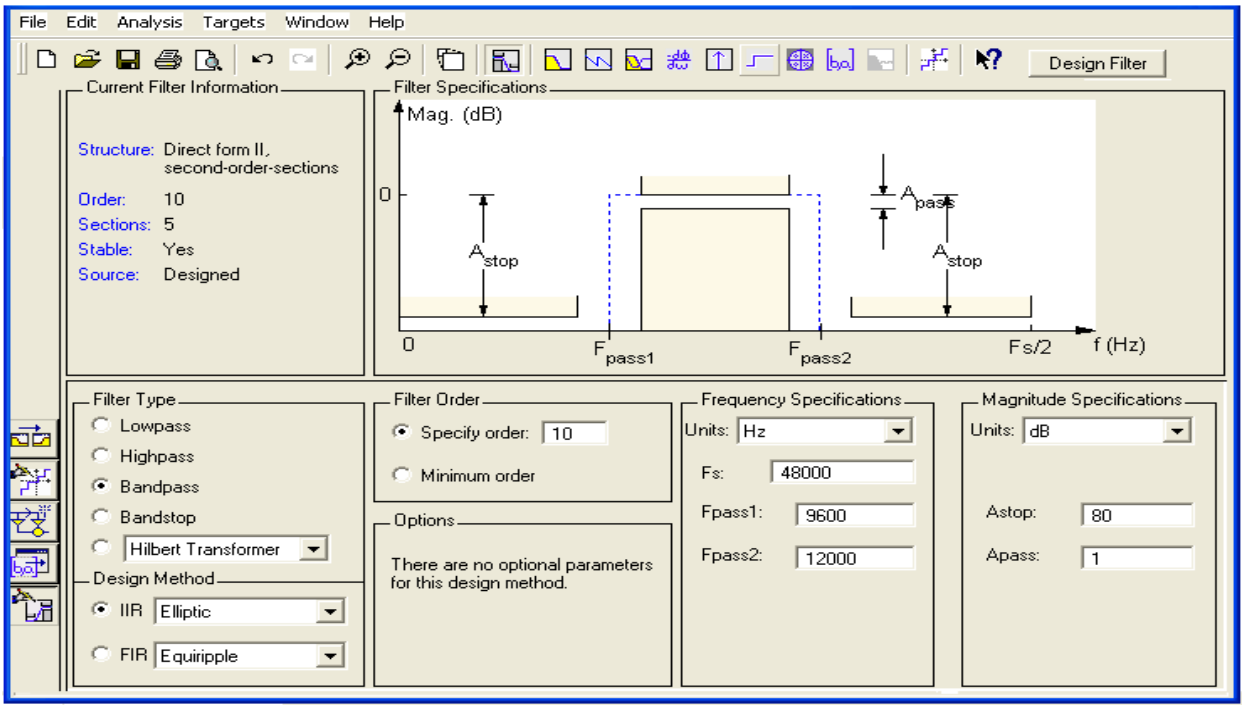


Рис. 1.3

На рис. 1.3 показан вид окна при выборе проектирования полосового фильтра. В окне просмотра представлены зоны допусков АЧХ с указанием их названий. После задания типа фильтра и его параметров достаточно нажать кнопку **Design Filter** для того, чтобы запустить программное конструирование фильтра под заданные параметры. После этого, используя кнопки панели инструментов, можно посмотреть и, если надо скорректировать полученные характеристики фильтра. Важным достоинством проектировщика фильтров является возможность реализации разработанных в нем фильтров в пакете Simulink. Спроектированный в программе fdatool фильтр может быть перенесен в графическую среду Simulink нажатием на кнопку **Realize Model**. На экране появится Simulink-модель, включающая один блок под именем Filter. Состав блока (в данном случае это структура разработанного фильтра) может быть выведен на экран двойным щелчком мыши. На рис. 1.4 представлено одно из звеньев фильтра, построенного согласно данным рис. 1.3.

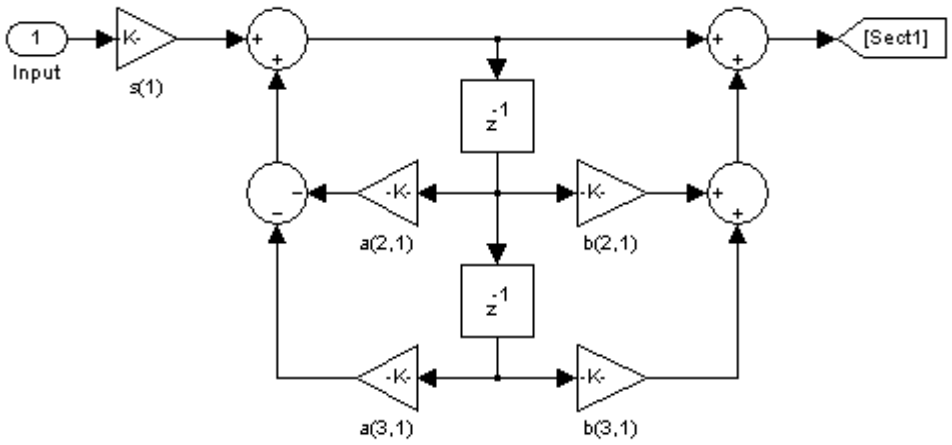


Рис. 1.4

Пример:

Синтезируем методом Ремеза нерекурсивный ФВЧ 32-го порядка с частотой среза 3,5 кГц, частотой дискретизации 12 кГц. Начало полосы задерживания зададим равным 3 кГц.

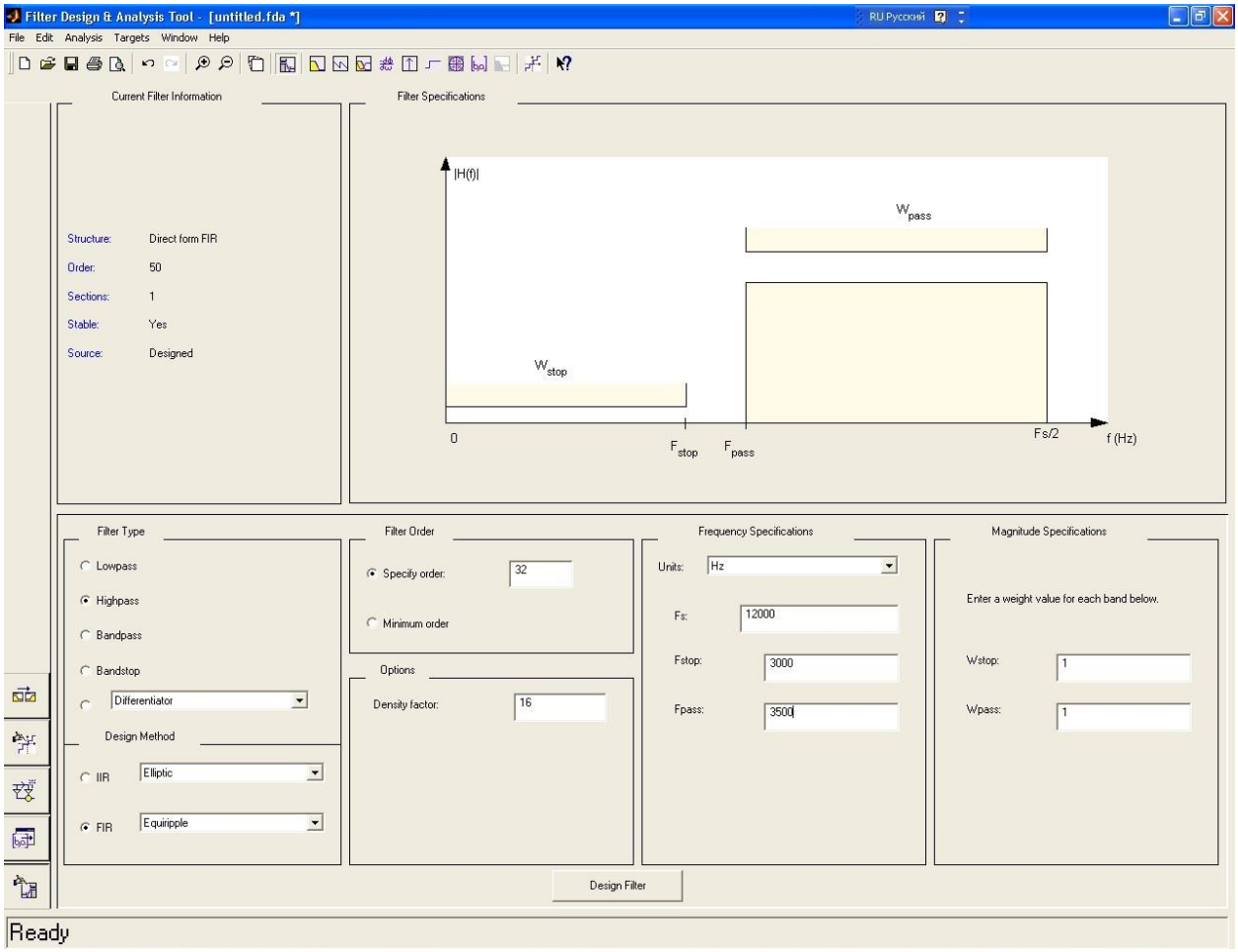


Рис. 1.5

В разделе Filter Type выберем тип фильтра, в нашем случае **highpass** – фильтр верхних частот (также в этом разделе представлены: фильтр нижних частот - **lowpass**, полосой фильтр - **bandpass** , режекторный фильтр – **bandstop** и специальные виды фильтров). В разделе **Design Method** выберем метод реализации фильтра: **IIR** - рекурсивный, **FIR** – нерекурсивный (в нашем примере это будет FIR –equiripple). В **Filter Order** зададим нужный порядок фильтра в графе **Specify order** (в примере зададим 32). При выборе пункта **Minimal order** программа сама автоматический подберет оптимальный порядок фильтра. В **Frequency specification** в графе **Units** выберем единицы измерения частоты (Гц, кГц, ГГц), в **Fs** зададим частоту дискретизации (12000 Гц), в **Fstop** – частоту задерживания (3000 Гц), в **Fpass** – частоту среза (3500 Гц). В разделе **Magnitude Specifications** в графе **Wstop** – зададим уровень ослабления в полосе задерживания, а в **Wpass** – неравномерность АЧХ в полосе пропускания.

3. Прядок выполнения работы

3.1 По заданию преподавателя выбрать исходные данные согласно табл. 1.1.
3.2 Синтезировать модель цифрового фильтра в системе MATLAB.
3.3 Построить АЧХ, ФЧХ, расположение нулей и полюсов передаточных функций, импульсную характеристику фильтра. Для этого использовать следующие функции: `freqz(b, a), zplane(b, a), impz(b, a).`
3.4 Синтезировать модель цифрового фильтра в программе FDATool, получить графики пункта 3.3.
3.5 Построить модель дискретного фильтра в пакете SimuLink.

4. Содержание отчета

4.1 Исходные данные моделирования.
4.2 Графики функций (согласно пункту 3.3, 3.4)
4.3 Коэффициенты рассчитанного фильтра.
4.4 Структурная схема звена спроектированного фильтра (SimuLink–модель).
4.5 Выводы по проделанной лабораторной работе и анализ полученных результатов.

5. Контрольные вопросы

5.1 Основные структуры цифровых фильтров. Рекурсивные и нерекурсивные фильтры.
5.2 Характеристики БИХ и КИХ фильтров.
5.3 Способы описания цифровых фильтров: Z-преобразование, передаточная функция, разностное уравнение.
5.4 Формы реализации дискретных фильтров. Соединения фильтров.
5.5 Устойчивость и реализуемость дискретных фильтров.
5.6 Основные методы проектирования цифровых фильтров.

Литература

1. Цифровая обработка сигналов / Л. М. Гольденберг и др. – М.: Радио и связь, 1990. – 256 с.
2. Одинец А. И., Гребенников А. И., Миронов С. Г. Цифровая обработка сигналов: Учеб. пособие. / Омск: Изд-во Наследие. Диалог-Сибирь, 2003. -64 с.
3. Цифровые фильтры в электросвязи и радиотехнике / А. В. Брунченко и др.; под ред. Л. М. Гольденберга. –М.: Радио и связь, 1982. –224 с.
4. Сергиенко А. Б. Цифровая обработка сигналов – СПб.: Питер, 2003. – 608 с.
5. Основы цифровой обработки сигналов: Курс лекций / А. И. Солонина и др. – СПб.: БХВ-Петербург, 2003. – 608 с.
6. Лазарев Ю. Ф. MatLAB 5.x – К.: Издательская группа BHV, 2000. – 384 с.

Таблица 1.1

Варианты заданий

№ вар.	1	2	3	4	5	6	7
	Рекур-сивный	Нерекур-сивный	Рекур-сивный	Нерекур-сивный	Рекур-сивный	Рекур-сивный	Рекур-сивный
	Эллипти-ческий	Обратное БПФ с оконной функцией Чебышева (R=60dB)	Баттерворта	Метод Ремеза	Чебышева I рода	Чебышева II рода	Баттерворта
Тип	ФВЧ	ФНЧ	Полосовой	ФВЧ	ФНЧ	ФНЧ	Режекторный
n	10	20	10	24	8	6	10
Rp	1	-	-		1	-	-
Rs	60	-	-		-	60	-
f0	4000	4500	300, 3400	1000	800	800	300, 3400
f1	-	-	-	1500	-	-	-
fs	12000	12000	8000	8000	8000	8000	8000

Лабораторная работа № 2

Изменение частоты дискретизации

1. Цель работы

Целью работы является изучение особенностей формирования сигналов с измененной частотой дискретизации.

2. Пояснения к работе

При решении различных задач обработки сигналов приходится увеличивать или уменьшать частоту дискретизации сигналов. Это необходимо, например, для согласования различных стандартов хранения и передачи дискретной информации. Классическим примером является преобразование аудиозаписей между форматами компакт-дисков (частота дискретизации 44,1 кГц) и цифровой записи R-DAT (частота дискретизации 48 кГц). В зависимости от значения этого коэффициента выделяют следующие варианты:

- *Интерполяция* – повышение частоты дискретизации в целое число раз;
- *Прореживание* (децимация) – понижение частоты дискретизации в целое число раз;
- *Передискретизация* – изменение частоты дискретизации в произвольное (в общем случае дробное) число раз.

Процедура понижения частоты дискретизации сигнала в N раз фактически означает, что из исходной последовательности берется каждый N-й отсчет, однако для предупреждения возможных побочных эффектов следует принять некоторые меры. Процесс прореживания, по сути, сводится к *дискретизации дискретного сигнала*. При этом имеют место все эффекты, при дискретизации сигналов. Если, например, в спектре исходного прореживаемого сигнала содержаться частоты, превышающие половину новой частоты дискретизации (то есть новую частоту Найквиста), это приведет к появлению в спектре выходного сигнала ложных частотных составляющих. Для устранения этого нежелательного эффекта следует, как и при дискретизации аналогового сигнала, предварительно пропустить сигнал через ФНЧ с частотой среза, равной новой частоте Найквиста. При этом чтобы сохранить фазовые соотношения во входном сигнале, следует использовать нерекурсивный фильтр с линейной ФЧХ.

В MATLAB прореживание выполняется с помощью функции `decimate`. Синтаксис ее вызова следующий:

```
y = decimate(x, r)
```

Здесь `x` – входной сигнал, `r` – целочисленный коэффициент понижения частоты дискретизации. Выходной параметр `y` – прореженный сигнал. Можно задать порядок фильтра, используя для этого третий числовой входной параметр. Вместо фильтра Чебышева возможно использование нерекурсивного фильтра n-го порядка.

В качестве предварительного фильтра по умолчанию используется ФНЧ Чебышева первого рода 8-го порядка с уровнем пульсаций в полосе пропускания 0,05 дБ и частотой среза, равной 0,8 новой (после прореживания) частоты Найквиста. Для устранения фазовых искажений применяется двунаправленная обработка входного сигнала.

Для демонстрации прореживания сформируем сигнал, мгновенная частота которого меняется по гармоническому закону от нуля до 2000 Гц и обратно в течение одной секунды:

```
>> fs = 10000;
>> t = 0:1/fs:1;
>> s = cos(2*pi*t*fs/10-fs/10*sin(2*pi*t));
```

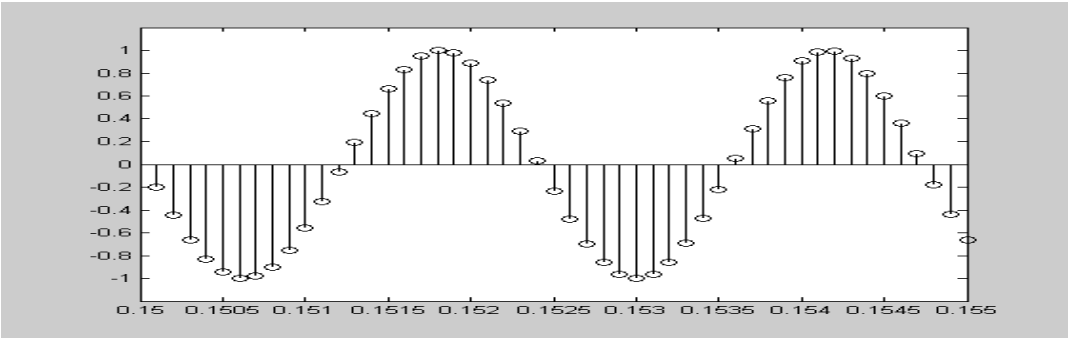


Рис. 2.1

Построим график сформированного сигнала (рис. 2.1):

```
>> stem(t, s);
```

Понизим частоту дискретизации сигнала в два раза путем выборки каждого второго отсчета и построим график полученной последовательности (рис. 2.2):

```
>> r = 2;
>> y = s(1:r:end);
>> stem((0:length(y)-1)*(1/(fs/r)), y);
```

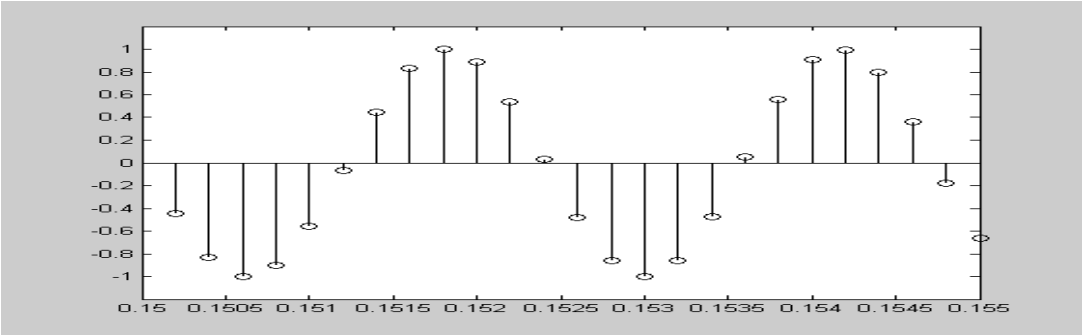


Рис. 2.2

Обратите внимание, что при использовании функции decimate автоматически осуществляется предварительная фильтрация сигнала перед прореживанием. В примере, приведенном выше, используется простое прореживание, поэтому, для того чтобы избежать наложения частот необходимо осуществлять предварительную фильтрацию вручную.

При интерполяции нам необходимо повысить частоту дискретизации в N раз, то есть «растянуть» входной сигнал, а образовавшиеся промежутки между отсчетами чем-то заполнить. Подобно тому, как прореживание сводится к дискретизации дискретного сигнала, процесс интерполяции оказывается подобным процессу восстановления непрерывного сигнала, только происходящему в дискретной области.

Исходный сигнал имеет периодический спектр, повторяющийся с частотой дискретизации $f_д$. Прежде всего мы растягиваем сигнал, добавляя между его отсчетами по N-1 нулей. При этом частота дискретизации сигнала станет равна $Nf_д$, но период повторения спектра останется прежним. - $f_д$. После этого необходимо пропустить полученный сигнал через ФНЧ с частотой среза $f_д/2$. В результате фильтрации получится интерполированный

сигнал, у которого частота дискретизации равна $Nf_д$, а спектр в полосе частот от нуля до $f_д/2$ остался прежним.

В MATLAB интерполяция выполняется с помощью функции `interp`. Синтаксис ее вызова следующий:

```
y = interp(x, r)
```

Здесь `x` – входной сигнал, `r` – целочисленный коэффициент повышения частоты дискретизации. Выходной параметр `y` – интерполированный сигнал. Данная функция содержит еще ряд параметров, однако они могут быть установлены по умолчанию.

```
>> r = 2;
>> y = interp(s, r);
>> stem((0:length(y)-1)*(1/(fs*r)), y);
```

График полученного сигнала приведен на рис. 2.3

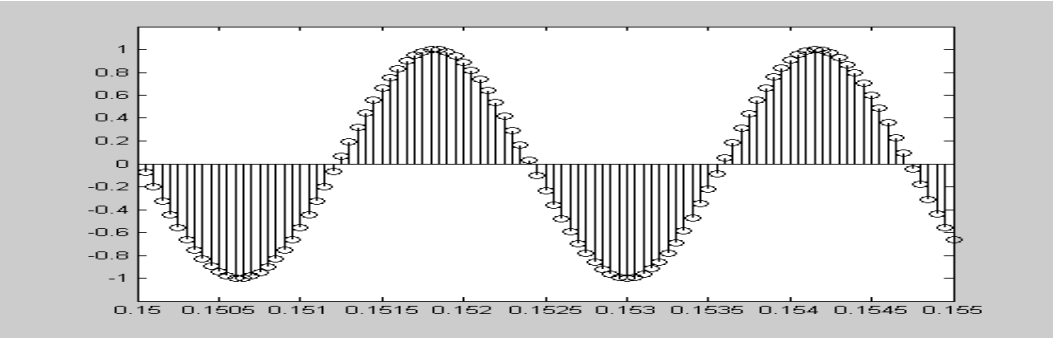


Рис. 2.3

Для выполнения передискретизации сигнала в MATLAB существует функция `resample`. Запись данной функции аналогична предыдущим.

```
y = resample (x, p, q)
```

Здесь `x` – входной сигнал, `p` и `q` – числитель и знаменатель дробного коэффициента изменения частоты дискретизации. Выходной параметр `y` – передискретизированный сигнал.

```
>> p = 3;
>> q = 4;
>> y = resample (s, p, q);
```

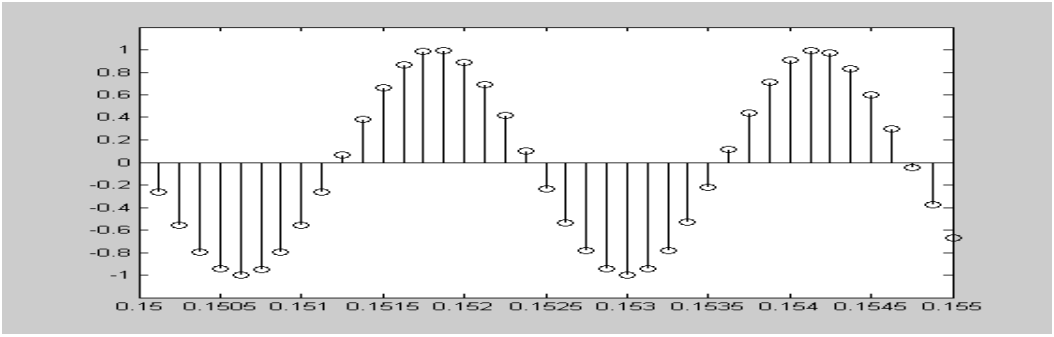


Рис. 2.4

Построим график полученного сигнала (рис. 2.4):

```
>> stem((0:length(y)-1)*(1/(fs*(p/q))), y);
```

3. Порядок выполнения работы

3.1. По заданию преподавателя, руководствуясь данными табл. 2.1 выбрать сигнал и параметры изменения частоты дискретизации.
3.2. Промоделировать сигнал в системе MATLAB.
3.3. Выполнить прореживание сигнала путем выборки каждого второго отсчета
```
>> y = s(1:r:end);
```
3.4. Построить спектрограммы исходного сигнала и сигнала после понижения частоты дискретизации. Для этого использовать команду
```
>> specgram(s, 512, fs, kaiser(256, 5), 220); (s – вектор сигнала).
```

3.5. Сделать вывод о появлении (или отсутствии) ложных частот в спектре сигнала с измененной частотой дискретизации. В случае появления ложных частот уменьшить коэффициент понижения частоты и повторить действия пунктов 3.3. При отсутствии ложных частот в спектре сигнала увеличить максимальную частоту сигнала так чтобы этот эффект проявился.
3.6. Выполнить пункты 3.3-3.5 используя для прореживания сигнала функцию
```
>>  y = decimate(x, r)  ;
```
3.7. Провести интерполяцию и передискретизацию сигнала.
3.8. Построить временные диаграммы и спектрограммы сигнала после интерполяции и передискретизации.

4. Содержание отчета

4.1 Исходные данные (номер варианта, тип сигнала и параметры изменения частоты дискретизации).
4.2 Графики функций (согласно порядку выполнения работы).
4.3 Выводы по проделанной лабораторной работе и анализ полученных результатов.

5. Контрольные вопросы

5.1 Аналоговые, дискретные и цифровые сигналы. Связь между аналоговыми и дискретными сигналами.
5.2 Аналого-цифровое и цифро-аналоговое преобразование.
5.3 Спектры аналоговых и дискретных сигналов, нормированные частоты.
5.4 Теорема Котельникова. Разложение сигналов в базис Котельникова. Восстановление непрерывного сигнала по набору дискретных отсчетов.
5.5 Свертка дискретных сигналов.
5.6 Общие сведения и основные понятия децимации, интерполяции и передискретизации сигналов.
5.7 Экспандер и компрессор частоты дискретизации.

Литература

1. Цифровая обработка сигналов / Л. М. Гольденберг и др. – М.: Радио и связь, 1990. – 256 с.
2. Одинец А. И., Гребенников А. И., Миронов С. Г. Цифровая обработка сигналов: Учеб. пособие. / Омск: Изд-во Наследие. Диалог-Сибирь, 2003. -64 с.
3. Сергиенко А. Б. Цифровая обработка сигналов – СПб.: Питер, 2003. – 608 с.
4. Основы цифровой обработки сигналов: Курс лекций / А. И. Солонина и др. – СПб.: БХВ-Петербург, 2003. – 608 с.
5. Лазарев Ю. Ф. MatLAB 5.x – К.: Издательская группа BHV, 2000. – 384 с.

Таблица 2.1

Варианты заданий

№ вар	Тип сигнала	Параметры изменения частоты дискретизации

1	Косинусоидальный сигнал с частотной модуляцией по пилообразному закону:
	`>> fs = 2000;`
	`>> t = 0:1/fs:2;`
	`>> s = vco(sawtooth(2*pi*t, 0.75), [0.1 0.4]*1000, fs);`
		`r = 2`
		`p = 3`
		`q = 4`

2	Сигнал с линейным законом изменения мгновенной частоты:
	`>> fs = 8000;`
	`>> t = 0:1/fs:2;`
	`>> f0 = 1000;`
	`>> t1 = 1`
	`>> f1 = 1500;`
	`>> s = chirp(t, f0, t1, f1, 'linear');`
		`r = 2`
		`p = 4`
		`q = 5`

3	Косинусоидальный сигнал с частотной модуляцией по закону функции Дирихле:
	`>> fs = 2000;`
	`>> t = 0:1/fs:2;`
	`>> s = vco(diric(2*pi*t, 4), [0.1 0.4]*1000, fs);`
		`r = 2`
		`p = 5`
		`q = 3`

4	Сигнал с квадратичным законом изменения мгновенной частоты:
	`>> fs = 8000;`
	`>> t = 0:1/fs:2;`
	`>> f0 = 1000;`
	`>> t1 = 1`
	`>> f1 = 1500;`
	`>> s = chirp(t, f0, t1, f1, 'quadratic');`
		`r = 2`
		`p = 3`
		`q = 5`

5	Косинусоидальный сигнал с частотной модуляцией по пилообразному закону:
	`>> fs = 8000;`
	`>> t = 0:1/fs:2;`
	`>> s = vco(sawtooth(2*pi*t, 0.75), [0.1 0.9]*1000, fs);`
		`r = 4`
		`p = 4`
		`q = 5`

6	Сигнал с экспоненциальным законом изменения мгновенной частоты:
	`>> fs = 8000;`
	`>> t = 0:1/fs:1;`
	`>> f0 = 1000;`
	`>> t1 = 1`
	`>> f1 = 1500;`
	`>> s = chirp(t, f0, t1, f1, 'logarithmic');`
		`r = 2`
		`p = 7`
		`q = 5`

7	Косинусоидальный сигнал с частотной модуляцией по закону функции Дирихле:
	`>> fs = 4000;`
	`>> t = 0:1/fs:2;`
	`>> s = vco(diric(2*pi*t, 3), [0.1 0.4]*1000, fs);`
		`r = 4`
		`p = 2`
		`q = 3`

Лабораторная работа № 3

Эффекты квантования в цифровых фильтрах

1. Цель работы

Целью работы является изучение влияния конечной точности вычислений на характеристики цифровых фильтров, а также ознакомление со способами учета конечной точности вычислений в системе MATLAB.

2. Пояснения к работе

Для создания объекта квантователя служит функция `quantizer`. У данного объекта немного свойств, причем числовым является только одно из них, а все строковые свойства имеют уникальные значения. В большинстве случаев требуется использовать полный вариант вызова, когда указываются имена и значения всех свойств:

```
q = quantizer('name1', value1, 'name2', value2, …)
```

Здесь 'name1', 'name2' и т. д. – имена свойств, а value1, value2 и т. д. – соответствующие значения. Список свойств квантованного фильтра приведен в табл. 3.1

Таблица 3.1

Имя свойства	Описание
Mode	Тип квантователя: 'double' – квантователь с плавающей запятой, стандартный 64-битовый формат double. Все остальные параметры в данном случае игнорируются; 'float' – квантователь с плавающей запятой; 'fixed' – квантователь с фиксированной запятой (вариант по умолчанию); 'single' – квантователь с плавающей запятой, стандартный 32-битовый формат single. Все остальные параметры в данном случае игнорируются; 'ufixed' – квантователь без знака с фиксированной запятой
RoundMode	Режим округления: 'ceil' – округление вверх; 'convergent' – округление к ближайшему целому значению, в случае равенства расстояний выбирается четное значение; 'fir' – округление к нулю; 'floor' – округление вниз (вариант по умолчанию); 'round' – округление к ближайшему целому значению, в случае равенства расстояний выбирается значение, большее по модулю
OverflowMode	Реакция на переполнение (для формата с фиксированной запятой): 'saturate' – при переполнении происходит насыщение (вариант по умолчанию); 'wrap' – при переполнении игнорируются «лишние» старшие разряды
Format	Двухэлементный вектор, первый элемент которого задает общее число двоичных разрядов, используемое для представления квантованных чисел. Второй элемент вектора для квантователей с фиксированной запятой указывает число разрядов дробной части, а для квантоавтелей с плавающей запятой – число разрядов, используемое для представления порядка. Значение по умолчанию равно [16 15]

Для примера создадим квантователь с фиксированной запятой, округлением вниз, переходом в режим насыщения при переполнении. Пусть при этом для целой части используется один разряд, а для дробной – 15:

```
>> q=quantizer('fixed','floor','saturate',[16 15])
```

Для создания объекта квантованного фильтра служит функция `qfilt` В простейшем случае она может вызываться без выходных параметров. Однако основным вариантом синтаксиса является следующий:

```
hq = qfilt('structure', {coef})
```

Здесь '`structure`' – строковый параметр, задающий форму реализации фильтра, а `{coef}` – массив ячеек, содержащий исходные (не квантованные) данные коэффициенты фильтра. Способ задания зависит от выбранной формы реализации фильтра, соответствующая информация сведена в табл. 3.2. Для задания свойств объекта (их список будет приведен ниже) можно указывать при вызове функции `qfilt` дополнительные параметры в виде пар «имя свойства – значение свойства»:

```
hq = qfilt('structure', {coef}, 'name1', value1, 'name2', value2, …)
```

Здесь 'name1', 'name2' и т. д. – имена свойств, а value1, value2 и т. д. – соответствующие значения. В табл. 3.2 приведен список некоторых форм реализации квантованных фильтров и способов задания их коэффициентов.

Таблица 3.2

Параметр 'structure'	Форма реализации фильтра	Способ задания коэффициентов
'df1'	Прямая форма	{coef}={b, a}, то есть параметр {coef}должен быть массивом ячеек, содержащим коэффициенты фильтра
'df1t'	Транспонированная форма	То же что для 'df1'
'df2'	Каноническая форма	То же что для 'df1'
'df2t'	Транспонированная форма	То же что для 'df1'
'fir'	Прямая форма нерекурсивного фильтра	{coef}={b},
'firt'	Транспонированная форма нерекурсивного фильтра	То же что для 'fir'

В качестве примера рассчитаем эллиптический фильтр ФНЧ пятого порядка, создадим для него объект квантованного фильтра и произведем с помощью этого фильтра обработку последовательности прямоугольных импульсов.

Напомним, что в общем случае функции расчета фильтров записываются в следующем виде:

`butter(n, w0, type)` – расчет фильтров Баттерворта;
`cheby1(n, Rp, w0, type)` расчет фильтров Чебышева первого рода;
`cheby2(n, Rs, w0, type)` – расчет фильтров Чебышева второго рода;
`ellip(n, Rp, Rs, w0, type)` – расчет эллиптических фильтров (фильтров Золотарева -Кауэра).

В представленных записях n – порядок рассчитываемого фильтра, w0 – частота среза, Rp – уровень пульсаций АЧХ в полосе пропускания (дБ), Rs – уровень пульсаций АЧХ в полосе задерживания (дБ), type – параметр, определяющий тип фильтра: для ФНЧ параметр type отсутствует, для ФВЧ type = 'high'.

Функция расчета нерекурсивного фильтра с линейной ФЧХ:

```
b=fir1(n, Wn, 'ftype', window)
```

Здесь n – порядок рассчитываемого фильтра, wn и 'ftype' определяют тип синтезируемого фильтра и его частоту (частоты) среза. Частоты среза задаются нормированными к частоте Найквиста (в диапазоне 0…1, величина 1 соответствует половине частоты дискретизации). Значения параметров зависят от типа фильтра. К примеру для ФНЧ параметр 'ftype' отсутствует, в то время как для ФВЧ 'ftype' = 'high'.

Пусть ФНЧ имеет частоту среза, равную 20% от частоты Найквиста, пульсации АЧХ в полосе пропускания 1дБ и ослабление в полосе задерживания 40дБ:

```
>> [b,a]=ellip(5, 1, 40, 0.2)
```

В командном окне автоматически выводятся коэффициенты передаточной функции фильтра.

```
b =  0.0153  -0.0221   0.0154   0.0154  -0.0221   0.0153

a =  1.0000  -3.9043   6.5819  -5.8940   2.7935  -0.5599
```

Создадим для данного фильтра квантованный фильтр с использованием квантователя, рассчитанного в предыдущем примере. В данном случае ряд коэффициентов фильтра по модулю превышают единицу. Чтобы применить ранее созданный квантователь коэффициенты фильтра необходимо привести к диапазону [-1, 1]. Поэтому при создании объекта квантованного фильтра мы разделим векторы b и a на 8:

```
>>  hq = qfilt('df2t',{b/8, a/8},'quantizer', q)

hq =
Quantized Direct form II transposed filter
Numerator
     QuantizedCoefficients{1}    ReferenceCoefficients{1}
  (1)       0.001892089843750   0.001915044555875759
  (2)      -0.002777099609375  -0.002760232429165185
  (3)       0.001922607421875   0.001923857337327012
  (4)       0.001922607421875   0.001923857337327022
  (5)      -0.002777099609375  -0.002760232429165193
  (6)       0.001892089843750   0.001915044555875762
Denominator
     QuantizedCoefficients{2}    ReferenceCoefficients{2}
  (1)       0.125000000000000   0.125000000000000000
  (2)      -0.488067626953125  -0.488037284420594630
  (3)       0.822723388671875   0.822741734035389300
  (4)      -0.736755371093750  -0.736744248400267980
  (5)       0.349182128906250   0.349188852654597430
  (6)      -0.070007324218750  -0.069991714941048944
   FilterStructure = df2t
```

```
       ScaleValues = []
  NumberOfSections = 1
  StatesPerSection = [5]
 CoefficientFormat = quantizer('fixed', 'floor', 'saturate', [16  15])
       InputFormat = quantizer('fixed', 'floor', 'saturate', [16  15])
      OutputFormat = quantizer('fixed', 'floor', 'saturate', [16  15])
MultiplicandFormat = quantizer('fixed', 'floor', 'saturate', [16  15])
     ProductFormat = quantizer('fixed', 'floor', 'saturate', [16  15])
         SumFormat = quantizer('fixed', 'floor', 'saturate', [16  15])
```

Теперь построим графики АЧХ и ФЧХ фильтров с помощью функции `freqz` (рис. 3.1):

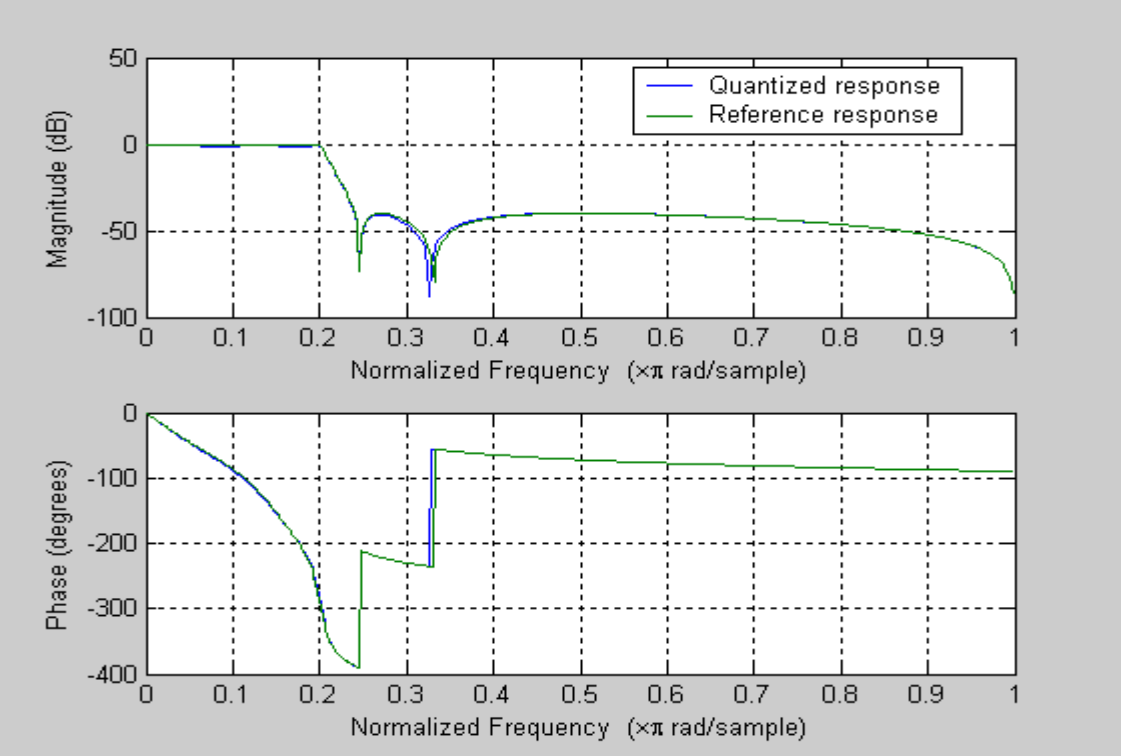


Рис. 3.1

На графиках можно видеть, что благодаря выполненному масштабированию коэффициентов и довольно высокой (15 двоичных разрядов после запятой) точности их квантованного представления, разница характеристик практически не заметна.

Квантованное быстрое преобразование Фурье

Для создания объекта квантованного быстрого преобразования Фурье (БПФ) служит функция `qfft`. В простейшем случае она может вызываться без выходных параметров, но поскольку обычно требуется выполнять БПФ заданной размерности, удобнее всего задавать размерность БПФ сразу же при создании объекта. Для этого нужно использовать в качестве параметров функции строку ‘length’ и значение размерности:

```
f = qfft('length', N)
```

Здесь `N` – размерность БПФ. Для задания свойств объекта (их список будет приведен ниже) можно указывать при вызове функции `qfft` дополнительные параметры в виде пар «имя свойства – значение свойства»:

```
f=qfft('name1', value1, 'name2', value2, …)
```

Здесь ' name1', 'name2' и т. д. – имена свойств, а value1, value2 и т. д. – соответствующие значения.

Вычисление квантованного БПФ осуществляется с помощью функций fft (прямое БПФ) и ifft (обратное БПФ). Отличие от обычного варианта использования этих функций состоит в том, что в начало списка параметров добавляется объект квантованного БПФ:

```
y = fft(f, x)
x = fft(f, y)
```

Здесь f – объект квантованного БПФ, x – сигнал во временной области, y – сигнал в частотной области. Если размерность неквантованного БПФ в MATLAB подстраивается под длину сигнала, то в случае квантованного БПФ все наоборот – сигнал дополняется нулями или усекается, чтобы его длина стала равна размерности БПФ. Список свойств квантованного БПФ приведен в табл. 3.3.

Таблица 3.3

Имя свойства	Описание
CoefficientFormat	Объект quantizer, задающий формат квантования коэффициентов, используемых при вычислении БПФ
InputFormat	Объект quantizer, задающий формат квантования входного сигнала
Length	Размерность БПФ
NumberOfSections	Число ступеней БПФ, равное log2(Length)/Log2(Radix)
MultiplicandFormat	Объект quantizer, задающий формат квантования чисел, умножаемых на коэффициенты БПФ
OutputFormat	Объект quantizer, задающий формат квантования выходного сигнала
ProductFormat	Объект quantizer, задающий формат квантования результатов операций умножения
Radix	Основание БПФ, которое может быть равно 2 (по умолчанию) или 4
ScaleValues	Коэффициенты масштабирования сигналов.
SumFormat	Объект quantizer, задающий формат квантования результатов операций сложения

В качестве примера создадим объект квантованного БПФ размерности 256 и затем вычислим спектр синусоидального сигнала с помощью обычной функции fft и с помощью квантованного объекта. При создании объекта необходимо задать только размерность БПФ, для всех остальных свойств могут быть оставлены значения по умолчанию.

```
  >> f = qfft('length', 256)
f =
             Radix = 2
            Length = 256
 CoefficientFormat = quantizer('fixed', 'round', 'saturate', [16  15])
       InputFormat = quantizer('fixed', 'floor', 'saturate', [16  15])
      OutputFormat = quantizer('fixed', 'floor', 'saturate', [16  15])
MultiplicandFormat = quantizer('fixed', 'floor', 'saturate', [16  15])
```

```
    ProductFormat = quantizer('fixed', 'floor', 'saturate', [32  30])
        SumFormat = quantizer('fixed', 'floor', 'saturate', [32  30])
NumberOfSections = 8
     ScaleValues = [1]
```

Пусть сигнал имеет такую частоту, что анализируемый фрагмент (256 отсчетов) содержит нецелое число периодов:

```
>> t = 0:255;                % дискретное время
>> s = 1/32*cos(2*pi*t/7); % период сигнала равен 7
```

Теперь рассчитаем два варианта спектра: с помощью обычного (вектор `sp`) и квантованного (вектор `sp_q`) БПФ и выведем графики их модулей (рис. 3.2, 3.3):

```
>> sp = fft(s);       % обычное БПФ
>> sp_q = fft(f, s); % квантованное БПФ
>> plot(t, abs(sp));
>> figure;
>> plot(t, abs(sp_q));
```

Сравнение графиков на рис. 3.2 и 3.3 показывает, что переполнения весьма существенно исказили результаты расчета – в спектре появились ложные составляющие. Это свидетельствует о необходимости либо уменьшить уровень входного сигнала (в данном случае, чтобы избавиться от переполнений, необходимо уменьшить амплитуду гармонического сигнала с 1/32 до 1/128), либо использовать в объекте `f` квантователи с большей верхней границей диапазона представимых чисел.

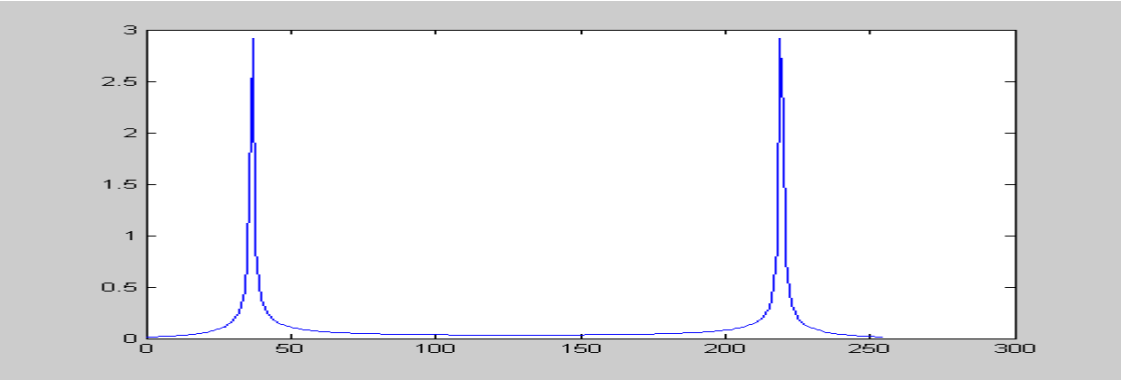


Рис. 3.2

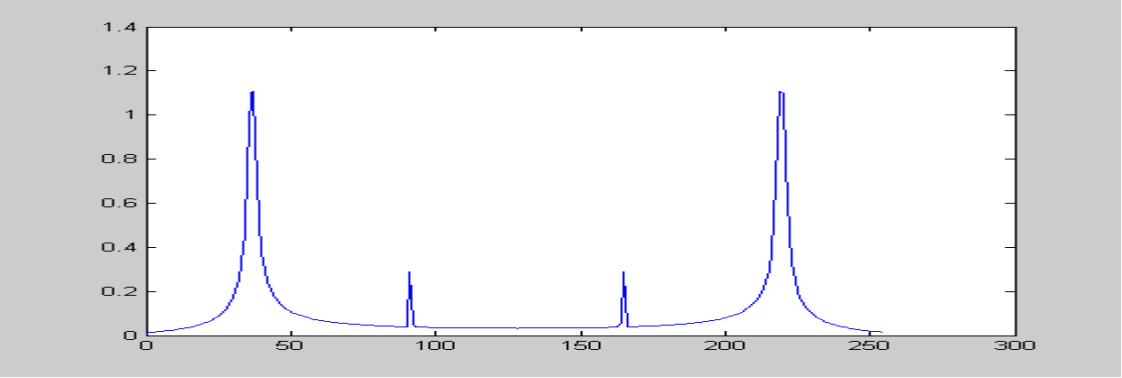


Рис. 3.3

3. Порядок выполнения работы

3.1 По заданию преподавателя и руководствуясь данными табл. 3.1 выбрать тип квантователя и ввести его в систему MATLAB.
3.2 Рассчитать неквантованный фильтр и создать соответствующий ему квантованный фильтр.
3.3 Построить АЧХ, ФЧХ, расположение нулей и полюсов передаточных функций, импульсные характеристики фильтров, определить порядок квантованного фильтра. Для этого использовать следующие функции: `freqz(hq)`, `zplane(hq)`, `impz(hq)`, `order(hq)`.
3.4 Получить отображение коэффициентов квантованного фильтра в виде двоичных строк. Для этого использовать функцию `num2bin( )`.
3.5 Используя данные табл. 3.2 выбрать тип сигнала и рассчитать неквантованное и квантованное БПФ. Построить графики Фурье-изображений.

4. Содержание отчета

4.1 Исходные данные моделирования.
4.2 Графики функций (согласно пункту 3.3, 3.5).
4.3 Коэффициенты рассчитанных фильтров в формате с фиксированной запятой и двоичном формате.
4.4 Выводы по проделанной лабораторной работе и анализ полученных результатов.

5. Контрольные вопросы

5.1 Форматы представления чисел.
5.2 Процесс квантования, шум квантования, неравномерное квантование. Связь динамического диапазона с параметрами квантования.
5.3 Квантование коэффициентов цифровых фильтров.
5.4 Переполнение разрядной сетки в процессе вычислений и округление результатов арифметических операций.
5.5 Масштабирование коэффициентов цифрового фильтра. Предельные циклы.
5.6 Погрешности цифровых систем. Источники погрешностей в цифровых системах.

Литература

1. Цифровая обработка сигналов / Л. М. Гольденберг и др. – М.: Радио и связь, 1990. – 256 с.
2. Одинец А. И., Гребенников А. И., Миронов С. Г. Цифровая обработка сигналов: Учеб. пособие. / Омск: Изд-во Наследие. Диалог-Сибирь, 2003. -64 с.
3. Цифровые фильтры в электросвязи и радиотехнике / А. В. Брунченко и др.; под ред. Л. М. Гольденберга. –М.: Радио и связь, 1982. –224 с.
4. Сергиенко А. Б. Цифровая обработка сигналов – СПб.: Питер, 2003. – 608 с.
5. Основы цифровой обработки сигналов: Курс лекций / А. И. Солонина и др. – СПб.: БХВ-Петербург, 2003. – 608 с.
6. Алгоритмы и процессоры цифровой обработки сигналов / А. И. Солонина и др. – СПб.: БХВ-Петербург, 2002, - 464 с.
7. Лазарев Ю. Ф. MatLAB 5.x – К.: Издательская группа BHV, 2000. – 384 с.

Таблица 3.1

Варианты заданий

№ вар.	1	2	3	4	5	6
Характеристики квантователя						
Mode	'fixed'	'fixed'	'fixed'	'fixed'	'fixed'	'fixed'
RoundMode	'floor'	'floor'	'floor'	'floor'	'floor'	'floor'
OverflowMode	'saturate'	'saturate'	'saturate'	'saturate'	'saturate'	'saturate'
Format	[16 15]	[16 15]	[10 9]	[16 15]	[16 15]	[16 15]
Характеристики фильтра						
Функция расчета фильтра	fir1	fir1	ellip	butter	cheby2	cheby2
Порядок фильтра	55	55	5	10	8	10
Тип фильтра	ФНЧ 'type' – отсутств.	ФВЧ 'type' = 'high'	ФВЧ 'type' = 'high'	ФНЧ 'type' – отсутств.	ФВЧ 'type' = 'high'	ФВЧ 'type' = 'high'
Частота среза	0.5	0.5	0.2	0.4	0.2	0.6
Тип окна	kaiser	kaiser	отсутств.	отсутств.	отсутств.	отсутств.
Rp, дБ	отсутств.	отсутств.	1	отсутств.	отсутств.	отсутств.
Rs, дБ	отсутств.	отсутств.	40	отсутств.	60	40

Таблица3.2

№ Вар.	Сигнал
1	`s = 1/32*cos(2*pi*t/10)+1/16*sin(2*pi*t/7)`
2	`s = 1/32*cos(2*pi*t/7)+1/128*sin(2*pi*t/5)`
3	`s = -1/32*cos(2*pi*t/13)+1/8*sin(2*pi*t/11)`
4	`s = 1/4*cos(2*pi*t/15)-1/128*sin(2*pi*t/19)`
5	`s = 1/32*cos(2*pi*t/7)+1/128*cos(2*pi*t/3)`
6	`s = 1/128*sin(2*pi*t/11)-1/32*sin(2*pi*t/7)`

Лабораторная работа № 4

Дискретное преобразование Фурье

1. Цель работы

Целью работы является исследование спектральных характеристик дискретных сигналов, а также изучение способов вычисления спектральных характеристик сигналов с использованием системы MATLAB

2. Пояснения к работе

В основе спектрального анализа лежит теория Фурье о возможности разложения любого периодического процесса с периодом $T = 2\pi/\omega = 1/f$ (где ω – круговая частота периодического процесса, а f – его частота в герцах) в бесконечную, но счетную сумму отдельных гармонических составляющих.

Любой процесс с периодом T может быть представлен в виде комплексного ряда Фурье. Совокупность комплексных амплитуд ряда может рассматриваться как изображение периодического процесса в частотной области. Наибольшее распространение получила следующая форма записи преобразования Фурье:

$$X(f) = \int_{-\infty}^{+\infty} x(t) \cdot e^{-j\cdot(2\cdot\pi\cdot f)\cdot t}\, dt. \qquad (1)$$

Данный интеграл (1) существует (является сходящимся к конечной величине) только для так называемых «двусторонне затухающих» процессов (т. е. таких, которые уменьшаются до нуля как при $t \to +\infty$, так и при $t \to -\infty$). Иначе говоря, его нельзя применять к так называемым «стационарным» колебаниям.

Указанное противоречие, касающееся условий существования интеграла (1) несколько сглаживается при численных расчетах, так как в этом случае можно иметь дело только с процессами ограниченной длительности, причем сам процесс в заданном диапазоне времени должен быть задан своими значениями в ограниченном числе точек.

В этом случае интегрирование заменяется суммированием и вместо вычисления интеграла (.1) расчеты ограничиваются вычислением суммы:

$$X[(k-1)\cdot\Delta f] = \sum_{m=1}^{n} x[(m-1)\cdot\Delta t]\cdot e^{-j\cdot 2\cdot\pi\cdot(k-1)\cdot(m-1)\cdot\Delta f\cdot\Delta t}. \qquad (2)$$

Здесь Δt - дискрет времени, m – номер точки от начала процесса, k – номер значения частоты, $\Delta f = 1/T$ дискрет частоты, T – промежуток времени на котором задан процесс.
Если обозначить дискрет времени Δt через Ts, ввести обозначения:

$$x(m) = x[(m\text{-}1)\cdot\Delta t]; \quad X(k) = X[(\text{k-}1)\cdot\Delta f],$$

а также учесть то, что число точек в которых задан процесс, равно:

$$n = T/\Delta t = T/Ts = 1/(\Delta f\cdot\Delta t), \qquad (3)$$

то соотношение (2) можно представить в более удобной форме:

$$x(k) = Ts\sum_{m=1}^{n} x(m)\cdot e^{-j\cdot 2\cdot\pi\cdot(m-1)\cdot(k-1)\,/\,\mathrm{n}}. \qquad (4)$$

Данное выражение носит название дискретного преобразования Фурье (ДПФ). Формально ДПФ периодической последовательности и ДПФ конечной последовательности полностью совпадают и разница между ними заключается в *трактовке результатов*, а именно:

Коэффициенты ДПФ *периодической* последовательности – это ее дискретный спектр, а коэффициенты ДПФ *конечной* последовательности – это дискретные отсчеты ее

непрерывного спектра на периоде, по которым гарантируется возможность точного восстановления непрерывного спектра. К примеру, ДПФ конечной последовательности $x(m) = a^m$ длины n равно:

$$x(k) = \frac{1-a^n}{1-a \cdot e^{-j\frac{2\cdot\pi}{n}k}}, k = 0,1,..n-1.$$

Точно такое же решение будет получено для ДПФ периодической последовательности $x(m) = a^m$ с периодом равным n.

Последовательность $x(m)$, вычисленная по формуле ОДПФ в действительности является не конечной, а *периодической* с периодом, равным ее длине n, поэтому полученный результат интерпретируется как *один период* последовательности $x(m)$, полагая, что за его границами $x(m) \equiv 0$.

В MATLAB прямое и обратное преобразования Фурье осуществляются с помощью функций `fft` и `ifft`. Вычисления осуществляются в соответствии с формулами:

$$y(k) = \sum_{m=1}^{n} x(m) \cdot e^{-j\cdot 2\cdot\pi\cdot(m-1)\cdot(k-1)/n}. \quad (5)$$

$$x(m) = \frac{1}{n}\sum_{k=1}^{n} y(k) \cdot e^{j\cdot 2\cdot\pi\cdot(m-1)\cdot(k-1)/n}. \quad (6)$$

Сравнивая (4) с (5), можно сделать вывод, что процедура `fft` находит дискретное Фурье-изображение заданного дискретного во времени процесса *x(t)*, поделенное на дискрет времени:

$$y(k) = X(k) / Ts \quad (7)$$

Из этого следует, что комплексный спектр разложения стационарного процесса равен поделенному на число измерений результату применения процедуры `fft` к заданному вектору измеренного процесса. Если же принять во внимание, что для большинства стационарных колебательных процессов амплитудный и фазовый спектры не зависят от длительности T конкретной реализации и выбранного дискрета времени Ts, то надо также сделать вывод, что для спектрального анализа стационарных процессов наиболее целесообразно применять процедуру `fft`, результат которой делить затем на число точек измерений.

Для применения процедуры `fft` необходимо сформировать частоты следующим образом: $f = 0 : df : Fmax$, $Fmax = 1/Ts$, $df = 1/T$. Однако процедура `fft` не дает непосредственно Фурье-изображения процесса. Чтобы получить Фурье-изображение, необходимо к полученному результату применить процедуру `fftshift` и перестроить вектор частот по алгоритму: $f = -Fmax/2 : 1/T : Fmax/2$.

Функция `fft` имеет следующий синтаксис вызова:

```
y = fft(x, N)
```

Здесь `x` – сигнал во временной области, N – длина вектора исходных данных. `y` – сигнал в частотной области.

В качестве примера рассчитаем ДПФ синусоидального сигнала, представленного 16 отсчетами с частотой дискретизации равной 1 и периодом, равным 4 отсчетам.

```
>> td = 0:15;              % формирование массива времени
>> T = 4;                  % период сигнала
>> x1d = cos(2*pi*td/T);   % Формирование сигнала
```

Построим график сформированного сигнала.

```
>> stem(td, x1d);          % график сигнала
```

На рисунке 4.1 для наглядности отсчеты сигнала соединены пунктирной линией.

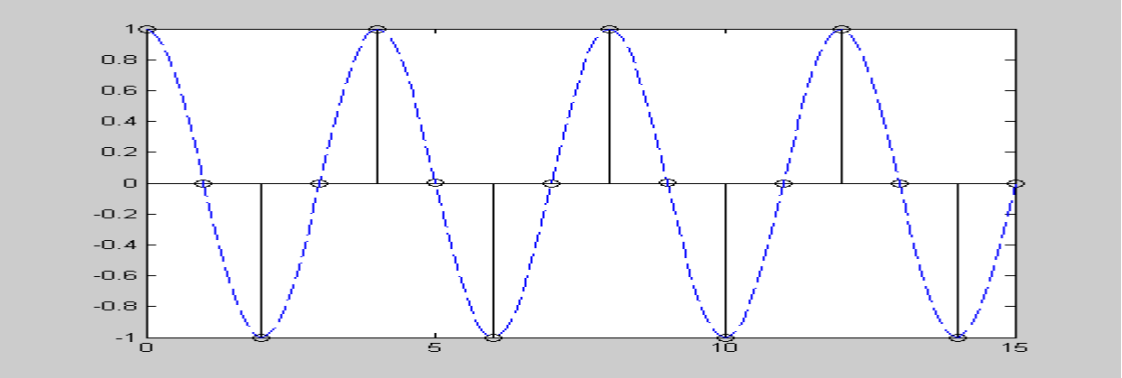


Рис. 4.1

```
>> y = fft(x1d);          % ДПФ
>> stem(td, abs(y1)); % график ДПФ
```

Результат выполнения команд представлен на рис. 4.2

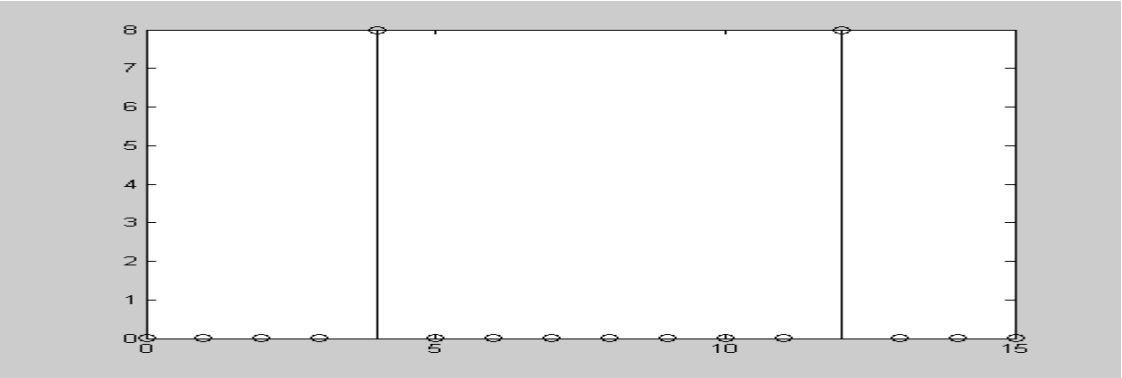


Рис. 4.2

Спектр одиночного фрагмента дискретной синусоиды является периодической непрерывной функцией частоты и имеет лепестковую структуру. Если построить непрерывный амплитудный спектр фрагмента, то он будет иметь вид, показанный на рис. 4.3.

```
>> hold on
>> [h, w] = freqz(x1d, 1, [], 16, 'whole');
>> plot(w, abs(h))
```

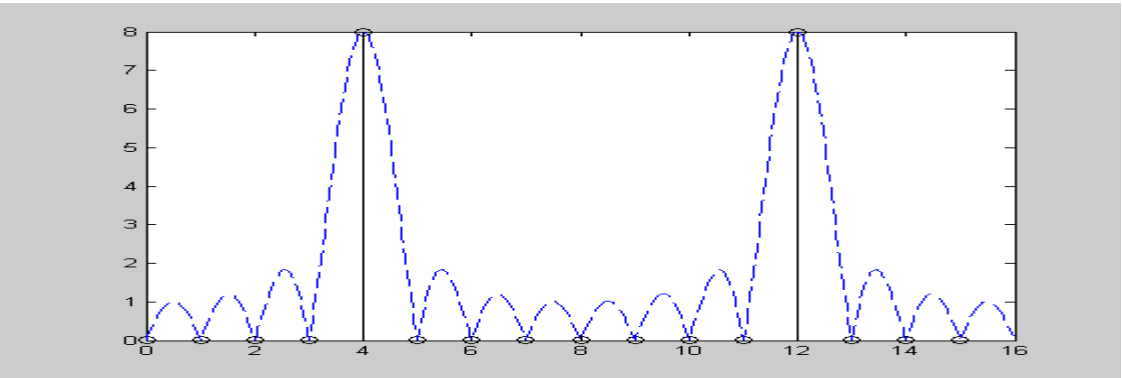


Рис. 4.3

Сформируем сигнал, являющийся периодическим продолжением рассмотренного выше фрагмента.

```
>> tdm = [[td-16, td, td+16]]; % формирование массива времени
>> x1dm = [x1d, x1d, x1d];      % продолженный сигнал
>> stem(tdm, x1dm);
```

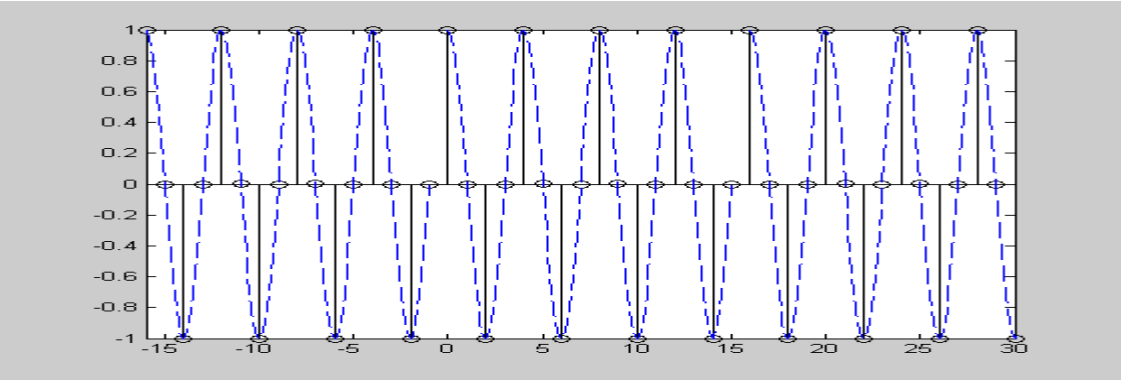


Рис. 4.4

Вычислим БПФ периодически продолженного сигнала.

```
>> y1 = fft(x1dm);
>> t = 0:47;
```

Построим график БПФ (рис. 4.5):

```
>> stem(t, abs(y1));
```

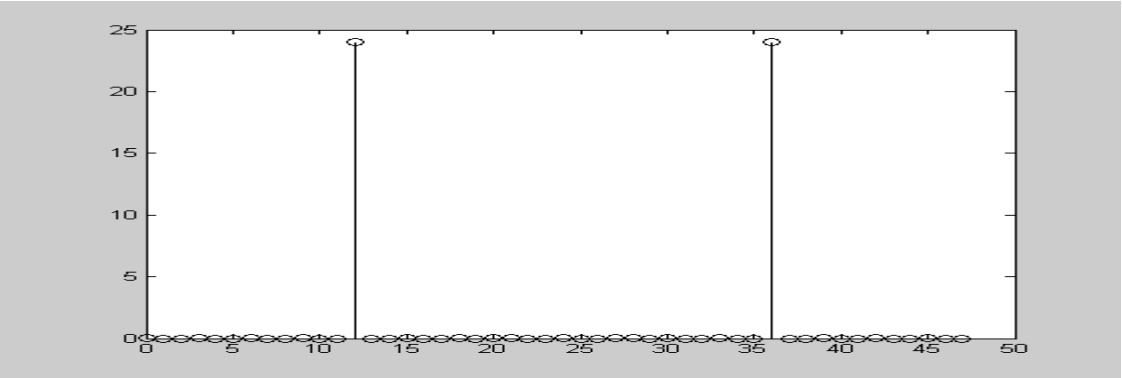


Рис. 4.5

Для того, чтобы перейти к реальным амплитудам и частотам выполняем следующие действия:

```
>> Fmax = 1;
>> df = 1/length(x1dm);
>> f = -Fmax/2:df:Fmax/2;
>> stem(f(1:length(x1dm)), abs(fftshift(y1)));
```

Результат выполнения последних четырех команд представлен на рис. 4.6.

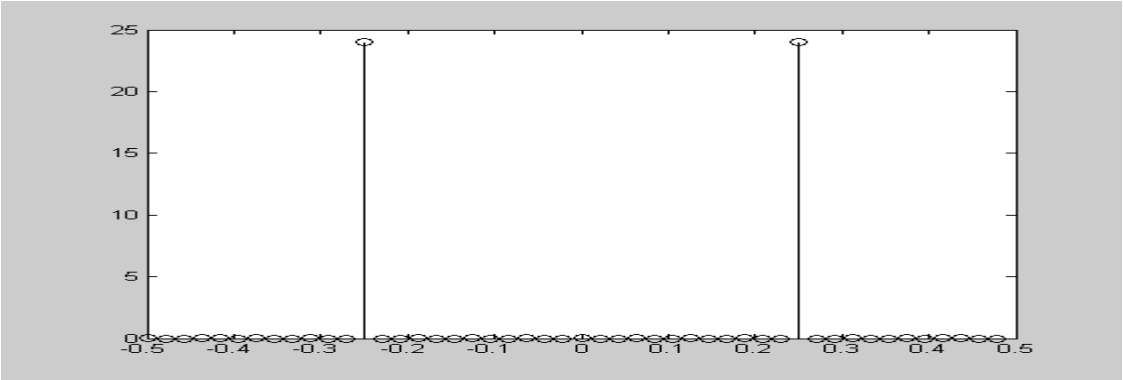


Рис. 4.6

Полученный результат умножаем на 2 и делим на число измерений:

```
>> stem(f(1:length(x1dm)), 2*abs(fftshift(y1))/48, 'k');
```

Результат последней операции показан на рис. 4.7.

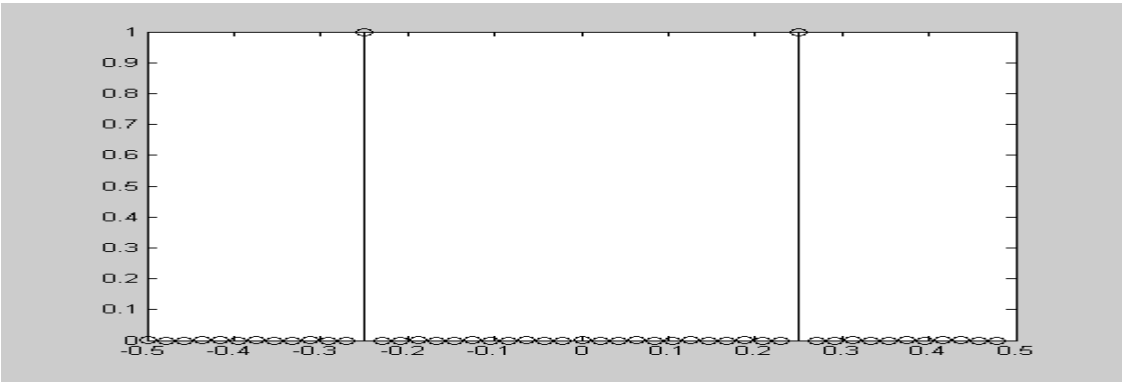


Рис. 4.7

Как можно видеть из рисунка, полученное Фурье-изображение сигнала полностью совпадает с теоретическим.

3. Порядок выполнения работы

3.1 По заданию преподавателя, пользуясь данными табл. 4.1 выбрать тип сигнала и ввести его в систему MATLAB.
3.2 Построить временную диаграмму сигнала.
3.3 Используя функцию `fft` вычислить ДПФ сигнала.
3.4 Построить график Фурье-изображения сигнала.
3.5 Ввести в систему MATLAB сигнал, представляющий собой периодическое продолжение во времени сигнала из табл 4.1.
3.6 Повторить действия п. 3.2 – 3.4 с сигналом, полученным при выполнении п. 3.5.

4. Содержание отчета

4.4 Временные диаграммы сигналов.
4.5 Графики Фурье-изображений сигналов.
4.6 Выводы по проделанной лабораторной работе и анализ полученных результатов.

5. Контрольные вопросы

5.1 Спектр дискретного сигнала.
5.2 Ряд Фурье. Разложение сигналов в ряд Фурье.
5.3 Интегральное преобразование Фурье.
5.4 Дискретное и быстрое преобразование Фурье.
5.5 Дискретное преобразование Фурье конечной и периодической последовательностей.
5.6 Свойства преобразования Фурье.

Литература

1. Цифровая обработка сигналов / Л. М. Гольденберг и др. – М.: Радио и связь, 1990. – 256 с.
2. Одинец А. И., Гребенников А. И., Миронов С. Г. Цифровая обработка сигналов: Учеб. пособие. / Омск: Изд-во Наследие. Диалог-Сибирь, 2003. -64 с.
3. Баскаков С. И. Радиотехнические цепи и сигналы. –М.: Высш. шк., 2000. – 462 с.
4. Сергиенко А. Б. Цифровая обработка сигналов – СПб.: Питер, 2003. – 608 с.
5. Основы цифровой обработки сигналов: Курс лекций / А. И. Солонина и др. – СПб.: БХВ-Петербург, 2003. – 608 с.
6. Алгоритмы и процессоры цифровой обработки сигналов / А. И. Солонина и др. – СПб.: БХВ-Петербург, 2002, - 464 с.
7. Лазарев Ю. Ф. MatLAB 5.x – К.: Издательская группа BHV, 2000. – 384 с.

Таблица 4.1

Варианты заданий

№ вар.	

1	Несимметричный прямоугольный импульс
	`>> td = 0:15;`
	`>> x1d = 0.1*td;`

2	Симметричный трапециевидный импульс
	`>> Fs = 1000;`
	`>> td = 0:1/Fs:0.1;`
	`>> T1 = 50*1/Fs;`
	`>> T2 = 25*1/Fs;`
	`>> s = 2*(T2*tripuls(td-25*1/Fs, T2)-T1*tripuls(td-25*1/Fs, T1))/(T2-T1);`

3	Симметричный прямоугольный импульс
	`>> Fs = 1000;`
	`>> td = 0:1/Fs:0.1;`
	`>> s = 2.5*tripuls(td-30*1/Fs, 60*1/Fs, 0);`

4	Односторонний экспоненциальный импульс
	`>> Fs = 1000;`
	`>> td = 0:1/Fs:0.1;`
	`>> s = exp(-90*td);`

5	Двусторонний экспоненциальный импульс
	`>> Fs = 1000;`
	`>> td = 0:1/Fs:0.1;`
	`>> s = exp(-90*abs(td-50*1/Fs));`

6	Односторонний экспоненциальный импульс
	`>> Fs = 1000;`
	`>> td = 0:1/Fs:0.1;`
	`>> s = exp(-200*td);`

7	Несимметричный прямоугольный импульс
	`>> td = 0:15;`
	`>> x1d = 0.1*td;`

Лабораторная работа № 5

Статистический анализ случайных процессов

1. Цель работы

Целью работы является изучение статистических характеристик сигналов, а также ознакомление с основными функциями анализа случайных процессов в системе MATLAB.

2. Пояснения к работе

К задачам статистического анализа процессов относится определение некоторых статистических характеристик, а именно: корреляционных характеристик, спектральных плотностей мощности и т. п.

Для оценки корреляционной матрицы случайного процесса по вектору его отсчетов с использованием временного усреднения служит функция `corrmtx`, имеющая следующий синтаксис:

```
[X, R] = corrmtx(x, m)
```

Здесь x – вектор отсчетов сигнала, m – размерность рассчитываемой корреляционной матрицы. Данная функция имеет еще несколько параметров, однако они не являются обязательными. Понятие «корреляционная» в MATLAB используется, разумеется, в зарубежной трактовке. Это означает, что функция `corrmtx` *не вычитает* из сигнала постоянную составляющую перед вычислением корреляционной матрицы.

В качестве примера рассчитаем корреляционную матрицу стационарного случайного процесса, сформированного путем пропускания белого Гауссова шума через фильтр:

```
>> Ts = 0.01; T = 100;        % Задание параметров процесса
>> t = 0:Ts:T;                % формирование массива времени
>> x1 = randn(1,length(t)); % формирование Гауссова шума
>> plot(t, x1)
```

График сформированного процесса показан на рис. 5.1.

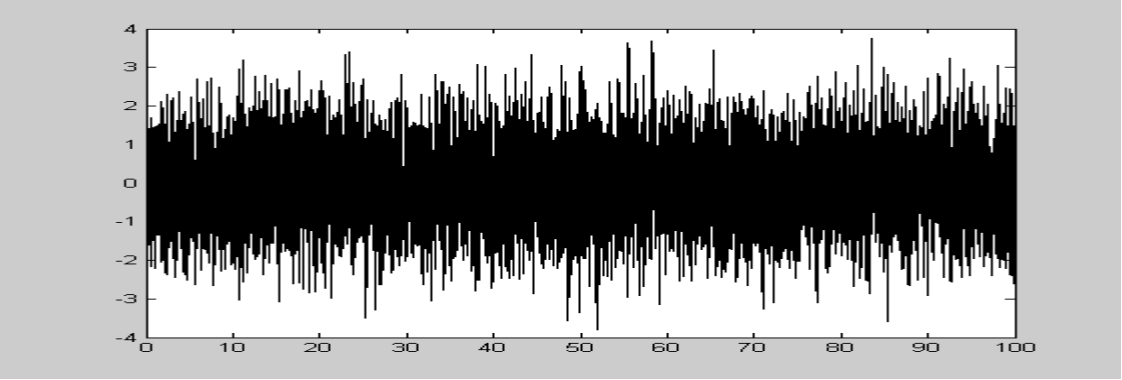


Рис. 5.1

Расчет параметров формирующего фильтра

```
>> om0 = 2*pi; dz = 0.05; A = 1; oms = om0*Ts;
>> a(1) = 1+2*dz*oms+oms^2;
>> a(2) = -2*(1+dz*oms);
```

```
>> a(3) = 1;
>> b(1) = A*2*dz*oms^2;
>> y1 = filter(b,a,x1); % формирование «профильтрованного» процесса
>> plot(t, y1);
```

Процесс, прошедший через формирующий фильтр представлен на рис. 5.2.

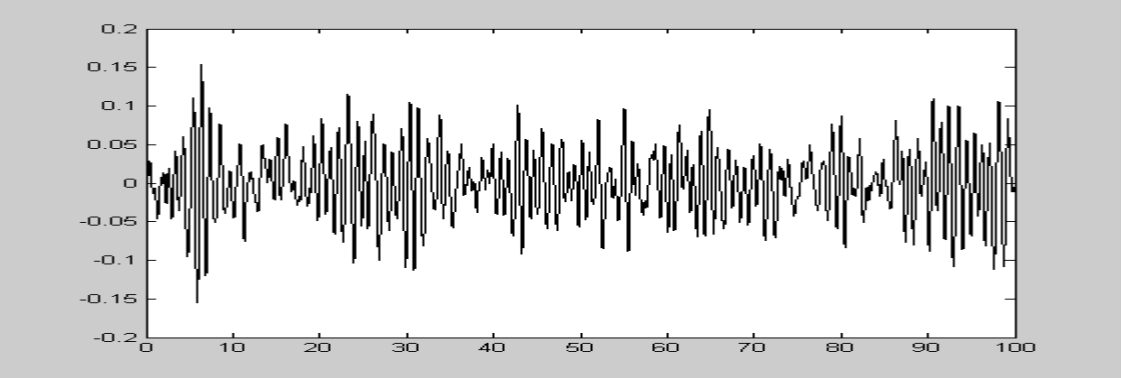


Рис. 5.2

Рассчитаем корреляционную матрицу и построим графики корреляционных функций (КФ) отсчетов белого Гауссова шума и процесса на выходе формирующего фильтра:

```
>> N = 20;                 % размер корреляционной матрицы
>> [tmp, R0] = corrmtx(x1, N); [tmp, R] = corrmtx(y1, N);
>> k = 0:N;
>> stem(k, R0(1,:)) % построение КФ Гауссова шума
>> stem(k, R(1,:))  % построение КФ «профильтрованного» процесса
```

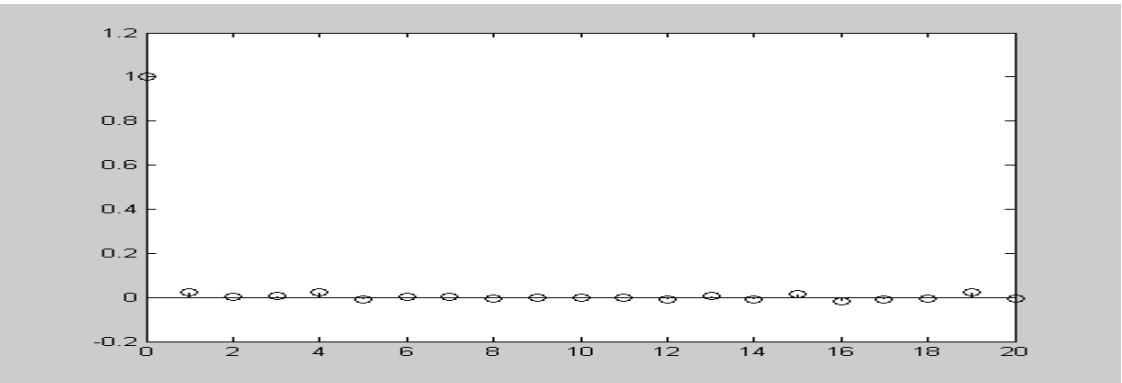


Рис. 5.3

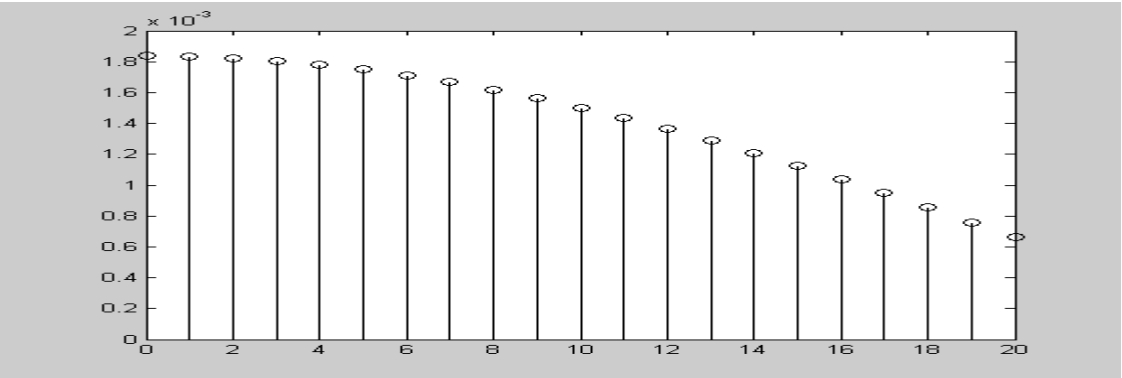


Рис. 5.4

Результаты выполнения команд представлены на рис. 5.3 и 5.4.

Рассчитаем спектр случайного процесса

```
>> df = 1/T; Fmax = 1/Ts;
>> f = - Fmax/2 : df : Fmax/2;% Формирование массива частот
>> dovg = length(f);
```

Рассчитаем ФИ процессов

```
>> Fu1 = fft(x1)/dovg; Fu2 = fft(y1)/dovg;
>> Fu1p = fftshift(Fu1); Fu2p = fftshift(Fu2);
>> A1 = abs(Fu1p); A2 = abs(Fu2p); % формирование модулей ФИ
```

Произведем расчет и построим графики спектральных плотностей мощности белого Гауссова шума и «профильтрованного» процесса:

```
>> S1 = Fu1p.*conj(Fu1p)*dovg;
>> S2 = Fu2p.*conj(Fu2p)*dovg;
>> stem(f, S1)
>> stem(f, S2)
```

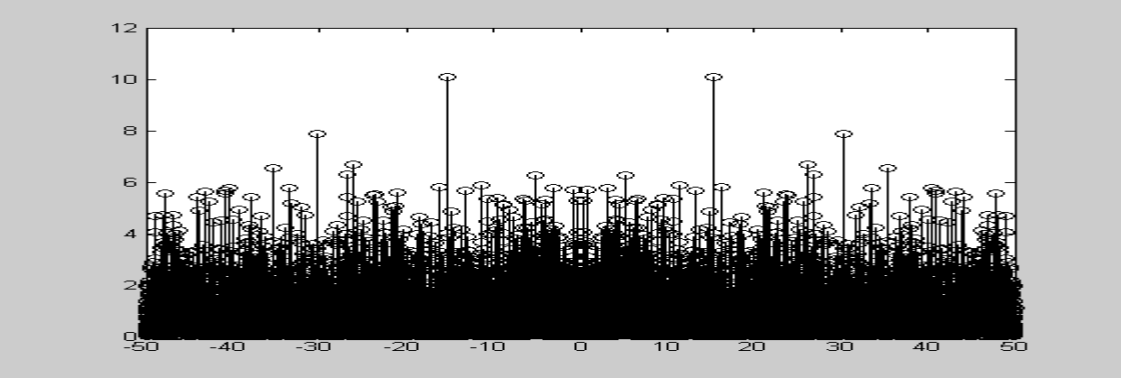


Рис. 5.5

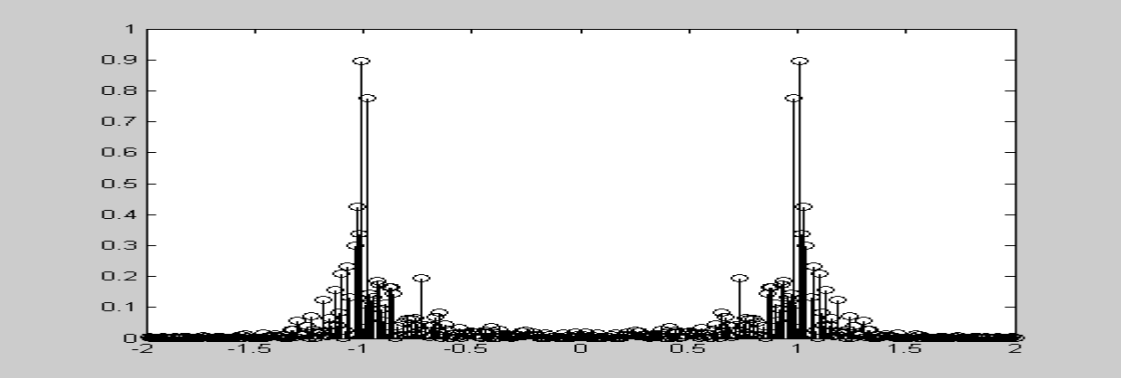


Рис. 5.6

Результаты расчета представлены на рис. 5.5, 5.6.

Программа обработки сигналов — SPTool (Signal Processing Tool)

Программа обработки сигналов SPTool (Signal Processing Tool) позволяет импортировать сигналы из MAT-файлов или рабочей области MATLAB, применять к сигналам различные методы спектрального анализа и рассчитывать дискретные фильтры с использованием функций пакета.

Следует отметить, что в части анализа и синтеза фильтров возможности программы SPTool являются более узкими, чем у программы FDATool, рассмотренной в лабораторной работе № 1 (единственным исключением является отсутствующая в FDATool возможность прямого редактирования расположения нулей и полюсов). Однако эти ограничения компенсируются возможностью экспорта рассчитанного фильтра из FDATool в SPTool.

Чтобы обрабатывать какие-либо сигналы с помощью SPTool, прежде всего необходимо сформировать эти сигналы в MATLAB, а затем импортировать полученные векторы значений этих сигналов в среду SPTool.

Допустим, мы сгенерировали случайные процессы x1 и y1 в соответствии с программой, приведенной выше. В результате в рабочем пространстве MATLAB появились векторы x1 и y1, каждый из которых содержит по 10001 элементу. Импортируем их в среду SPTool.

Войдя в среду SPTool, выберем из меню **File** команду **Import**. После этого откроется окно **Import to SPTool**, представленное на рис. 5.7.

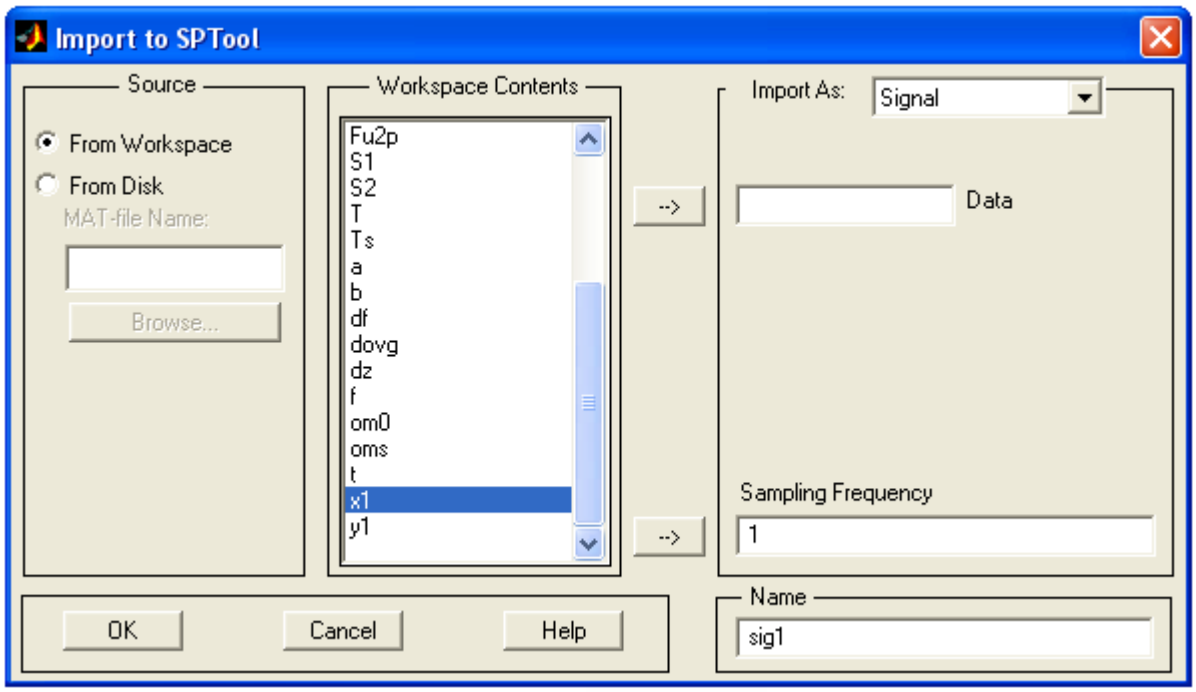


Рис. 5.7

В области **Source** этого окна необходимо выбрать переключатель **From Workspace**. Во второй области **Workspace Contents** будут видны все имена переменных рабочего пространства. Выбрав при помощи мыши необходимую переменную, следует нажать кнопку со стрелкой, указывающей на поле ввода **Data**. После этого в поле ввода должно появиться имя выбранной переменной.

Затем в поле **Sampling Frequency** нужно ввести желаемое значение частоты дискретизации. В поле **Name** необходимо указать имя, под которым введенный вектор будет записан в программе SPTool.

После этого следует нажать кнопку **OK**, и импорт сигнала в программу SPTool будет произведен. Окно **Import to SPTool** исчезнет, а в области **Signals** окна программы SPTool появится запись с имени вектора сигнала. Кнопка **View** под этой областью станет доступной.

Кроме того, станет доступной кнопка **Create** под областью **Spectra**. Это означает, что можно находить спектральные характеристики импортированного сигнала.

На рис. 5.8 показан вид окна программы SPTool при просмотре графика сигнала.

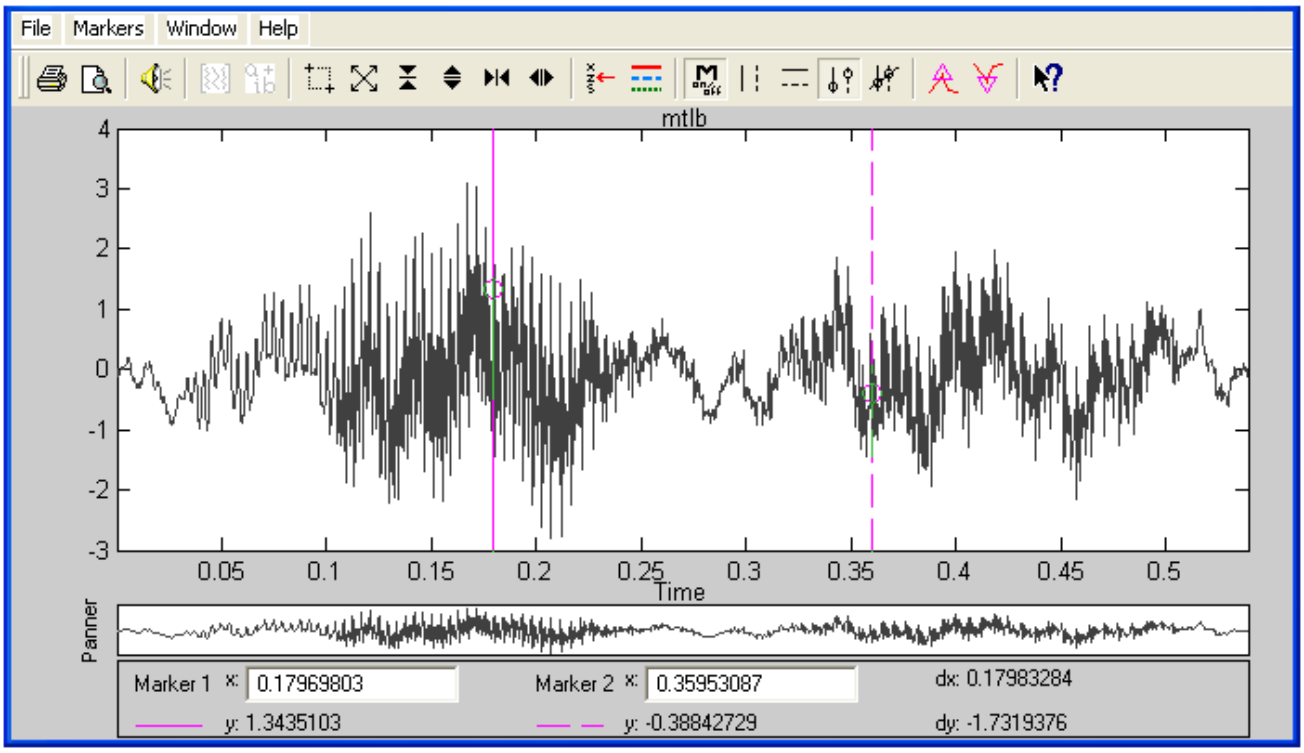


Рис. 5.8

После ввода сигналов в SPTool можно найти оценки спектральных свойств этих сигналов. Для этого достаточно в области сигналов окна SPTool отметить сигнал, оценку которого мы хотим получить и нажать на кнопку **Create**. После этого на экране появится окно **Spectrum Viewer** (рис. 5.9). Основным параметром данного окна является метод вычисления спектра сигналов.

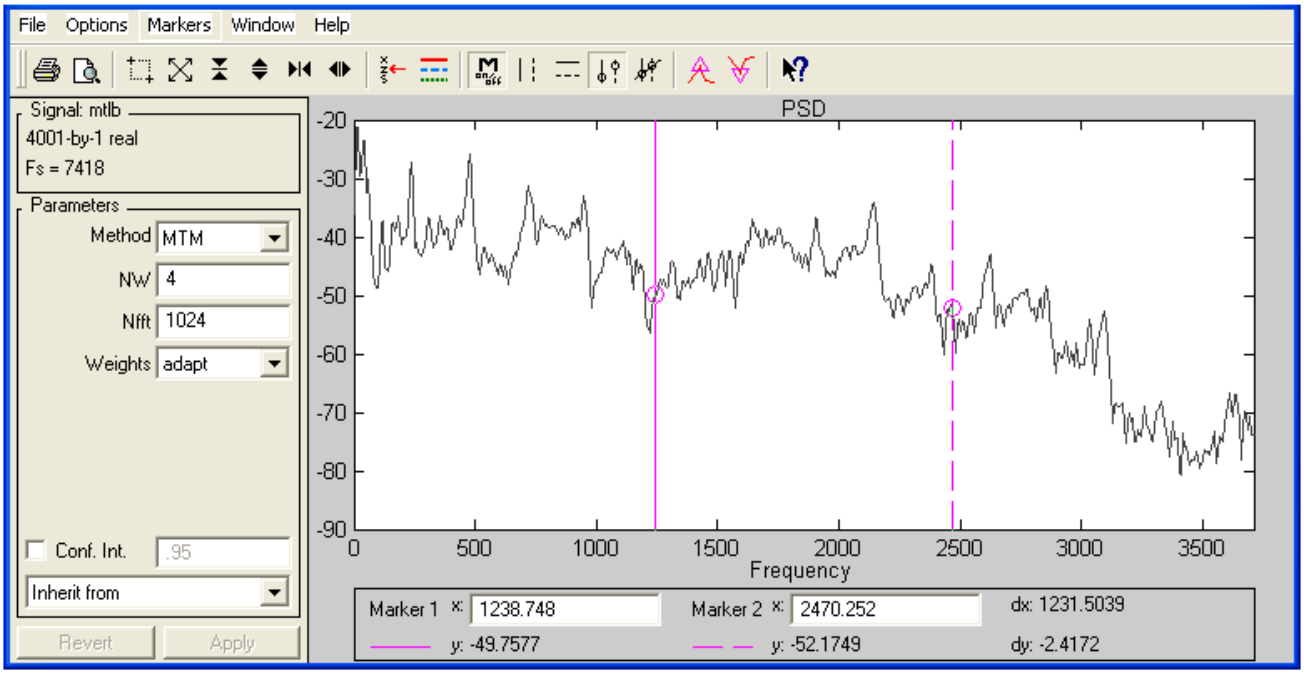


Рис. 5.9

3. Порядок выполнения работы

3.1 Используя функцию генерации случайных чисел `randn` промоделировать реализацию белого Гауссова шума с дисперсией, равной единице, частотой дискретизации 100 Гц длительностью 100 с.
3.2 Пользуясь данными табл. 5.1 построить формирующий фильтр.
3.3 Пропустить реализацию белого шума через построенный фильтр.
3.4 Построить гистограммы реализации белого шума и выходного сигнала формирующего фильтра. Для этого использовать функцию `hist(x, 15)`
3.5 Определить оценки математического ожидания, среднеквадратического отклонения, дисперсии сигналов. Для этого использовать функции `mean(x), std(x), var(x).`
3.6 Вычислить ДПФ сигналов, построить графики ФИ.
3.7 Рассчитать спектральные плотности мощности сигналов, построить графики.
3.8 Произвести расчет спектров сигналов в программе SPTool.
3.9 Сравнить результаты, полученные при выполнении п. 6 и 8.

4. Содержание отчета

4.1 Осциллограммы сигналов.
4.2 Графики функций (согласно пунктам 4, 6, 7).
4.3 Численные значения оценок математического ожидания, среднеквадратического отклонения, дисперсии сигналов.
4.4 Выводы по проделанной лабораторной работе и анализ полученных результатов.

5. Контрольные вопросы

5.1 Вероятностные характеристики случайных процессов.
5.2 Корреляционные функции случайных процессов. Некоррелированность и статистическая независимость.
5.3 Спектральные характеристики случайных процессов. Белый шум.
5.4 Взаимосвязь между корреляционными функциями и спектрами сигналов.

Литература

1. Цифровая обработка сигналов / Л. М. Гольденберг и др. – М.: Радио и связь, 1990. – 256 с.
2. Одинец А. И., Гребенников А. И., Миронов С. Г. Цифровая обработка сигналов: Учеб. пособие. / Омск: Изд-во Наследие. Диалог-Сибирь, 2003. -64 с.
3. Баскаков С. И. Радиотехнические цепи и сигналы. –М.: Высш. шк., 2000. – 462 с.
4. Сергиенко А. Б. Цифровая обработка сигналов – СПб.: Питер, 2003. – 608 с.
5. Основы цифровой обработки сигналов: Курс лекций / А. И. Солонина и др. – СПб.: БХВ-Петербург, 2003. – 608 с.
6. Тихонов В. И. Статистическая радиотехника. – М.: Радио и связь, 1982. – 624 с.
7. Лазарев Ю. Ф. MatLAB 5.x – К.: Издательская группа BHV, 2000. – 384 с.

Варианты заданий

Таблица 5.1.

№ вар.	Расчет формирующего фильтра
1	>> om0 = 2*pi*8; dz = 0.05; A = 1; >> oms = om0*Ts; >> a(1) = 1+2*dz*oms+oms^2; >> a(2) = -2*(1+dz*oms); >> a(3) = 1; >> b(1) = A*2*dz*oms^2;
2	>> alpha = 0.9; >> sigma = 2; >> a(1) = 1; >> a(2) = -alpha; >> b(1) = sigma*sqrt(1-alpha^2);
3	>> alpha = 0.3; >> sigma = 1; >> a(1) = 1; >> a(2) = -alpha; >> b(1) = sigma*sqrt(1-alpha^2);
4	>> om0 = 2*pi*10; dz = 0.05; A = 1; >> oms = om0*Ts; >> a(1) = 1+2*dz*oms+oms^2; >> a(2) = -2*(1+dz*oms); >> a(3) = 1; >> b(1) = A*2*dz*oms^2;
5	>> alpha = 0.8; >> sigma = 1; >> a(1) = 1; >> a(2) = -alpha; >> b(1) = sigma*sqrt(1-alpha^2);
6	>> om0 = 2*pi*5; dz = 0.05; A = 1; >> oms = om0*Ts; >> a(1) = 1+2*dz*oms+oms^2; >> a(2) = -2*(1+dz*oms); >> a(3) = 1; >> b(1) = A*2*dz*oms^2;
7	>> om0 = 2*pi*3; dz = 0.05; A = 1; >> oms = om0*Ts; >> a(1) = 1+2*dz*oms+oms^2; >> a(2) = -2*(1+dz*oms); >> a(3) = 1; >> b(1) = A*2*dz*oms^2;

Лабораторная работа № 6

Изучение работы адаптивного цифрового фильтра

1. Цель работы

Целью работы является изучение принципов адаптивной цифровой фильтрации случайных сигналов.

2. Пояснения к работе

Для решения задачи синтеза оптимального цифрового фильтра с постоянными (не изменяемыми) значениями импульсной характеристики необходима полная информация о входных воздействиях (полезном сигнале и помехах). Эта информация может быть использована для создания системы фильтрации, обеспечивающей экстремальное значение соответствующего критерия оптимальности. Однако во многих случаях цифровые фильтры с постоянными параметрами не могут быть использованы, так как статистические свойства сигналов изменяются во времени. Поэтому необходимо сначала обучать цифровые фильтры по обучающим статистикам, а затем осуществлять слежение за ними, если они медленно меняются.

Если частотные характеристики цифровых фильтров зависят от спектров обрабатываемых сигналов, то такие фильтры называют адаптивными [1–2]. Импульсная характеристика адаптивного цифрового фильтра изменяется в процессе работы согласно определенному алгоритму. Под алгоритмом адаптивной цифровой фильтрации понимается рекуррентная процедура пересчета вектора значений импульсной характеристики на предыдущем шаге в вектор "новых" значений импульсной характеристики для следующего шага. Требования к АЧХ адаптивных фильтров обычно не задаются, поскольку их характеристики изменяются во времени.

Под адаптивным понимается такой алгоритм принятия решения, при построении которого для преодоления априорной неопределенности используется предварительное обучение [1]. Цифровому фильтру предъявляются «примеры», на которых он «обучается», т. е. на его вход подаются обучающие сигналы и отслеживаются значения выходного сигнала (соответствуют ли они требуемым). Обрабатывая обучающие сигналы, фильтр подстраивает свои характеристики в соответствии с выбранным критерием оптимальности. Наиболее часто используются критерии минимума среднеквадратической ошибки (СКО), максимума правдоподобия и др. [3].

После обучения на ограниченном количестве примеров адаптивный фильтр должен быть способен «правильно» обрабатывать другие входные воздействия (отличающиеся от обучающих сигналов). Таким образом, функционирование адаптивного фильтра улучшается в результате взаимодействия с окружающей средой.

Во многих случаях априорно устойчивые адаптивные алгоритмы очень сложны в реализации (например, включают процедуры обращения матриц). Поэтому для упрощения реализации, а также для уменьшения необходимой производительности вычислителя используют различные рекуррентные методы

Одним из вариантов построения алгоритмов функционирования адаптивных фильтров является использование теории *нечетких* (fuzzy) множеств [3]. Модель адаптивного цифрового фильтра на основе теории нечетких множеств представлена на рис. 6.1-6.2 [4]. Для создания модели была использована графическая среда Simulink.

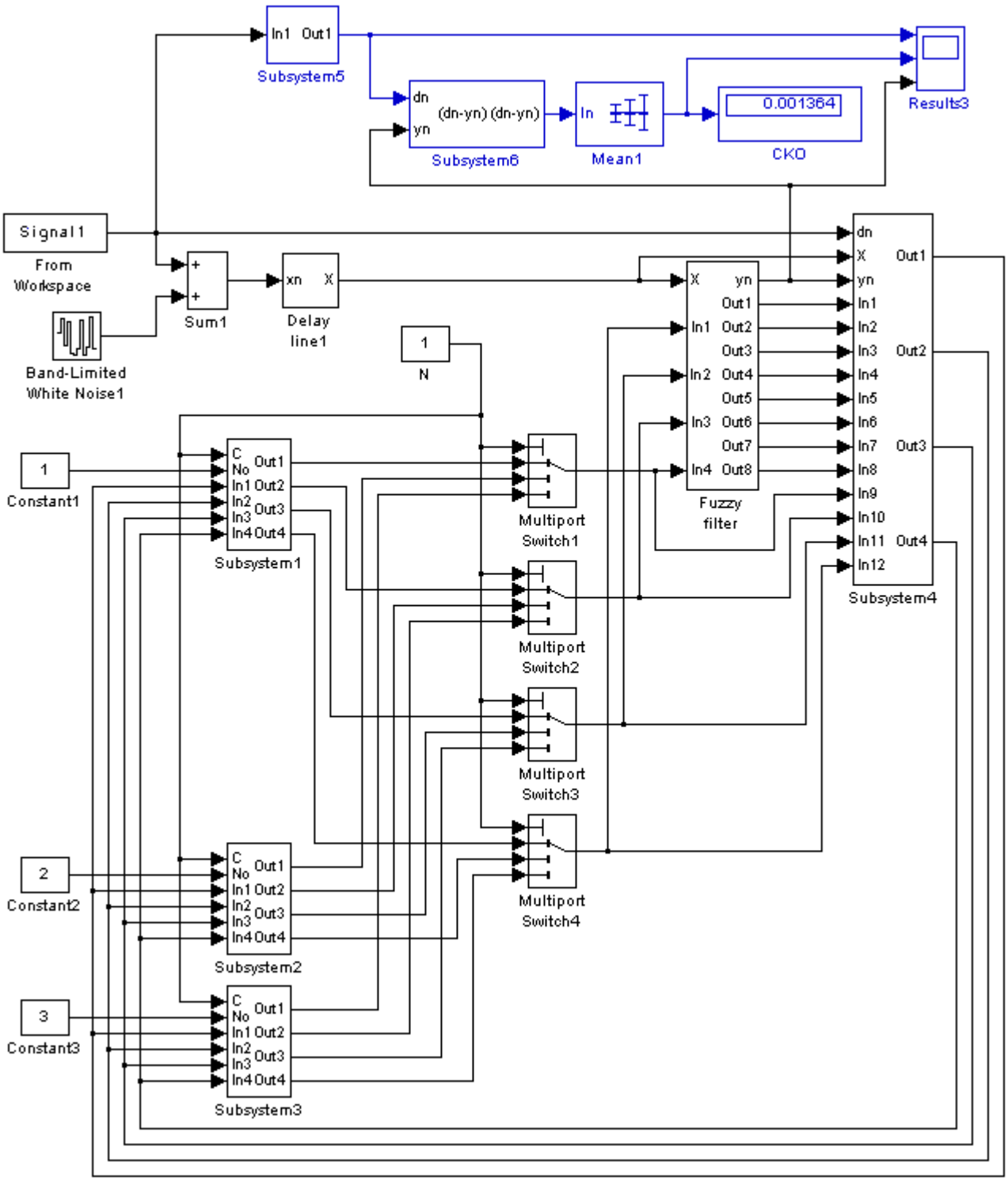


Рис. 6.1 Компьютерная модель адаптивного цифрового фильтра (первая часть)

Первая часть модели (рис. 6.1) предназначена для выполнения обучения цифрового фильтра. На ее вход подаются обучающие сигналы. В качестве критерия качества обучения используется критерий минимума СКО. После проведения обучения, т. е. непосредственно для обработки сигналов, используется вторая часть модели (рис. 6.2).

Рассмотренный адаптивный цифровой фильтр является фильтром нижних частот. Он позволяет изменять ширину полосы пропускания в соответствии с характеристиками обрабатываемого сигнала.

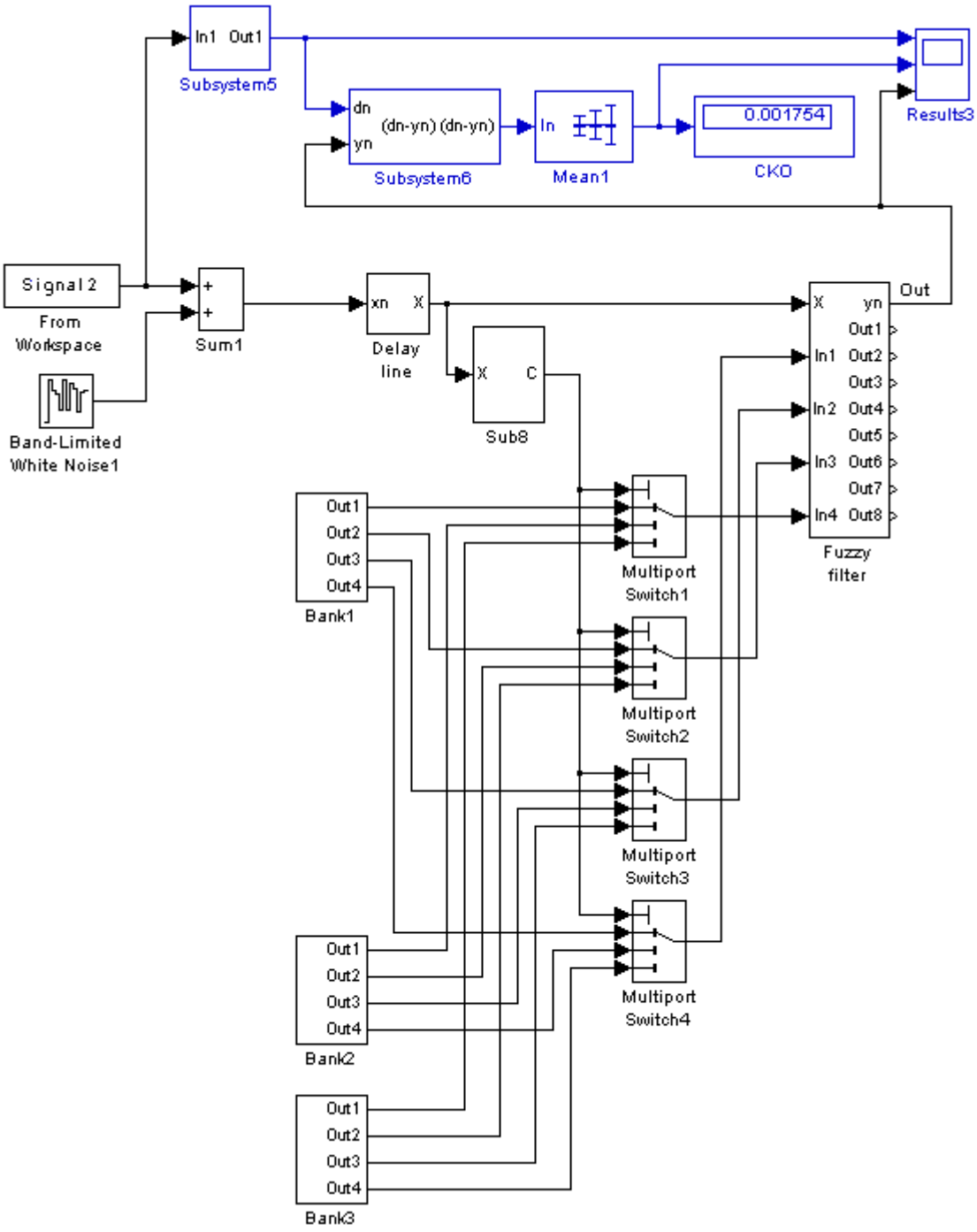


Рис. 6.2 Компьютерная модель адаптивного цифрового фильтра (вторая часть)

3. Порядок выполнения работы

3.1 Изучить теоретическую часть.

3.2 Загрузить файлы компьютерной модели адаптивного цифрового фильтра, представленной на рис. 6.1-6.2.

3.3 По заданию преподавателя выбрать входные сигналы и модель помех.

3.4 Получить следующие осциллограммы:

- входных сигналов в отсутствии шума.

- обучающего сигнала.
- входного и выходного сигналов фильтра при фиксированном значении дисперсии шума.

3.5 Построить графики зависимости среднеквадратичной ошибки от соотношения сигнал/шум для выбранных сигналов.

4. Содержание отчета

4.1 Структурная схема модели цифрового фильтра в системе MATLAB.
4.2 Осциллограммы сигналов.
4.3 Графики функций (согласно пункту 3.4).
4.4 Выводы по проделанной лабораторной работе и анализ полученных результатов.

5. Контрольные вопросы

5.1 Понятие нечеткого множества. Функция принадлежности нечетких множеств.
5.2 Основные принципы цифровой фильтрации сигналов на основе теории нечетких множеств.
5.3 Методы построения функции принадлежности нечетких множеств, используемые в задачах цифровой фильтрации сигналов.
5.4 Постановка задачи оптимальной фильтрации случайных сигналов. Критерий оптимальности.

Литература

1. Уидроу Б., Стирнз С. Адаптивная обработка сигналов. – М. : Радио и связь, 1989. – 440 с.
2. Основы цифровой обработки сигналов: Курс лекций / А. И. Солонина и др. – СПб.: БХВ-Петербург, 2003. – 608 с.
3. Методы робастного, нейро-нечеткого и адаптивного управления / К. А. Пупков и др.; Под ред. Н. Д. Егупова. –М .: МГТУ им. Н. Э. Баумана, 2001. –744 с.
4. Вешкурцев Ю. М., Бычков Е. Д., Титов Д. А. Цифровой фильтр на основе теории нечетких множеств с адаптивно изменяемыми функциями принадлежности // Известия вузов России. Радиоэлектроника. 2007. Вып. 2. – С. 43-50.
5. Лазарев Ю. Ф. MatLAB 5.x – К.: Издательская группа BHV, 2000. – 384 с.
6. Дьяконов В. П. MATLAB 6.5 SP1/7 + Simulink 5/6 Основы применения – М.: СОЛОН-Пресс, 2005. – 800 с.

СОДЕРЖАНИЕ