Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Промышленная электроника»

ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ

ПРАКТИКУМ
по выполнению лабораторных работ
для студентов специальности 1-39 80 03
«Электронные системы и технологии»
дневной и заочной форм обучения

Электронный аналог печатного издания

Гомель 2023

УДК 621.3.083(075.8)
ББК 32.965.6я73
Ц75

Рекомендовано к изданию научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 7 от 09.03.2022 г.*)

Составители: *В. В. Щуплов, Н. А. Красовская*

Рецензент: доц. каф. «Информационные технологии» ГГТУ им. П. О. Сухого
канд. техн. наук, доц. *В. С. Захаренко*

Ц75 **Цифровая** обработка сигналов : практикум по выполнению лаборатор. работ для студентов специальности 1-39 80 03 «Электронные системы и технологии» днев. и заоч. форм обучения / сост.: В. В. Щуплов, Н. А. Красовская. – Гомель : ГГТУ им. П. О. Сухого, 2023. – 44 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

ISBN 978-985-535-513-8.

Представлены лабораторные работы с краткими теоретическими сведениями, порядком их выполнения, заданиями для самостоятельной работы.

Для студентов специальности 1-39 80 03 «Электронные системы и технологии» дневной и заочной форм обучения.

УДК 621.3.083(075.8)
ББК 32.965.6я73

ISBN 978-985-535-513-8

Лабораторная работа № 1

Моделирование обработки сигналов в программе Scilab

1. Основные сведения о языке Scilab

1.1. Константы языка Scilab

1.1.1. Вещественные числа

В настоящем разделе дано беглое описание основных конструкций языка Scilab.

Scilab поддерживает различные способы записи вещественных чисел. Например, допустимы записи: 4, -5.2, 32e3, 7.8e-5, -12E-10, -12D-10. Разделителем между мантиссой и порядком при вводе могут служить символы e, E, d, D. При выводе на экран Scilab использует символ D. Заметим, что разделителем между целой и дробной частями числа, как это принято в программировании, служит символ «точка», а не «запятая». Запись -5,2 соответствует двум выражениям (-5 и 2).

В среде Scilab не делается различия между целыми и вещественными числами.

1.1.2. Мнимые числа

Для задания мнимых (комплексных) чисел используется предопределенная константа %i, обозначающая мнимую единицу $i = \sqrt{-1}$. Например, записи 43*%i, 5-3*%i соответствуют комплексным числам $43i$, $5-3i$.

Для извлечения вещественной и мнимой части числа используются функции real() и imag(). Так, если набрать в командной строке выражение imag(5-3*%i), то получим результат ans = -3.

Это значит, что переменной с резервированным именем ans (от англ. *answer* – «ответ») присвоено значение мнимой части комплексного числа $5-3i$.

1.1.3. Стандартные константы

Часто используемые величины предопределены в Scilab как следующие константы:

- %e – число *e* (основание натурального логарифма);
- %pi – число π;
- %inf – «бесконечность»;

- %nan – «не число» («*not a number*»);
- %t или %T – логическая единица, «истина» («*true*»);
- %f или %F – логический нуль, «ложь» («*false*»);
- %z – полином с единственным нулевым корнем и аргументом *z*;
- %s – полином с единственным нулевым корнем и аргументом *s*.

1.2. Основные типы данных

1.2.1. Скалярные объекты

Оператором присваивания

```
a=1
```

вводится *вещественная переменная a*, которой присваивается значение 1.

Оператор

```
b=%t
```

вводит *переменную логического типа*, для которой устанавливается значение %t («истина»).

Строка символов («*string*») вводится с помощью одиночных или двойных апострофов, например:

```
s='This is a string'
```

На языке Scilab можно описывать данные типа «многочлен». Выполним оператор:

```
p=poly(3,'z')
```

Его результатом будет многочлен первой степени от агрумента *z*, имеющий один корень 3:

```
p =
      -3 + z
```

Встроенная функция *poly* имеет два аргумента. Первым аргументом является массив корней многочлена (о массивах см. п. 1.2.2).

Второй аргумент – строковая константа из одного символа. При дальнейших действиях этот символ в программе выступает в качестве аргумента многочлена.

Многочлен можно задавать и его коэффициентами. Для этого сначала определим следующий многочлен, имеющий один нулевой корень:

```
z=poly (0,'z')
```

Фактически мы определили независимую переменную *z*, через которую обычными математическими операциями можно задавать другие многочлены. Например, оператор

```
p=1+3*z+4.5*z^2
```

приводит к результату

p =

$$1 + 3z + 4.5z^2$$

1.2.2. Массивы

Более сложными объектами языка являются массивы и списки.

С точки зрения синтаксиса такие математические объекты, как векторы и матрицы представляются массивами с одним или двумя измерениями (соответственно). Термины «массив» и «матрица» (или «вектор») будут далее использоваться как синонимы и употребляться в зависимости от контекста. Язык Scilab позволяет работать и с массивами, имеющими более двух измерений, которые нельзя трактовать как матрицы.

Для задания массивов через значения их элементов используются квадратные скобки. Например, массив X, состоящий из шести элементов, задается оператором

```
X=[1 3 4 5 3 7]
```

Введенный так массив X соответствует вектор-строке

$$X = [1,2,4,5,3,7] \in \mathbf{R}^{1\times 6}.$$

Обращение к элементам массива производится через круглые скобки. Например, для доступа ко второму элементу следует ввести команду x=X(2). В результате получим x=3.

В отличие от массива список может содержать элементы различных типов.

1.2.3. Списки

Список – это упорядоченная последовательность элементов, каждый из которых, в свою очередь, может быть либо списком, либо *атомарным неделимым элементом.* Список может использоваться для группировки разнотипных объектов в единый объект данных. Для создания списка используется оператор list. Элементы списка могут быть различных типов. Например:

```
L=list(a,-(1:5), Mp,['this','is';'a','list'])
```

Получим следующий список:

```
L =
                L (1)
      3.    5.    2.    6.    8.    3.    5.    8.
                L (2)
      - 1.  - 2.  - 3.  - 4.  - 5.
                L (3)
      1 + 3 z + 4 . 5 z 2          1 - z
      1                            z + 3z^2 + 4.5z^3
                L(4)
!     this is  !
!              !
!     a list   !
```

Создадим теперь *типизированный список*, с помощью которого можно описывать новые абстрактные типы данных в SciLAB. Названия класса и его полей задаются с помощью вектора – строки в первом элементе типизированного списка. Например:

```
Lt=tlist(['mylist','color','position', 'weight'], 'blue',[0,1],10)
```

Первый аргумент функции list – вектор строк, первый элемент которого описывает сам список, а последующие определяют типы элементов списка:

```
Lt =
        Lt(1)
!mylist color position weight!
        Lt(2)
blue
        Lt(3)
0       1.
        Lt(4)
10.
```

Таким образом, типу ’color’ будет соответствовать элемент ’blue’. Обращение к элементу типизированного списка осуществляется следующим образом: Lt(’color’). Добавить к списку еще один элемент можно так: L($+1)=B. Таким образом, мы добавили в конец списка матрицу *B*. Объединить два списка можно командой NewL=lstcat(L,L1). Этим мы сформировали новый список, который состоит из элементов списков L и L1.

1.3. Действия над матрицами

Для начала создадим вектор из восьми элементов. Для этого введем в командной строке оператор:

```
a = [3 5 2 6 8 3 5 8]
```

После нажатия клавиши Enter в командном окне появится ответ:

```
a =
      3. 5. 2. 6. 8. 3. 5. 8.
```

Для перехода к следующей строке при определении матрицы используется символ «;». Например, в результате выполнения оператора:

```
X = [2 6 -3 2; 6 7 9 8]
```

получим массив X размера (2 × 4):

X =

2. 6. - 3. 2.
6. 7. 9. 8.

Он соответствует матрице:

$$X = \begin{bmatrix} 2 & 6 & -3 & 2 \\ 6 & 7 & 9 & 8 \end{bmatrix} \in \mathbf{R}^{2\times 4}.$$

Элементы массивов должны быть одного общего типа, но не обязательно числового. Так, кроме массива вещественных чисел можно задать и массив строк:

St = [’this’ ’is’ ’a ’ ’string’ ’array’]

Получим переменную:

St =

!this is a string array!

Тогда в ответ на оператор whos, вызывающий на экран список используемых системой переменных, получим:

St string 1 by 5 120

Это означает, что двумерный массив St строкового типа имеет размерность 1 по первому измерению, размерность 5 – по второму и занимает 120 байт памяти.

Можно задать и массив логических констант:

B = [%t %f],

или массив (матрицу) многочленов.

Для определения многочленной матрицы необходимо сначала задать составляющие ее многочлены. Зададим многочлен $P(z)$ вида $P(z) = 1 + 3z + 4{,}5z^2$:

```
P = 1+3*%z+4.5*%z^2;
```

Заметим, что символ «;» в конце оператора используется, чтобы результат его выполнения не выводился на дисплей.

Теперь создадим матрицу многочленов:

```
Mp = [P, 1-%z; 1, %z*P]
```

В командном окне увидим результат:

Mp =

$1 + 3z + 4.5z^2$ $\quad 1 - z$

$1 \qquad\qquad\quad z + 3z^2 + 4{,}5z^3$

Для того чтобы узнать размер массива, используется оператор size(). Например, узнаем размер ранее введенного массива A. Результатом выполнения команды size(A) будет: ans= 2. 4. Это значит, что массив **A** имеет две строки и четыре столбца.

Рассмотрим теперь некоторые операции над матрицами. В результате выполнения команды C=X+4 мы увидим:

```
C =
    6.    10.    1.    6.
   10.    11.   13.   12.
```

Это значит, что каждый элемент матрицы *C* получен увеличением элементов матрицы *X* на 4.

Создадим теперь две матрицы размером 2 x 3:

```
A= [2 4 3; 7 5 1];
B= [5 2 8; -1 9 0];
```

Попробуем их перемножить:

```
D=A*B
```

Результатом будет сообщение об ошибке, так как операции матричного умножения в системе Scilab понимаются в векторно-матричном смысле:

!--error 10
inconsistent multiplication

Для получения корректного ответа транспонируем одну из матриц. Операция транспонирования вещественных матриц задается символом «’» (апостроф):

C= X’

Получим результат:

C=

2. 6.
6. 7.
-3. 9.
2. 8.

Выполним теперь операторы:

D=A’*B
и
E=A*B’

В первом случае получим:

D =

3. 67. 16.
15. 53. 32.
14. 15. 24.

Во втором случае получили:

E =

42. 34.
53. 38.

Математически это значит, что в результате действий с матрицами $A, B \in \mathbf{R}^{2\times 3}$ получены матрицы:

$$D = A^{\mathrm{T}}B \in \mathbf{R}^{3\times 3}$$

и

$$E = AB^{\mathrm{T}} \in \mathbf{R}^{2\times 2}.$$

Если необходимо перемножить массивы поэлементно, то следует использовать оператор «.*», как показано в следующем примере:
D=A.*B

Получим:

```
D =
    10.     8.   24.
    -7.    45.    0.
```

Конечно, эта операция выполнима только для массивов одинаковой размерности. Аналогично поэлементное деление массивов задается операцией «./», а возведение в степень каждого элемента массива – операцией «.^». Если первым операндом (сомножителем или делимым) является число, то надо следить за использованием символа «.». Так, результатом выполнения оператора:

```
A=1./[1 2; 3 4]
```

будет

```
A =
    - 2.     1.
      1.5  - 0.5
```

То есть вычислена матрица $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} - 1 = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}$

Оператор

```
A=1../[1 2; 3 4]
```

приведет к массиву, элементы которого обратны исходным:

```
A =
    1.          0.5
    0.3333333   0.25
```

Коэффициенты характеристического многочлена матрицы можно найти с помощью функции *poly*:

```
Q=poly(A,'q')
```

Получим:

```
Q =
```

$$840 - 2038q + 1849q^2 - 820q^3 + 190q^4 - 22q^5 + q^6$$

Ниже перечислены еще несколько полезных функций:
det(A) – определитель матрицы A;
inv(A) – обратная к A матрица;
trace(A) – след (сумма диагональных элементов) матрицы A;
spec(A) – собственные числа и собственные векторы матрицы A.

1.4. Действия над многочленами

Создадим новый многочлен: W=5-2*z+1.1*z^2; и перемножим его с ранее созданным многочленом P:

```
S=P*W
```

Получим:

```
S =
```

$$2\ 3\ 4\ 5 + 13z + 17.6z^2 - 5.7z^3 + 4.95z^4$$

Корни многочлена можно найти с помощью функции *roots*:

```
r=roots(s)
```

Получим ответ:

```
r =
   -0.3333333 + 0.3333333i
   -0.3333333 - 0.3333333i
    0.9090909 + 1.928473
    0.9090909 - 1.928473i
```

Мы видим, что результат содержит мнимые числа.

Построим теперь график функции (многочлена) $S(z)$ для $z \in [0,3]$ с шагом $\Delta z = 0{,}05$. Для этого выполним последовательность операторов:

```
x=0:0.05:3;
y=horner(S,x);
xbasc();
plot(x,y)
```

Первой командой мы задали массив точек, в которых хотим вычислить значение многочлена $S(z)$. С помощью функции *horner* производится вычисление значений многочлена $S(z)$ в точках x и запись вычисленных значений в массив y. Далее с помощью функции *xbasc* очищается графическое окно. Затем с помощью функции *plot* получаем график функци $S(z)$. Результат показан на рис. 1.1.

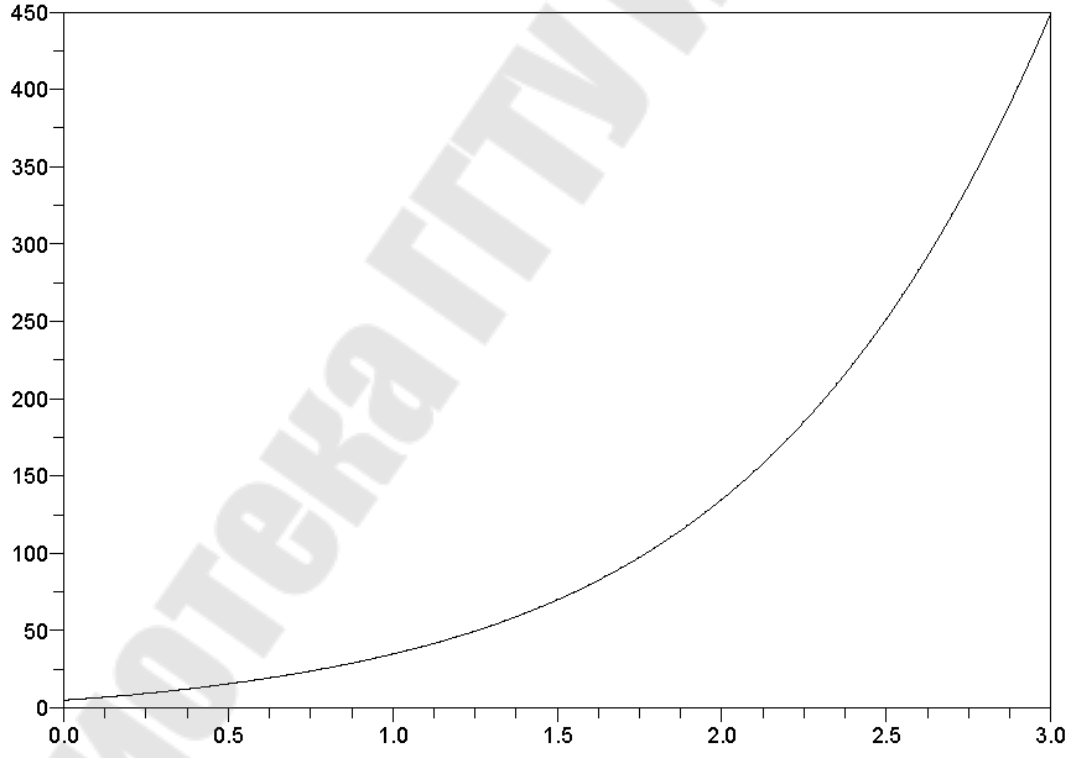


Рис. 1.1. График функции $S(z)$

Деление многочленов осуществляется функцией *pdiv*. Разделим многочлен $W(z)$ на многочлен $P(z)$:

```
[R,Q]=pdiv(P,W)
```

Получим результат деления:

```
Q =
    4.0909091
R =
    - 19.454545 + 11.181818z,
```

где Q – частное, а R – остаток от деления.

1.5. Пользовательские функции

1.5.1. Скрипт-файлы

Пользователь имеет возможность определить собственные программные модули, называемые *скрипт-файлами* (от англ. *script* – «сценарий»). Скрипт-файл не содержит заголовка и представляет собой последовательность команд. Эффект от его выполнения такой же, как если бы находящиеся в нем команды были набраны в командной строке. Создать скрипт-файл можно с помощью встроенного редактора Scipad или любого другого текстового редактора, например, Notepad (Блокнот) в текстовом формате (plain text). Скрипт-файл запускается на исполнение функцией *exec*. Например, если мы создали и сохранили скрипт-файл под именем script1.txt, то запустить его на исполнение следует строчкой exec('script.txt'). Если скрипт-файл не находится в рабочей директории, то в функции *exec* следует указать полный путь к нему.

1.5.2. Подпрограмма-функция

Определить собственную подпрограмму-функцию пользователь может, оформив ее в виде файла с расширением sci, используя встроенный редактор Scipad или любой другой текстовый редактор. Имя созданного файла и определяет имя функции, которое будет использоваться при ее вызове. В отличие от скрипт-файла подпрограмма-функция должна содержать заголовок и оператор окончания. Заголовок идет в первой строке и начинается оператором function. После этого в заголовке указываются имена выходных переменных (если переменных несколько, то их имена перечисляются в квадратных скобках). Затем после знака равенства указывается имя функции и приводится заключенный в круглые скобки список имен входных (формальных) переменных, раз-

деленных запятой. После заголовка идет само тело функции, за которым следует оператор окончания endfunction.

В качестве примера напишем подпрограмму, которая будет вычислять значения функции:

$$y(t)=\begin{cases} 1/\exp(t), \text{ если } t \succ 0, \\ \exp(t), \text{ иначе.} \end{cases} \qquad (1.1)$$

Для этого создадим редактором Scipad следующий файл:

```
function y=func (t)
y=zeros (t);
for j=1:length (t)
   if t (j)>0 then
   y(j)=1/%e^t(j);
   else
   y(j)=%e^t(j);
   end,
   end endfunction
```

Функцией *zeros* создается массив нулевых элементов такого же размера как и входной массив *t*. Далее внутри цикла for производится поэлементная проверка значений аргумента на выполнение условия (оператор if) и вычисление значений заданной функции. Сохраним файл в рабочей папке. Теперь вызовем функцию *func* из командного окна. Для этого выполним команду exec('func.sci'); После этого ее можно использовать так же как и стандартные функции языка Scilab.

1.6. Работа с графикой

1.6.1. Двумерные графики

В качестве первого примера вычислим значения функции (1.1) в точках отрезка [−5,5] с шагом 0,01 и построим ее график. Для этого выполним в командном окне следующую последовательность команд:

```
q=-3:0.001:3;
p=func(q);
xbasc();
```

```
xtitle('y(t)','t','y');
 plot(q,p);xgrid;
```

Результат показан на рис. 1.2.

В качестве другого примера построим график значений многочлена и отобразим расположение его корней.

Зададим многочлен $W(z)$ и вычислим его корни:

```
W=1-5*%z^2+%z^3-2*%z^4+4*%z^5-%z^6;
s=roots(W)
```

Получим:

```
s =
      0.4656907
    - 0.4070419
    - 0.4031434 + 0.9855944i
    - 0.4031434 - 0.9855944i
    1.3825675
    3.3650705
```

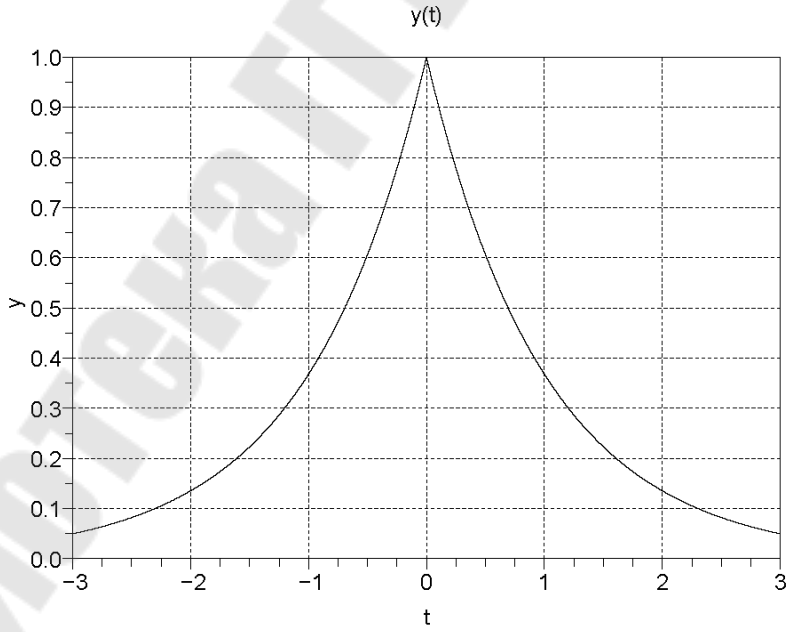


Рис. 1.2. График функции $y(t)$ вида (1.1)

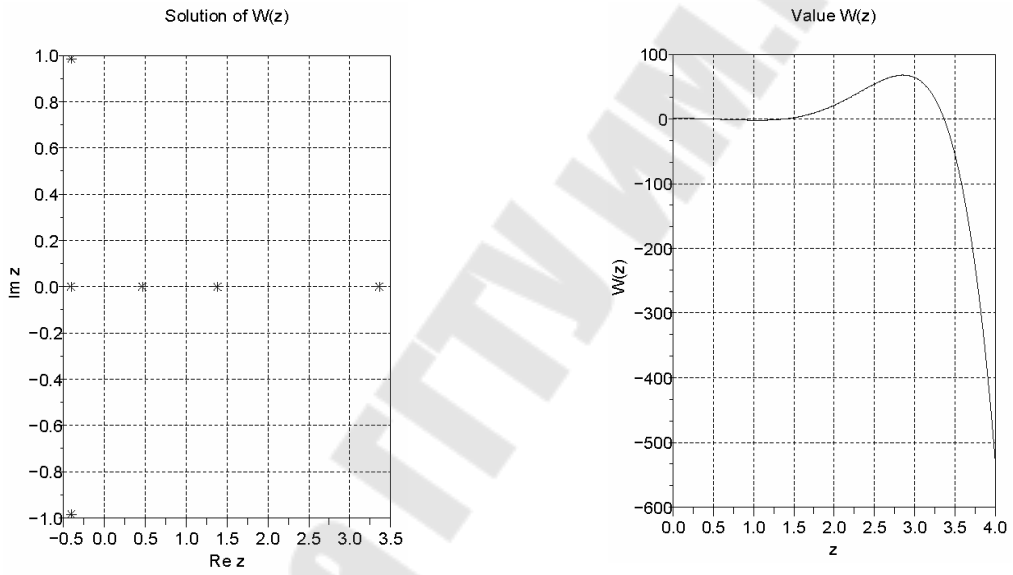
Теперь найдем значения многочлена $W(z)$ на отрезке $z \in [0,4]$ с шагом $\Delta z = 0{,}05$. Выполним следующие операторы:

```
x=0:0.05:4;
 y=horner(W,x);
```

Отобразим полученные результаты графически:

```
xbasc();
subplot(121) xtitle( 'Solution of W(z)', 'Re z', 'Im z'); xgrid();
plot(real(s),imag(s),'*');
subplot(122) xtitle( 'Value W(z)', 'z', 'W(z)');
xgrid(); plot(x,y);
```

Результаты показаны на рис. 1.3.



а) *б*)

Рис. 1.3. Корни (*а*) и значения (*б*) многочлена *W*(*z*)

Оператор subplot используется, если нужно вывести несколько графиков в одном графическом окне. Так, запись subplot(121) означает, что график будет выведен в первой ячейке «таблицы» графиков, которая получена разделением графического окна на одну строку и два столбца (нумерация ячеек слева направо, сверху вниз). С помощью функции *xtitle* обозначены оси и заголовок графиков, а функция *xgrid* служит для нанесения сетки на график.

Многочлен *W*(*z*) имеет комплексные корни. Чтобы выделить их вещественные и мнимые части, использованы встроенные функции *real*(*s*) и *imag*(*s*).

Чтобы вывести несколько графиков в одном графическом окне, используется функция *xsetech*. Построим еще три графика. В качестве примера напишем в виде скрипт-файла следующие команды:

```
t=(0:0.05:5)';
xbasc();
xsetech([0,0,0.5,0.5]);
plot2d(t, 2*sin(t)); xgrid();
xsetech([0.5,0,0.5,0.5]);
plot2d3(t,1.5+cos(t)); xgrid();
xsetech([0,0.5,1,0.5]);
plot2d([t t], ... [2*sin(t) 1.5+cos(t)],[12 10]); xgrid();
```

Теперь сохраним наш скрипт-файл под именем graph с расширением sce. Запустим его на выполнение командой: *exec*('graph.sce');

Получим графики, показанные на рис. 1.4. Функции *xsetech* в качестве аргумента передается вектор из четырех элементов, которые являются координатами графика относительно графического окна. Первые два элемента – координаты левого верхнего угла, а третий и четвертый – координаты правого нижнего угла. На втором графике с помощью функции *plot2d3* нарисована гистограмма, а на третьем показано, как можно построить несколько графиков на одной координатной сетке. Последним аргументом указаны цвета линий: число 12 соответствует синему цвету, а 10 – голубому.

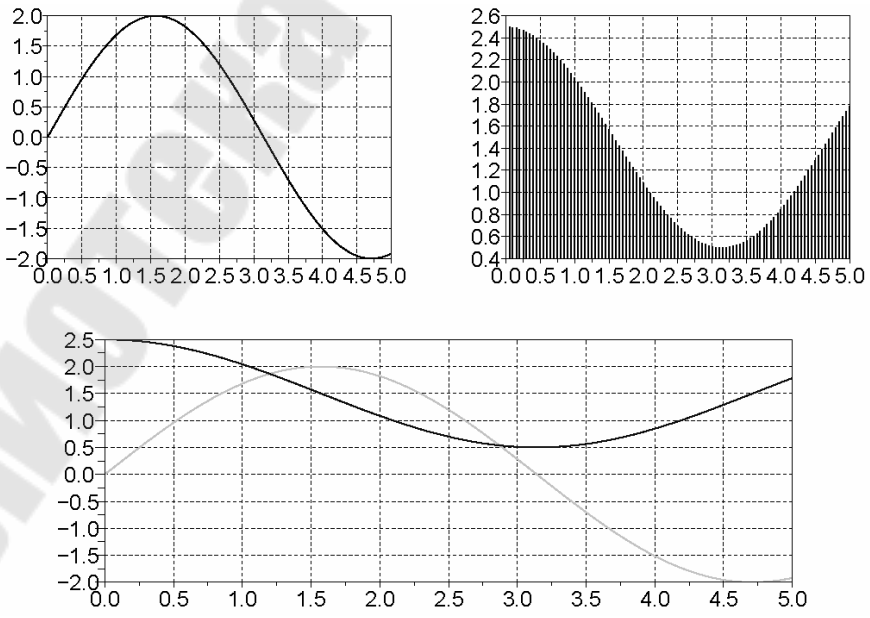


Рис. 1.4. Пример использования функции *xsetech*

1.6.2. Построение трехмерных изображений

Построим трехмерное графическое изображение тора. Поверхность тора в декартовых координатах (*x*, *y*, *z*) может быть задана параметрически через независимые аргументы α и β, принимающие значения от −π до π следующим образом:

$$x = \sin\alpha(2 - \sin\beta);$$
$$y = \cos\alpha(2 + \sin\beta);$$
$$z = \cos\beta.$$

Составим программу:

```
[al,bet]=meshgrid(-%pi:%pi/20:%pi);
x=sin(al).*(2+sin(bet));
y=cos(al).*(2+sin(bet));
z=cos(bet);
```

Функция *meshgrid* создает двумерные массивы al и bet с заданным интервалом (в нашем примере – [−π, π]) и шагом изменения (здесь шаг равен π/20). Отобразим полученную поверхность графически последовательностью команд:

```
xbasc();
xgrid();
mesh(x,y,z);
```

Функция *mesh* строит аксонометрическое изображение поверхности, показанное на рис. 1.5.

Функция, позволяющая строить раскрашенные трехмерные поверхности, – *surf*. Пример ее использования показан в следующей программе:

```
    [x,y]=meshgrid(-10:0.5:10);
z = 0.1*exp(-(0.05*x.^2+0.01*y.^2))-...
    0.7*exp(-(0.1*x.^2+0.3*y.^2))+exp(-(0.25*x.^2+0.9*y.^2));
surf(x,y,z,'facecol','interp');
```

Данной программой поверхность задана функцией $z = f(x, y)$ вида:

$$z = 0{,}1\exp(-(0{,}05x^2 + 0{,}01y^2)) -$$
$$- 0{,}7\exp(-(0{,}1x^2 + 0{,}3y^2)) + \exp(-(0{,}25x^2 + 0{,}9y^2)).$$

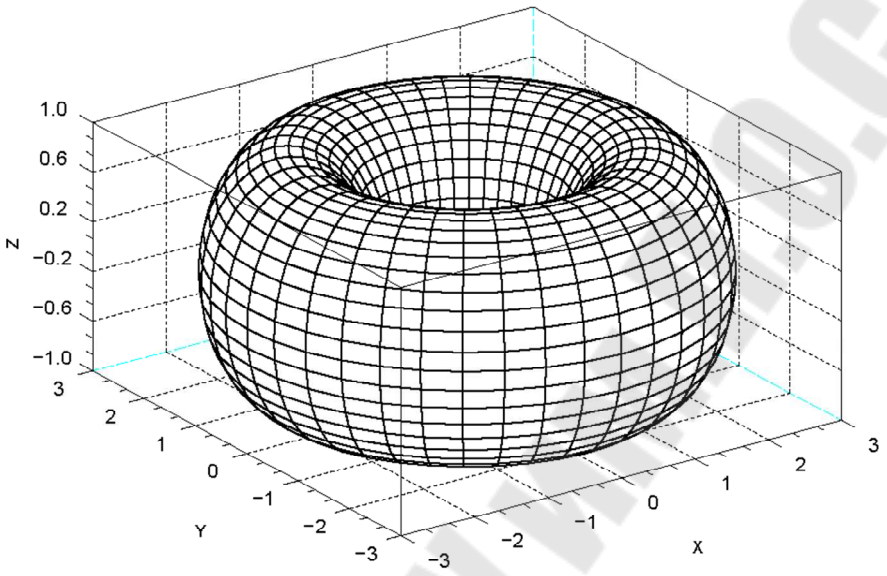


Рис. 1.5. Поверхность тора

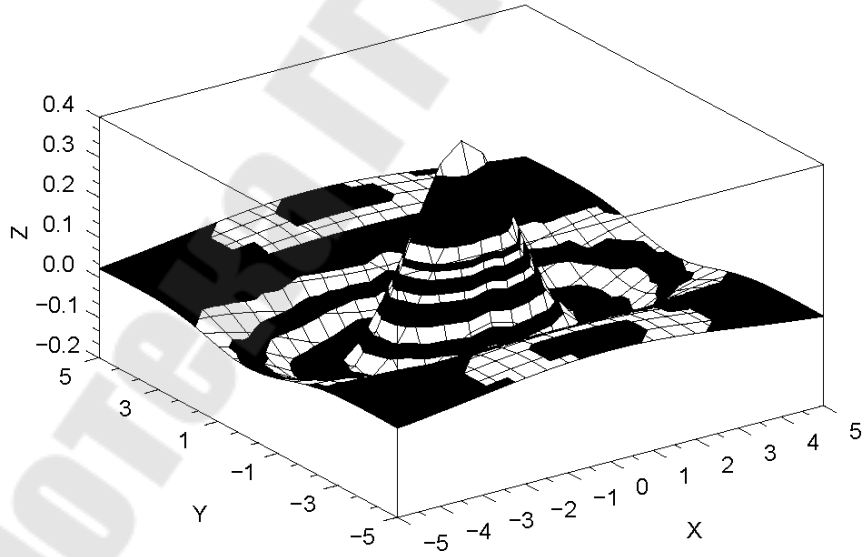


Рис. 1.6. Использование функции *surf*

Результат показан на рис. 1.6. Свойством facecol задается схема затенения. Это свойство может быть установлено как ’flat’ (по умолчанию) – раскраска граней, или ’interp’ – выполняется билинейная цветовая интерполяция.

Рассмотрим пример редактирования графика. Построим график поверхности $z = x^2 + y^2$. Выполним команды:

```
[x,y]=meshgrid(-3: 0.25: 3);
z=x.^2+y.^2;
xbasc();
mesh(x,y,z);
xset('window',1);
a=gca();
a.grid=[1 2 -1];
a.labels_font_size=3;
a.x_label.text='x';
a.x_label.font_size = 4;
a.y_label.text='y';
a.y_label.font_size = 4;
a.z_label.text='z';
a.z_label.font_size = 4;
a.title.text="z=x^2+y^2";
a.title.font_size = 4;
mesh(x,y,z);
a.isoview="on";
```

В результате (рис. 1.7) построено два графика одной поверхности: первый – с настройками по умолчанию; второй – отредактированный.

Команда xset('window',1) использована для того, чтобы вывод второго графика был произведен в новое графическое окно. Командой gca был получен описатель (handle) осей графика, через который можно менять их свойства.

Командой a.grid= [1 3 -1] устанавливается координатная сетка по осям *x*, *y* и *z* соответственно: по оси *x* – сетка черного цвета, по оси *y* – зеленого, а по оси *z* – вообще отсутствует.

Свойство labels_font_size устанавливает размер шрифта шкалы. Далее были установлены названия и размеры шрифта осей и графика.

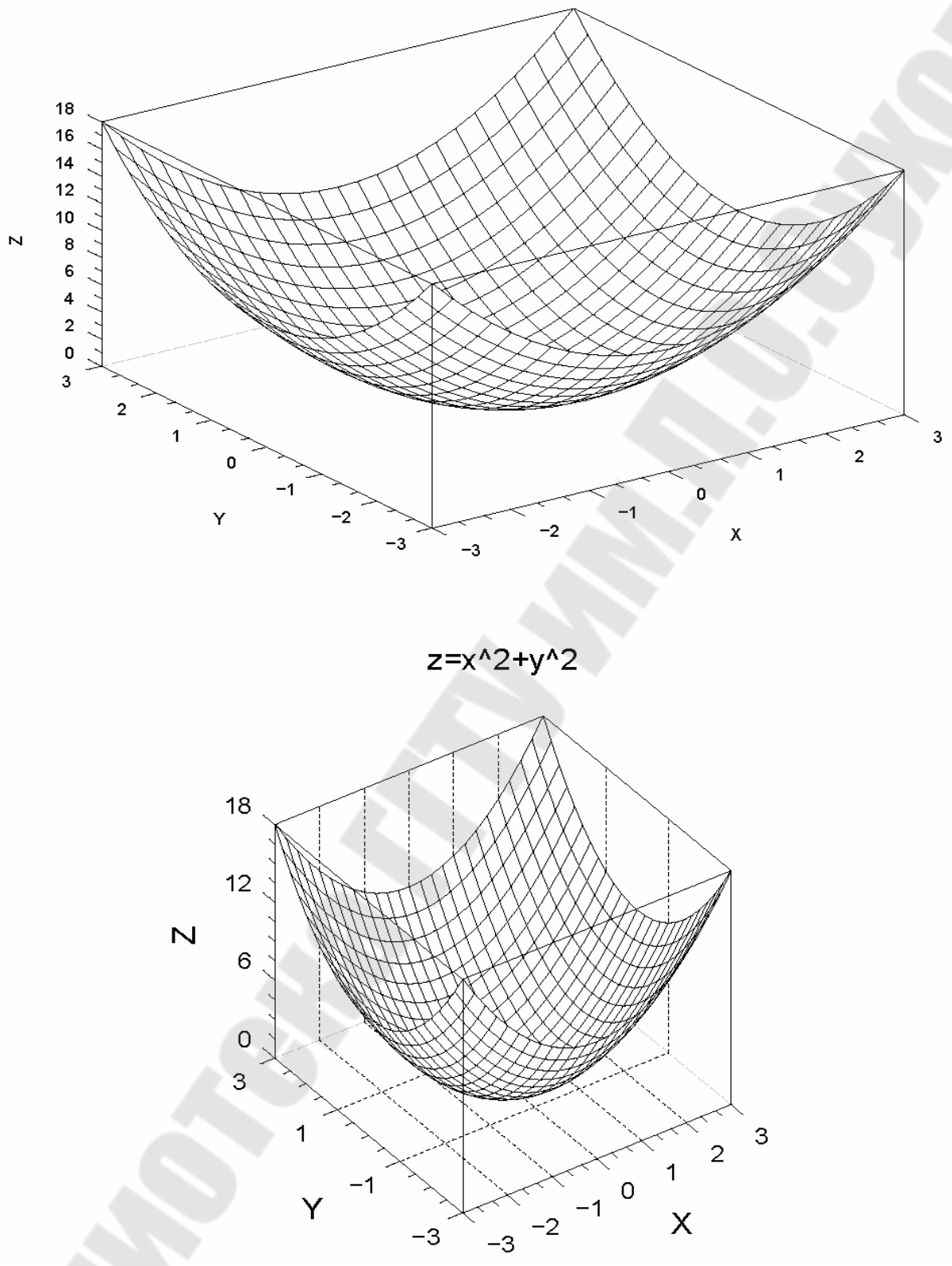


Рис. 1.7. Пример редактирования изображения

Лабораторная работа № 2

Моделирование фильтрации сигналов

2.1. Функции обработки сигналов

В среде Scilab имеется много функций обработки сигналов. Ниже будут рассмотрены основные из них, а именно – отображение сигнала, расчет фильтров и оценка спектра сигнала [5].

2.1.1. Фильтрация и расчет фильтров

Линейный стационарный фильтр может быть рассчитан операцией свертки. Рассмотрим дискретный сигнал $x(n)$ и дискретный фильтр $h(n)$, тогда результат свертки $y(n)$ определяется как

$$y(n) = \sum_{k=-\infty}^{\infty} h(n-k)x(k).$$

Свертка осуществляется функцией *convol*:

```
x=1:10;
h=ones(1,2);
y=convol(x,h)
```

С помощью функции *ones* создается матрица, состоящая из единиц, параметры этой функции – количество строк и столбцов создаваемой матрицы (в данном примере – одна строка и два столбца). Результат свертки:

```
y =
    1.3. 5. 7. 9. 11. 13. 15. 17. 19. 10
```

Фильтрацию дискретных сигналов линейных систем, созданных функцией *syslin*, можно выполнить функцией *flts*. Рассмотрим пример фильтра нижних частот:

```
f=iir(3,’lp’,’butt’,[.1 0],[0 0])
```

Функцией *irr* задается фильтр с бесконечной импульсной характеристикой (БИХ-фильтр). Первый параметр функции – порядок

фильтра, второй – строка, указывающая тип фильтра (lp – фильтр низких частот), третий аргумент указывает схему расчета фильтра (butt – фильтр Баттерворта), далее вектором из двух элементов задается частота среза (для фильтров низких и высоких частот используется только первый элемент вектора), последний аргумент – вектор из двух элементов – значения ошибки (используется только при фильтрах Чебышева и Кауэра). Полученный фильтр:

f =

$$\frac{2\ 3\ 0.0180989 + 0.0542968z + 0.0542968z^2 + 0.0180989z^3}{-0.2780599 + 1.1828933z - 1.7600419z^2 + z^3}$$

Далее необходимо определить фильтр как линейную систему и задать входной сигнал:

```
f=syslin('d',f);
t=1:150;
x1=sin(2*%pi*t/20);
x2=sin(2*%pi*t/4);
x=x1+x2;
```

Проведем фильтрацию сигнала и построим соответствующие графики:

```
y=flts(x,f);
xsetech([0,0,1,.5])
plot(x)
xtitle('Signal input','time','amplitude')
xsetech([0,0.5,1,.5])
plot(y)
xtitle('Filtered signal','time','amplitude')
```

Функция *flts* позволяет получить временную характеристику дискретной системы *f* с входным сигналом *x*. Результат показан на рис. 2.1.

Расчет фильтров с конечной импульсной характеристикой (КИХ-фильтр):

$$H(z) = \sum_{k=0}^{N} h_k z^{-k}$$

может быть выполнен в Scilab функцией *wfir*:

```
[wft,wfm,fr]=wfir('lp',35,[.15 0],'kr',[2.8 0]);
xsetech([0,0,1,1/3])
plot2d(fr,20*log10(wfm))
xtitle('Low-Pass filter with Kaiser window',...
'freq (Hz)','Amplitude (dB)')
[wft,wfm,fr]=wfir('sb',111,[.2 .4],'hm',[0 0]);
xsetech([0,1/3,1,1/3])
plot2d(fr,20*log10(wfm))
xtitle('Stop Band filter with Hamming window',...
'freq (Hz)', 'Amplitude (dB)')
[wft,wfm,fr]=wfir('bp',55,[.20 .340],'ch',[.005 -1]);
xsetech([0,2/3,1,1/3])
plot2d(fr,20*log10(wfm))
xtitle('Band Pass filter with Chebyshev window',...
'freq (Hz)', 'Amplitude (dB)')
```

Было создано три КИХ-фильтра: фильтр нижних частот с окном Кайзера, режекторный фильтр с окном Хэмминга и полосовой фильтр Чебышева.

Первый аргумент функции – строка, в которой указан тип фильтра; второй аргумент – порядок фильтра (целое число); третий аргумент – вектор из двух элементов, указывающий промежуток срезаемых частот; четвертый аргумент – строка, в которой указывается тип окна; пятый аргумент – вектор из двух элементов, задающий параметры окна.

Выходные параметры: *fr* – сетка частот; *wfm* – амплитудно-частотная характеристика (АХЧ) фильтра (связанная с *fr*); *wft* – коэффициенты фильтра. Результат показан на рис. 2.2.

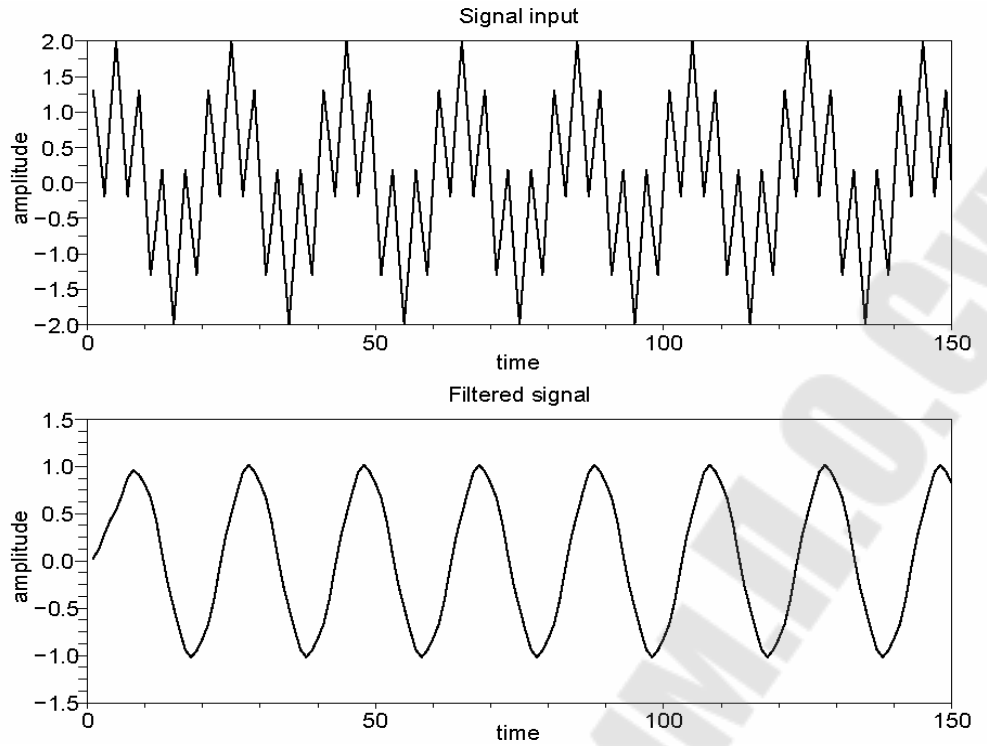


Рис. 2.1. Фильтрация сигнала при помощи функции *flts*

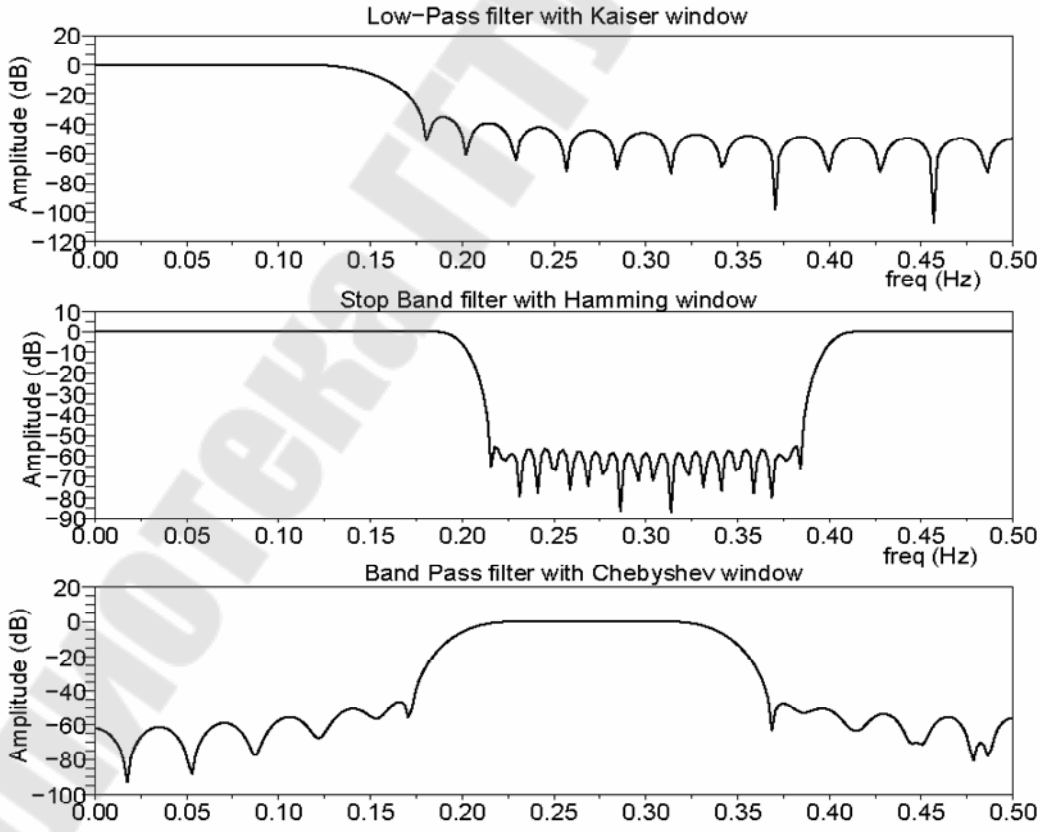


Рис. 2.2. Амплитудно-частотная характеристика КИХ-фильтров, полученных при помощи функции *wfir*

При вызове этой функции без параметров параметры фильтра вводятся пользователем интерактивно при помощи последовательности диалоговых окон. Для расчета КИХ-фильтров, оптимизированных по минимаксному критерию, используется функция *eqfir*. Эта функция выдает фильтр с лучшими характеристиками, чем функция *wfir*, а также может быть использована для расчета многополосного фильтра. Рассмотрим пример, позволяющий сравнить фильтры, рассчитанные этими двумя функциями:

```
[wft,wfm,fr]=wfir('lp',45,[.23 0],'kr',[5.6 0]);
bedge=[0 .2;.2875 .5]; des=[1 0];
wate=[.025 1];
hn=eqfir(45,bedge,des,wate);
xsetech([0,0,1,.5])
plot2d(fr,20*log10(wfm)),
xgrid() xtitle('Low-Pass filter using wfir',...
'freq (Hz)', 'Amplitude (dB)')
xsetech([0,.5,1,.5])
[hm,fr]=frmag(hn,256);
plot2d(fr,20*log10(hm)),
xgrid() xtitle('Low-Pass filter using eqfir',...
'freq (Hz)', 'Amplitude (dB)')
```

Первый аргумент функции *eqfir* – целое число, задающее длину фильтра; *bedge* – матрица размером *M* x 2, определяющая границы *M* полос пропускания фильтра; *des* – вектор длиной *M*, задающий желаемое значение для каждой полосы частот; *wate* – вектор длиной *M*, задающий допустимую величину ошибки для каждой полосы частот. Функцией *frmag* были получены значения частоты hm в точках fr. Входные параметры функции – вектор коэффициентов фильтра и число точек частотной характеристики. Результат показан на рис. 2.3.

Ниже приведен пример многополосного фильтра, полученного при помощи функции *eqfir*:

```
bedge=[0 .1;.15 .2;.25 .3;.35 .4;.45 .5;];
des=[0 1 0 .5 0];
wate=[1 .025 1 1 1];
hn=eqfir(45,bedge,des,wate);
[hm,fr]=frmag(hn,256);
plot2d(fr,hm), xgrid()
xtitle('5-band filter', 'freq (Hz)', 'Amplitude (dB)')
```

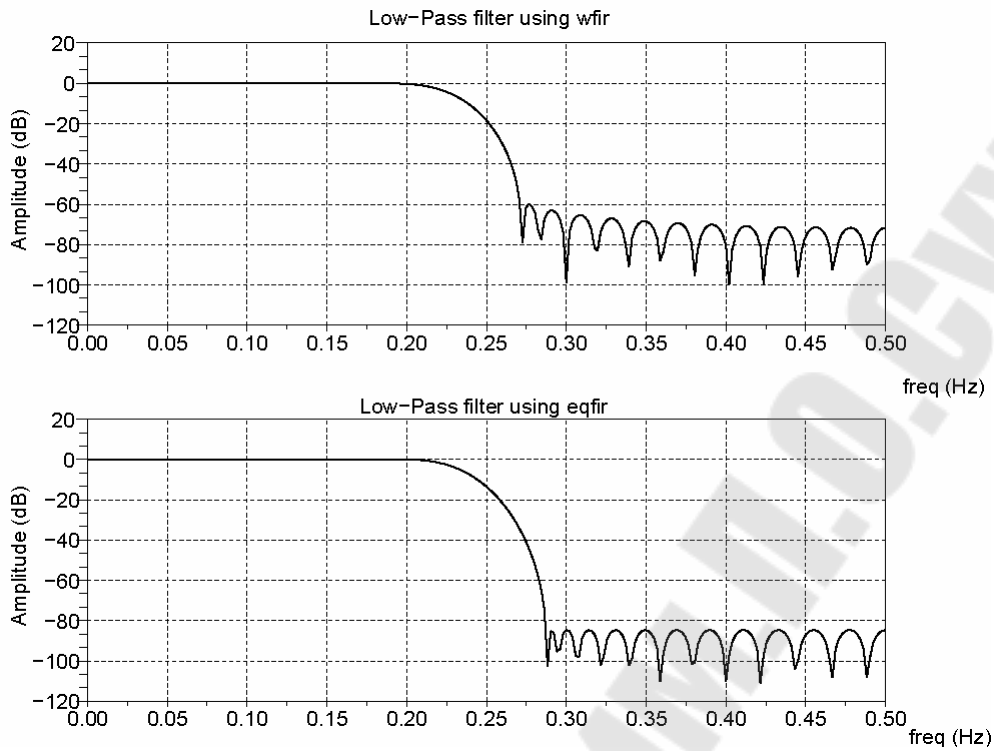


Рис. 2.3. Сравнение амплитудно-частотной характеристики фильтров, созданных функциями *wfir* и *eqfir*

Результат показан на рис. 2.4.

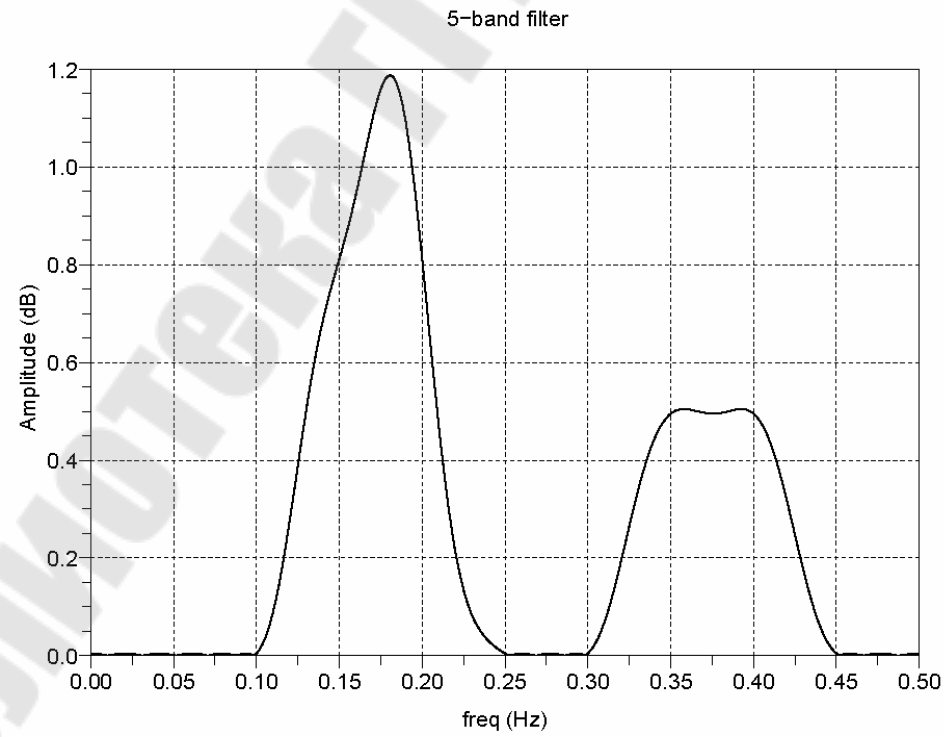


Рис. 2.4. Амплитудно-частотная характеристика многополосного фильтра, полученного при помощи функции *eqfir*

Для расчета фильтров с бесконечной импульсной характеристикой помимо функции *iir* может быть использована функция *eqiir*. Основное различие между ними в том, что в функции *eqiir* порядок фильтра не задается.

Произведем расчет эллиптического фильтра низких частот, полосового фильтра Чебышева и режекторного фильтра Баттерворта:

```
[c,g]=eqiir('lp','ellip',2*%pi*[.1 .2],.05,.025);
[hf1,fr1]=frmag(g*prod(c(2))/prod(c(3)),256);
[c,g]=eqiir('hp','cheb2',2*%pi*[.3 .4],.05,.025);
[hf2,fr2]=frmag(g*prod(c(2))/prod(c(3)),256);
[c,g]=eqiir('sb','butt',2*%pi*[.1 .2 .3 .4],.05,.025);
[hf3,fr3]=frmag(g*prod(c(2))/prod(c(3)),256);
xsetech([0,0,1,1/3])
plot2d(fr1,hf1),
xgrid() xtitle('Elliptic low-pass filter', 'freq (Hz)',...
'Amplitude (dB)')
xsetech([0,1/3,1,1/3]) plot2d(fr2,hf2),
xgrid() xtitle('Chebyshev band-pass filter', 'freq (Hz)',...
'Amplitude (dB)')
xsetech([0,2/3,1,1/3])
plot2d(fr3,hf3), xgrid()
xtitle('Butterworth stop-band filter', 'freq (Hz)',...
'Amplitude (dB)')
```

Функция *eqiir* имеет следующие параметры: первый – строка, указывающая тип фильтра; второй – строка, определяющая схему расчета фильтра; третий – вектор из четырех элементов, задающий частоты среза (для фильтров низких и высоких частот указываются только первые два элемента вектора); четвертый – пульсация в полосе пропускания; пятый – величина пульсации в полосе непропускания.

Функция возвращает вектор рациональных функций первого и второго порядка, дающих при перемножении передаточную функцию фильтра.

Результат показан на рис. 2.5.

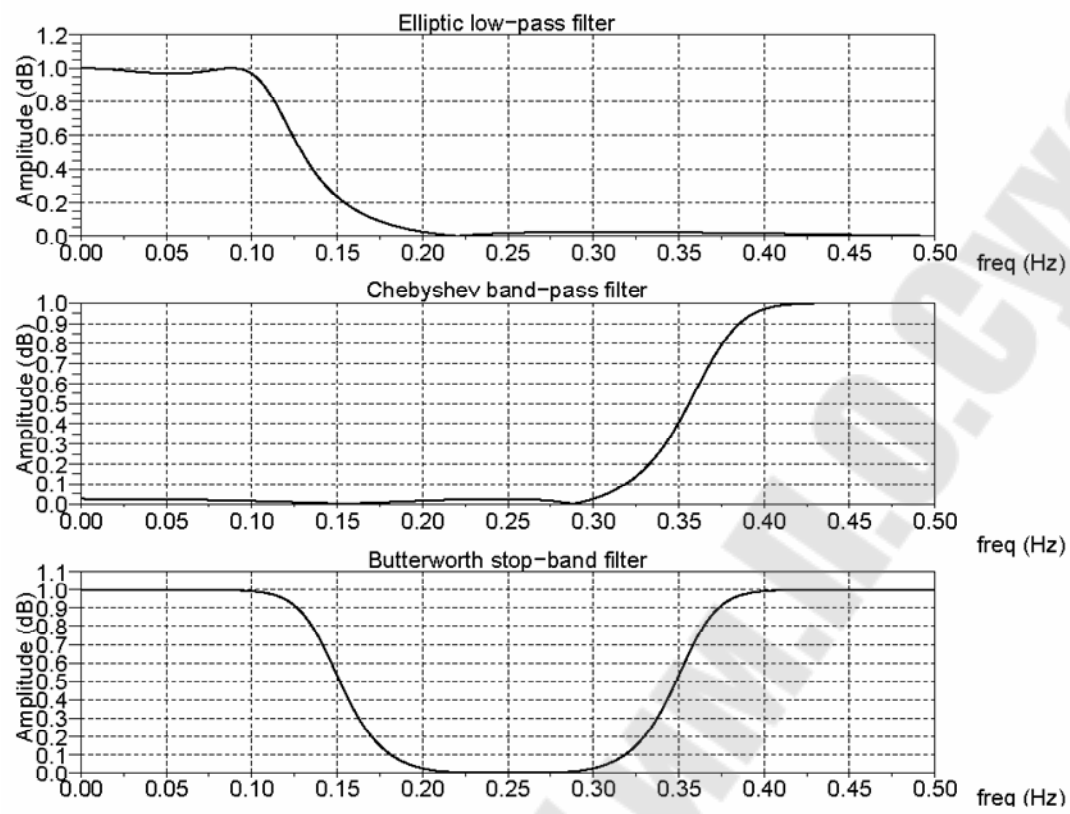


Рис. 2.5. Амплитудно-частотная характеристика БИХ-фильтров, полученных при помощи функции *eqiir*

2.2. Вычисление спектральной функции

Спектральная функция детерминированного конечного дискретного сигнала $x(n)$ определяется как квадрат модуля изображения Фурье этого сигнала:

$$S_x(w) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{-jwn}\right|^2.$$

Рассмотрим несколько способов расчета спектра сигнала с использованием среды Scilab.

2.2.1. Вычисление спектра случайного сигнала метдом периодограмм

Первый метод – метод периодограмм. Периодограмма конечной последовательности определяется как

$$I(w) = \frac{1}{U}\left|\sum_{n=0}^{N-1} w(n)x(n)e^{-jwn}\right|^2,$$

где $w(n)$ – вырезающая функция.

Если для вычисления сигнала используются K сегментов длиной N, то периодограмма оценки спектра $\hat{S}_x$ является средним значением K периодограмм:

$$\hat{S}_x(w) = \frac{1}{K}\sum_{k=0}^{K-1} I_k.$$

Эти периодограммы усреднены и нормализованы для достижения требуемой оценки спектральной функции сигнала.

Вычисление спектральной функции методом периодограмм осуществляется функцией *pspect*.

В качестве примера рассмотрим случайный сигнал, полученный пропусканием белого шума с единичной дисперсией через фильтр низких частот:

```
rand('normal');
rand('seed',0);
x=rand(1:4096-33+1);
nf=33;
bedge=[0 .1;.125 .5];
des=[1 0]; wate=[1 1];
h=eqfir(nf,bedge,des,wate);
y=convol(h,x);
sm=pspect(100,200,'tr',y);
[hf,fr]=frmag(h,100);
xsetech([0 0 1 .5])
plot2d(fr,20*log10(hf)), xgrid();
xtitle('Squared magnitude of the filter response',...
'freq (Hz)','magnitude (db)') xsetech([0 .5 1 .5])
plot2d(fr,20*log10(sm(1:100))), xgrid();
xtitle('Estimate of the spectrum',...
'freq (Hz)','magnitude (db)')
```

Для вызова функции *pspect* необходимо передать следующие параметры: сдвиг сегмента данных, длина каждого сегмента данных, тип окна (в данном примере используется треугольное окно) и вектор данных, задающих сигнал. В функцию можно передать вектор второго сигнала, тогда будет найдена оценка их взаимного спектра. Результат показан на рис. 2.6.

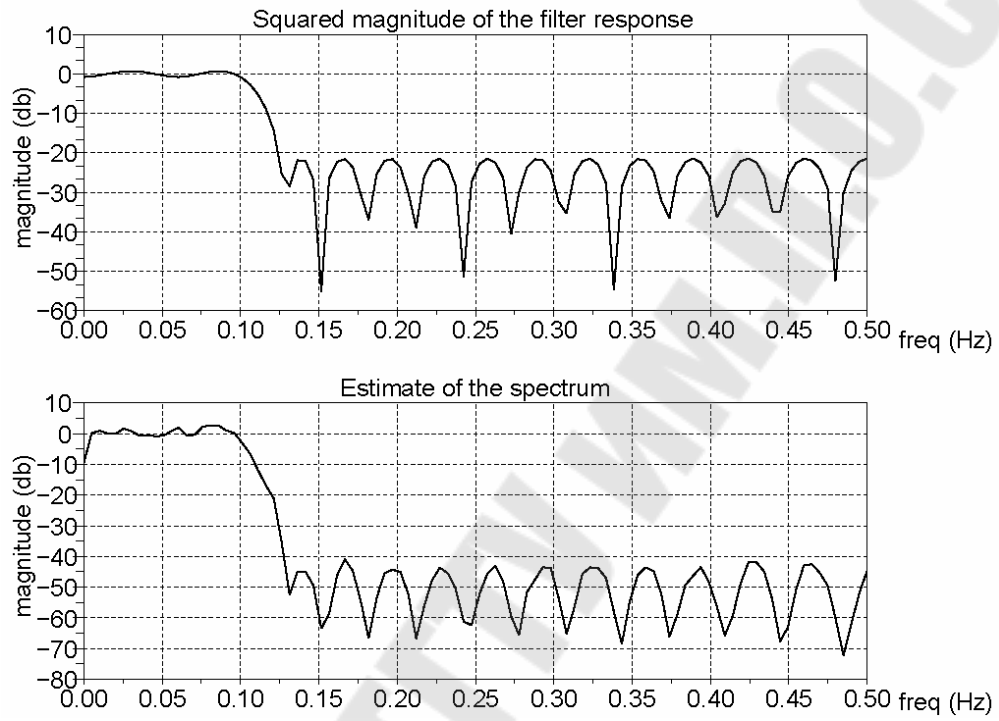


Рис. 2.6. Спектр сигнала, рассчитанный по *pspect*

2.2.2. Вычисление спектра случайного сигнала методом корреляции

Второй метод – метод корреляции – заключается в вычислении преобразования Фурье усредненной оценки автокорреляционной функции.

Для каждого *N*-точечного сегмента данных $x_k(n)$ оценка $2M$ точек автокорреляционной функции вычисляется следующим образом:

$$\hat{R}_k(m) = \sum_{n=0}^{N-1-m} x(n+m)x^*(n).$$

Для $m = 0, \pm 1, \pm 2, ..., \pm M$ оценка спектральной функции принимает вид:

$$\hat{S}_x = (w) = F\{\tilde{R}_x(m)w(m)\},$$

где F – оператор преобразования Фурье; $w(m)$ – вырезающая функция; $\widetilde{R}_x(m)$ – среднее значение K оценок:

$$\widetilde{R}_x = \frac{1}{K}\sum_{k=1}^{K}\widetilde{R}_k,$$

Корреляционный метод оценки спектра реализован функцией *cspect*.

Рассмотрим применение этой фунции, используя сигнал из предыдущего примера:

```
rand('normal');
rand('seed',0);
x=rand(1:4096-33+1);
nf=33;
bedge=[0 .1;.125 .5];
des=[1 0];
wate=[1 1];
h=eqfir(nf,bedge,des,wate);
y=convol(h,x);
sm=cspect(33,200,'ch',y,[.02,-1]);
[hf,fr]=frmag(h,100);
xsetech([0 0 1 .5])
plot2d(fr,20*log10(hf)), xgrid();
xtitle('Squared magnitude of the filter response',...
'freq (Hz)','magnitude (db)')
xsetech([0 .5 1 .5])
plot2d(fr,20*log10(sm(1:100))), xgrid();
xtitle('Estimate of the spectrum','freq (Hz)',...
'magnitude (db)')
```

Первые два параметра функции *pspect* – количество интервалов корреляции и количество преобразуемых точек; следующие параметры, как и в *pspect*, – выбор окна (в данном случае – окно Чебышева) и вектор, задающий сигнал.

Последний аргумент – параметры выбранного окна. Результат показан на рис. 2.7.

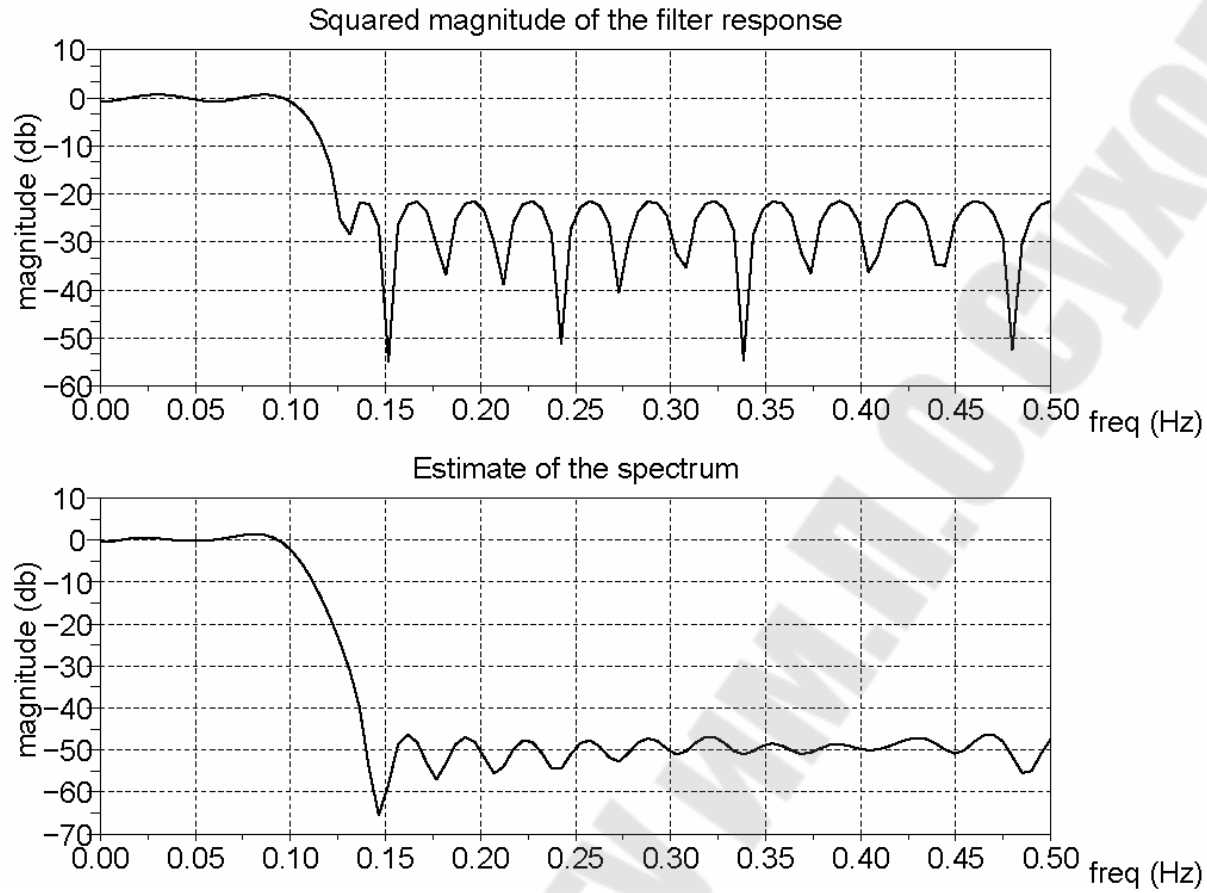


Рис. 2.7. Спектр сигнала, рассчитанный по *cspect*

2.2.3. Вычисление спектра квазипериодического сигнала преобразованием Фурье

Обратимся к задаче нахождения спектра сигнала по дискретной выборке на основе преобразования Фурье [1].

Рассмотрим квазипериодический процесс, представляющий собой сумму гармоник с несоизмеримыми частотами:

$$y(t) = \sin(t) + 0,75\sin\left(5\pi^{-1}t\right).$$

Зададимся интервалом дискретности T_0 = 0,05 и числом точек отсчетов $N = 2^{10} = 1024$.

Это соответствует длине реализации T = 51,2. Программа вычисления энергетического спектра представлена ниже:

```
clf();
dt=0.05;
Nt=2^10;
t=0:dt:(Nt-1)*dt;
```

```
y=sin(t)+0.75*sin(5/%pi*t);
plot2d(t,y)
xtitle('y(t)')
```

В этом фрагменте программы задается число точек для вычисления спектра и находится период дискретости.

Для этого вводится массив момента отсчета *t* с постоянным шагом *dt*, затем формируется подлежащий дальнейшей обработке массив *y*, соответствующий значениям функции *y*(*t*) для моментов отсчета *t*.

Оператором plot2d(*t*, *y*) выводится график этой функции.

После этого выполняется дискретное преобразование Фурье (ДПФ) с помощью процедуры fft и вычисляются энергетические спектры сигнала:

```
Y=fft(y,-1);
PY=abs(Y).^2/Nt;
wn2=2*%pi/dt;
dw=wn2/Nt;
```

Построим график спектра процесса в диапазоне частот [0,5] $с^{-1}$:

```
w=0:dw:5;
lw=length(w);
xset('window',1)
plot2d(w,PY(1:lw))
 xtitle('S(w)')
```

Результаты приведены на рис. 2.8.

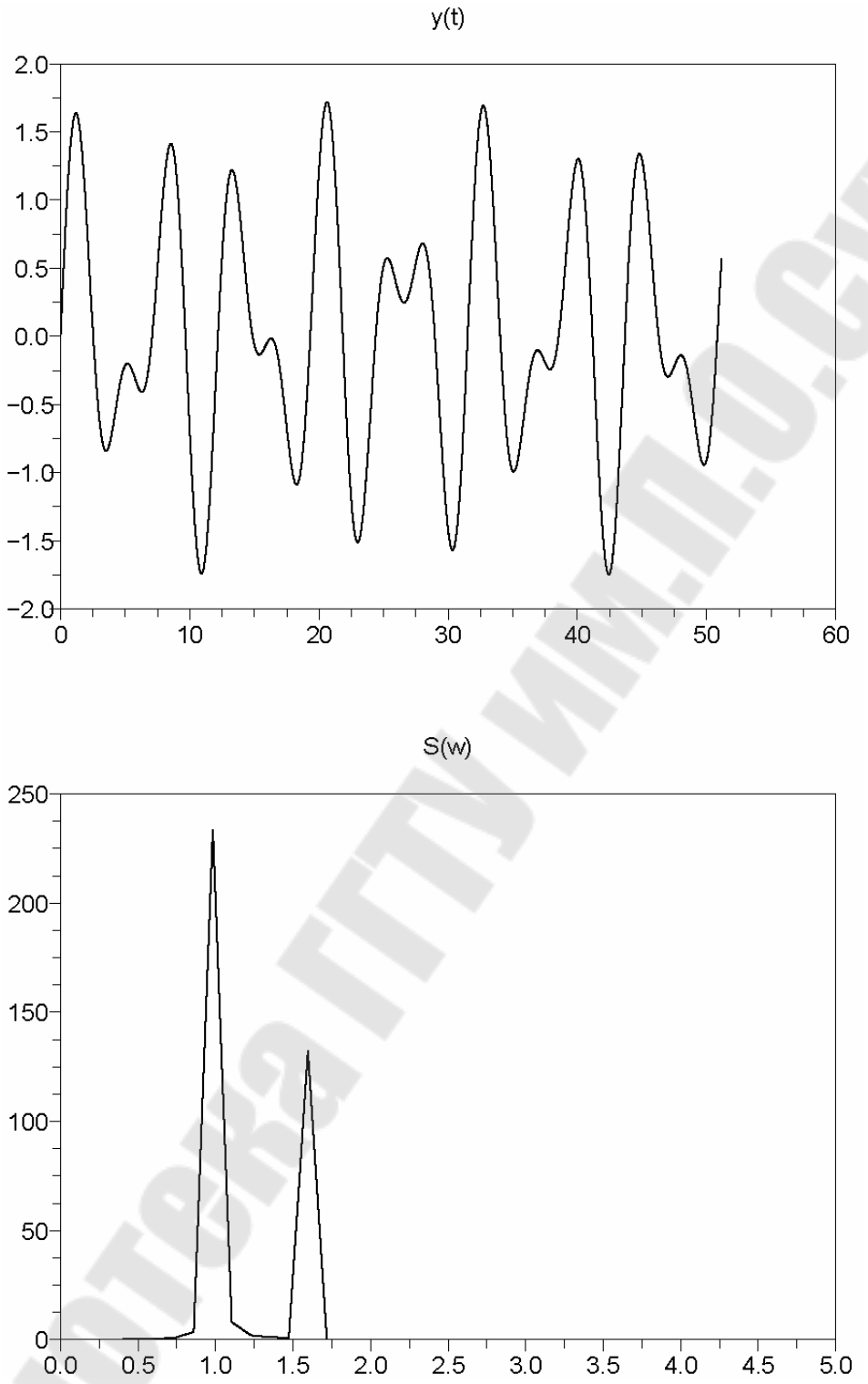


Рис. 2.8. Квазипериодический сигнал и его спектр

Лабораторная работа № 3

Исследование явления Гиббса

3.1. Основные сведения о явлении Гиббса

При исследовании реальных сигналов с помощью преобразований Фурье мы вынужденно сталкиваемся с двумя видами ограничений. Во-первых, применение дискретного преобразования Фурье предполагает конечность ряда выборочных наблюдений из реального сигнала, а во-вторых, в реальности нам всегда приходится использовать ограниченное число членов ряда Фурье даже для непрерывных сигналов. Явлением Гиббса называется особенность поведения частичных сумм ряда Фурье в окрестностях точки разрыва исследуемой функции. Эта особенность была впервые обнаружена Уилбрейамом в 1848 г., а позже переоткрыта Гиббсом в 1898 г. Суть проблемы состоит в том, что большинство методов анализа и обработки сигналов представляют собой или имеют в своем составе операцию свертки сигналов и различных функций. При этом в аналитических уравнениях как сам сигнал, так и оператор свертки, выполняющий определенную задачу обработки данных и реализующий определенную частотную функцию системы обработки, могут быть бесконечно большими. Реальная обработка сигналов или численное моделирование на ЭВМ может иметь дело только с ограниченными выборками данных и коэффициентов разложения. При стремлении представить эту функцию ограниченным спектром (или частичной суммой ряда Фурье) возникают два неприятных эффекта: 1) крутизна перепада в аппроксимирующей функции «размывается», так как она не может быть больше чем крутизна в нулевой точке, которую имеет наибольшая используемая гармоника; 2) по обе стороны от «размытых» перепадов появляются выбросы и затухающие осцилляции с частотой, близкой к частоте первого отброшенного члена ряда (на «радиотехническом жаргоне» – «звон»).

Указанные выбросы могут быть весьма значительными и достигают 9 % на основной частоте и 18 % – на удвоенной частоте основной гармоники.

Рассмотрим популярный пример: сигнал $s(t)$ типа меандра с периодом $T = 2t_{\text{И}}$ – последовательность прямоугольных импульсов со скважностью, равной двум, т. е. когда длительности импульсов и пауз между ними равны (рис. 3.1).

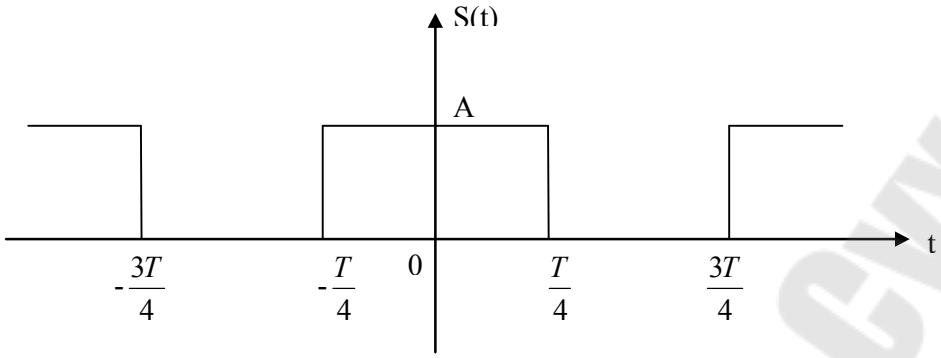


Рис. 3.1. Меандр

Разложим периодический сигнал $s(t)$ в тригонометрический ряд Фурье. Данный сигнал является четной функцией. Тогда k-я гармоника разложения меандра в ряд Фурье будет определяться как

$$a_k = \frac{2}{T}\int_{\frac{-\tau_И}{2}}^{\frac{\tau_И}{2}} A_0 \cos\left(\frac{2\pi}{T}kt\right)dt = \frac{2A_0}{\pi k}\sin\left(\frac{\pi k \tau_И}{T}\right).$$

У меандра $T = 2\tau_И$, поэтому

$$a_k = A_0 \frac{\sin\left(\frac{\pi k}{2}\right)}{\frac{\pi k}{2}}.$$

Тогда ряд Фурье можно записать в виде:

$$s(t) = \frac{A_0}{2} + \frac{2A_0}{\pi}\left(\cos\left(\frac{2\pi t}{T}\right) - \frac{1}{3}\cos\left(3\frac{2\pi t}{T}\right) + \frac{1}{5}\cos\left(5\frac{2\pi t}{T}\right) - \frac{1}{7}\cos\left(7\frac{2\pi t}{T}\right) \pm \ldots\right).$$

При усечении ряда Фурье до N членов слева и справа от точки разрыва возникают пульсации (рис. 3.2).

На примыкающих к разрыву участках сумма ряда Фурье дает заметные пульсации, причем на графиках рис. 3.2 заметно, что амплитуда этих пульсаций не уменьшается с ростом числа суммируемых гармоник – пульсации лишь сжимаются по горизонтали, приближаясь к точке разрыва. Это явление называется эффектом Гиббса. Амплитуда первого (самого большого) выброса составляет примерно 9 % от величины скачка.

Рис. 3.2. Графическое представление явления Гиббса

3.2. Анализ искажений из-за явления Гиббса

Ограничение числа членов ряда Фурье в разложении функции с разрывами первого рода, вызывающее явление Г иббса, эквивалентно пропусканию меандра (или другой подобной разрывной функции) через идеальный фильтр нижних частот (ФНЧ). В результате становится очевидным, что явление Гиббса можно рассматривать как результат операции свертки меандра $s_1(t)$ с импульсной характеристикой идеального ФНЧ:

$$g_1(t) = \sin(2\pi F_{\text{гр}}t)/\pi t,$$

где $F_{\text{гр}}$ – граничная частота ФНЧ:

$$s_2(t) = s_1(t) * g_1(t).$$

При этом чем больше полоса пропускания $F_{\text{гр}}$, тем выше частота возникающего колебательного процесса в $g_1(t)$ и в выходном сигнале $s_2(t)$ (рис. 3.2).

В большинстве источников, описывающих явление Гиббса, указывается, что колебания в выходном сигнале возникают из-за разрывов в исходном сигнале. Отсюда вытекает, что для предотвращения явления Гиббса следует устранить разрывы во входном сигнале (т. е. сгладить его). Для проверки этого соображения воспользуемся блок-схемой, представленной на рис. 3.3.

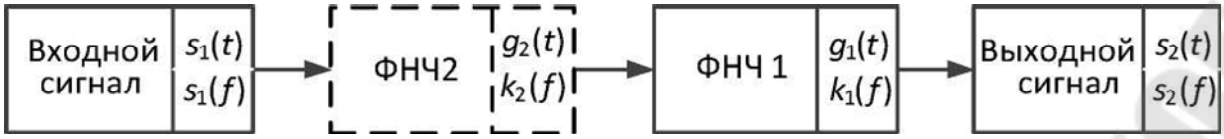


Рис. 3.3. Блок-схема преобразования сигнала с учетом действия фильтров

Здесь ФНЧ 1 – идеальный фильтр нижних частот, действие которого приводит к возникновению явления Гиббса; ФНЧ 2 – предварительный сглаживающий фильтр, который должен устранить разрывы в сигнале, поступающем на вход ФНЧ 1.

В качестве примера рассмотрим действие идеального ФНЧ 2 с полосой пропускания $k_2(f) = nF_{\text{гр}}$, т. е. в n раз более широкой, чем у ФНЧ 1. Под действием ФНЧ 2 функция на входе ФНЧ 1 станет безразрывной. Однако из-за n-кратной разницы в полосах пропускания фильтр ФНЧ 1 просто «не заметит» действия фильтра ФНЧ 2 и того, что входная функция стала безразрывной – результат останется прежним, и явление Гиббса будет таким же как и без предварительного сглаживающего ФНЧ 2.

По-видимому, само явление возникает не только из-за того, что входная функция имеет разрывы. Фактически его причиной является ограничение спектра входного сигнала идеальным фильтром ФНЧ 1, АЧХ которого имеет разрывы, а импульсная характеристика носит колебательный (осциллирующий) характер.

Отсюда следует, что для полного устранения явления Гиббса нужно, чтобы импульсная характеристика фильтра ФНЧ 1 не содержала колебаний. Соответственно, чтобы ослабить явление Гиббса, требуется уменьшение амплитуды колебаний в импульсной характеристике ФНЧ 1. Например, при использовании ФНЧ 1 с безразрывной треугольной АЧХ, имеющей граничную частоту полосы пропускания $2F$, импульсная характеристика будет иметь вид:

$$g_{1\text{тр}}(t) = \sin^2(2\pi Ft)/(\pi t)^2,$$

благодаря чему амплитуда нежелательных колебаний в выходном сигнале существенно ослабляется.

Для полного устранения явления Гиббса необходимо использовать фильтр ФНЧ 1, в импульсной характеристике которого будут полностью отсутствовать колебательные процессы, что накладывает определенные ограничения на крутизну АЧХ этого фильтра.

Изложенные соображения иллюстрирует рис. 3.4, где представлены АЧХ и импульсные характеристики двух видов фильтров:

– идеального фильтра нижних частот с граничной частотой F и колебательной импульсной характеристикой sin(x/.x);

– фильтра нижних частот с косинусквадратичной АЧХ с граничной частотой F (по уровню –6 дБ) и импульсной характеристикой, в которой почти отсутствуют колебания (амплитуда колебаний на порядок ниже, чем у идеального фильтра), благодаря чему явление Гиббса для сигналов с разрывами также ослабляется на порядок.

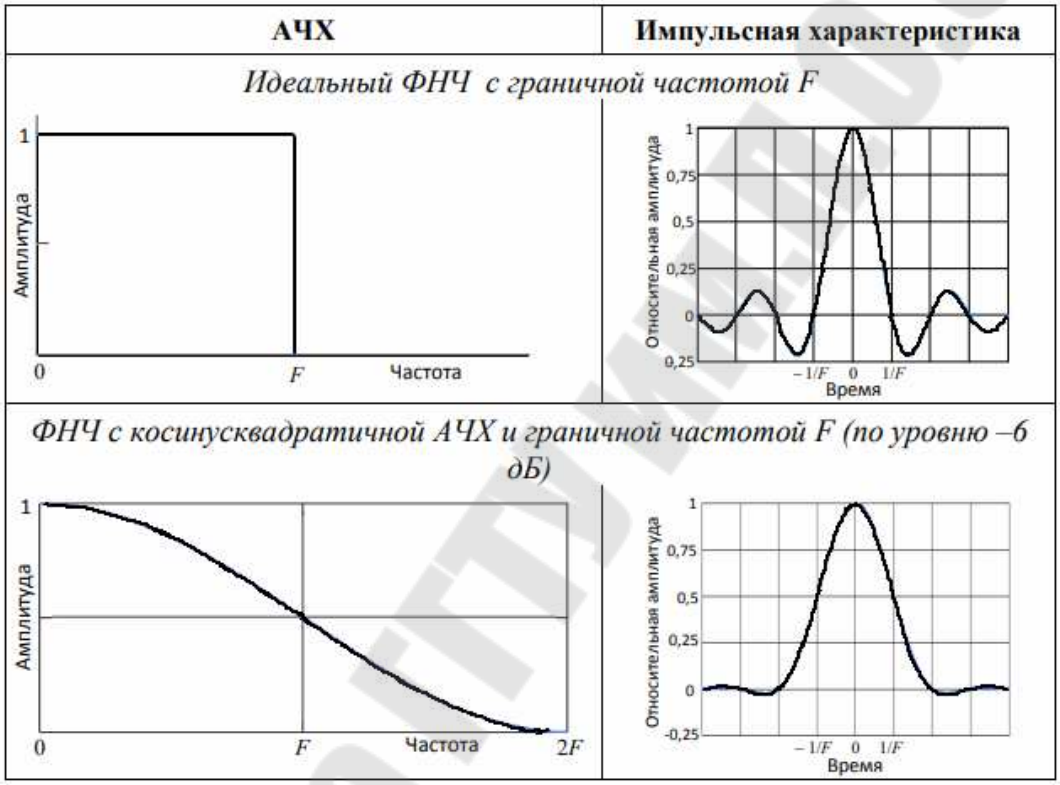


Рис. 3.4. Характеристики идеального и косинусквадратичного фильтра нижних частот

Понятно, что такое ослабление амплитуды колебаний в разрывном сигнале происходит при любой полосе пропускания F фильтра нижних частот с косинусквадратичной АЧХ.

Таким образом, использование для ограничения спектра входной разрывной функции ФНЧ с неколебательной импульсной характеристикой исключает появление колебаний в выходном сигнале, порождающих явление Гиббса. При этом происходит плавное ограничение спектра входной разрывной функции.

Подбор формы характеристик ФНЧ позволяет устанавливать в выходном сигнале любое допустимое значение выбросов в местах разрывов – приблизительно от 9 % (явление Гиббса) до 0.

3.3. Применение оконных функций для устранения явления Гиббса

Альтернативный способ устранения или ослабления явления Гиббса основывается на применении оконных функций. Отметим различия двух методов устранения явления Гиббса.

Действие фильтра нижних частот описывается операцией свертки входного сигнала с (неколебательной) импульсной характеристикой ФНЧ, благодаря чему исходная функция сглаживается и разрывы устраняются. В частотной области это выражается перемножением спектра сигнала и АЧХ фильтра (фурье-образа импульсной характеристики).

Действие оконной (*взвешивающей*) *функции* при спектральном анализе описывается операцией перемножения входного сигнала и оконной функции. В частотной области это отображается сверткой спектра сигнала и спектра (фурье-образа) оконной функции. Нетрудно показать, что применение оконных функций в сигнальной области не приводит к сглаживанию спектра сигнала и не позволяет устранить явление Гиббса.

Устранение (или ослабление) явления Гиббса достигается применением оконных функций к спектру разрывного сигнала (о чем, правда, не всегда четко говорится в литературных источниках). При этом спектр сигнала перемножается с оконной функцией. Но в этом случае оконная функция может рассматриваться как АЧХ некоторого предварительного сглаживающего фильтра, действие которого подробно рассмотрено выше. На рис. 3.5 показаны некоторые известные оконные функции.

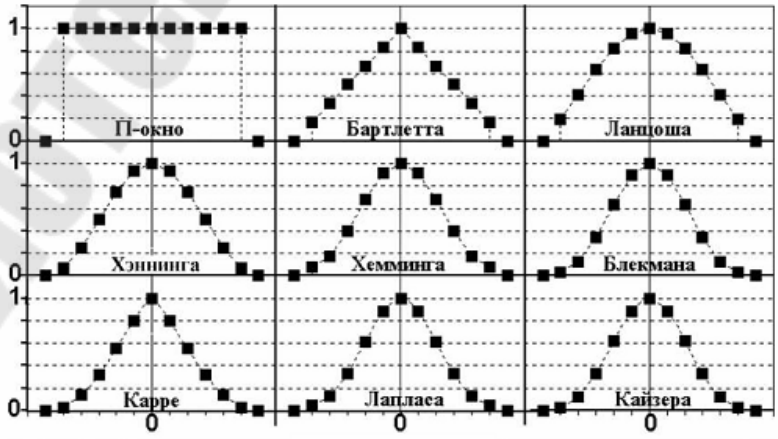


Рис. 3.5. Спектральные оконные функции (дискретизированные)

Явление Гиббса возникает из-за резкого ограничения спектра входной функции, имеющей разрывы первого рода.

С радиотехнической точки зрения явление Гиббса порождается разрывностью амплитудно-частотной характеристики идеального ФНЧ, который описывает операцию ограничения спектра входной разрывной функции (т. е. отбрасывания высокочастотных составляющих ее ряда Фурье).

Для устранения или ослабления явления Гиббса следует ограничивать спектр разрывного сигнала с помощью ФНЧ с импульсной характеристикой, в которой отсутствуют или ослаблены колебательные процессы, т. е. с АЧХ, имеющей достаточно плавный спад. При этом появляется возможность управления интенсивностью колебаний в выходном сигнале, которые в предельном случае использования идеального фильтра низких частот приводят к возникновению явления Гиббса.

Применение для устранения или ослабления явления Гиббса спектральных оконных функций эквивалентно выбору для ограничения спектра разрывных функций фильтра низких частот с неколебательной импульсной характеристикой.

Выполнение лабораторной работы включает следующее:

1) построить графики меандра $s(t)$, разложенного в ряд Фурье для гармоник с номерами

$$k = 1, 2, 3, 4, 10.$$

По времени t взять интервал от $-T$ до $+T$. Значение T задает преподаватель;

2) найти спектр одиночного прямоугольного импульса;

3) задать фильтр с прямоугольной АЧХ с разной полосой пропускания (от одной до восьми гармоник спектра сигнала);

4) вычислить спектр сигнала на выходе фильтра;

5) определить сигнал на выходе фильтра;

6) сравнить сигналы на выходе фильтра с разложением в ряд Фурье (см. п. 1).

Литература

1. Сергиенко, А. Б. Цифровая обработка сигналов / А. Б. Сергиенко. – СПб. : Питер, 2002. – 608 с.

Содержание

Учебное электронное издание комбинированного распространения

ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ

Практикум
по выполнению лабораторных работ
для студентов специальности 1-39 80 03
«Электронные системы и технологии»
дневной и заочной форм обучения

Составители: **Щуплов** Вячеслав Валентинович
Красовская Наталья Александровна

Электронный аналог печатного издания

Редактор *Т. Н. Мисюрова*
Компьютерная верстка *И. П. Минина*

Подписано в печать 22.02.23.
Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».
Ризография. Усл. печ. л. 2,56. Уч.-изд. л. 2,86.
Изд. № 11.
http://www.gstu.by

Издатель и полиграфическое исполнение
Гомельский государственный
технический университет имени П. О. Сухого.
Свидетельство о гос. регистрации в качестве издателя
печатных изданий за № 1/273 от 04.04.2014 г.
пр. Октября, 48, 246746, г. Гомель