Dr. Bob Davidov

Digital filter design in MATLAB

ПРОГРАММНЫЕ СРЕДСТВА ИЗМЕРЕНИЯ И ОБРАБОТКИ СИГНАЛОВ

Фильтрация сигналов. Средства разработки фильтров в MATLAB

Цель работы: Изучить технологии проектирования цифровых фильтров в MATLAB**.**

Задача работы: построить программные коды спроектированных фильтров.

Приборы и принадлежности: Персональный компьютер, MATLAB.

ВВЕДЕНИЕ

MATLAB обладает мощными средствами оперативного проектирования аналоговых и цифровых фильтров, оценки их качества и перевода их в удобные форматы, например, детальные структурные схемы цифровых фильтров, которые легко переводятся в программные коды любого языка программирования. В этой работе представлены средства MATLAB для проектирования фильтров и даны примеры перевода спроектированных фильтров в программные коды.

ОБЩИЕ СВЕДЕНИЯ

Существует большое количество типов фильтров отличающихся по своим характеристикам. Например, характеристики фильтров с одним и тем же числом коэффициентов могут отличаться как показано на рисунке ниже. Средства проектирования позволяют выбрать оптимальный фильтр для решения конкретной задачи.

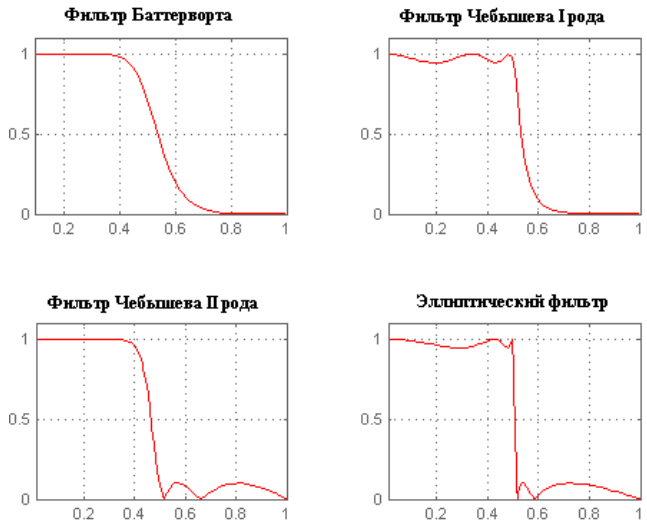


Рис. 1. Сравнительные амплитудные характеристики фильтров.

MATLAB содержит следующие средства для расчета фильтров:

- Набор команд, выполняемых последовательно в m-файле или командном окне.
- Пакет sptool;
- Пакет fdatool. Содержит средства перевода спроектированных фильтров в VHDL код (Меню > Targets > Generate HDL).
- Библиотеки Simulink

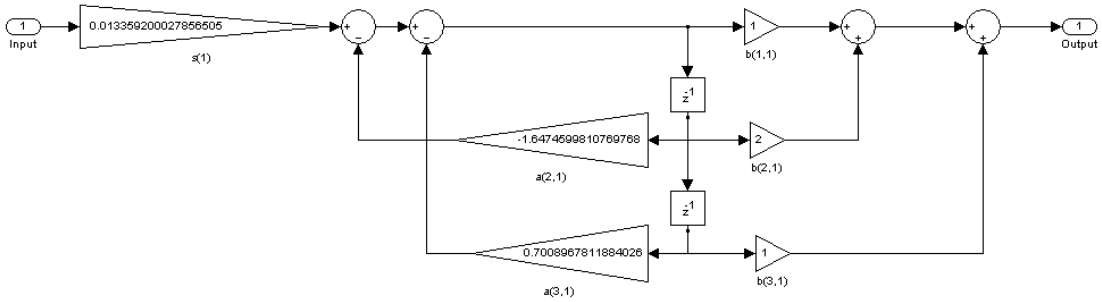


Рис. 1. Пример блок схемы фильтра в Simulink

m-КОМАНДЫ ДЛЯ РАСЧЕТА ФИЛЬТРОВ

Функция fir1 вычисляет цифровые фильтры по методу обратного преобразования Фурье с использованием окон:

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

где

b – вектор n + 1 коэффициентов фильтра $B(z) = b(1) + b(2)z^{-1} + \ldots + b(n+1)z^{-N}$.

n – порядок фильтра (целое четное число);

Wn – вектор относительных частот среза, для многополосного фильтра Wn = [w_1 w_2 w_3 w_4 w_5 ... w_n], $0 < \omega < w_1$, $w_1 < \omega < w_2$, ..., $w_n < \omega < 1$. Частоты среза w_i должны быть меньше единицы – нормализованной частоты дискретизации.

'ftype' – тип фильтра ('low'– ФНЧ; 'high' – ФВЧ; 'bandpass'- полосовой, 'stop' – режекторный;

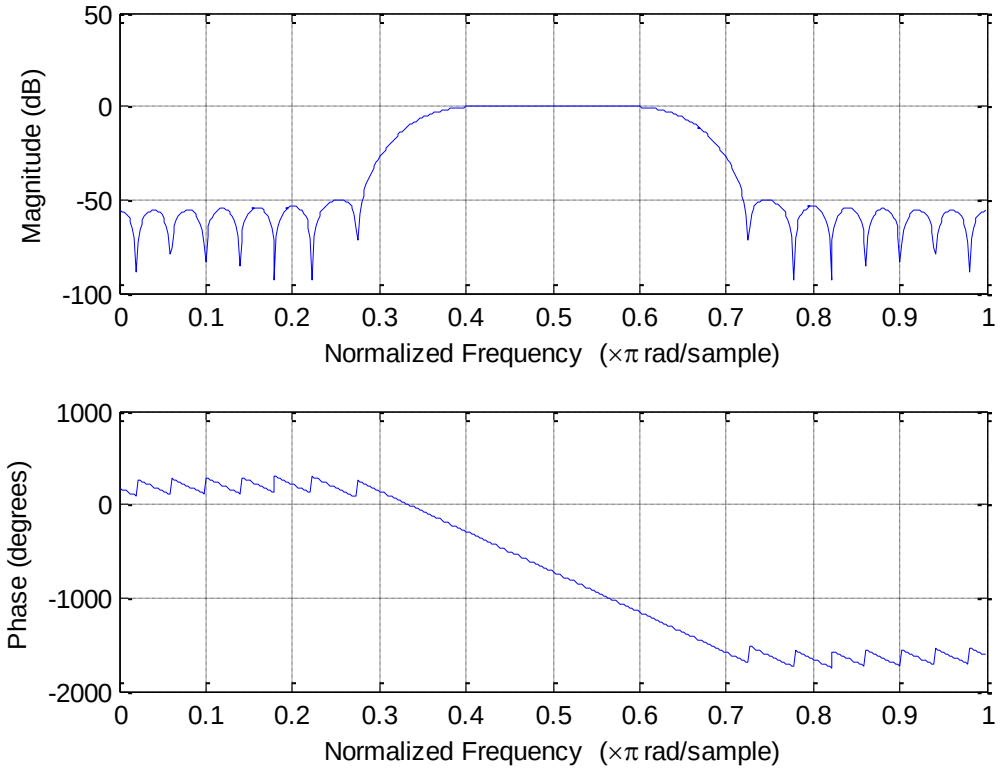


Рис. 2. Пример Амплитудно-Частотной Характеристики (**АЧХ**) и Фазо-Частотной Характеристики (**ФЧХ**) полосового фильтра. >>b = fir1(48,[0.35 0.65]); freqz(b,1,512)

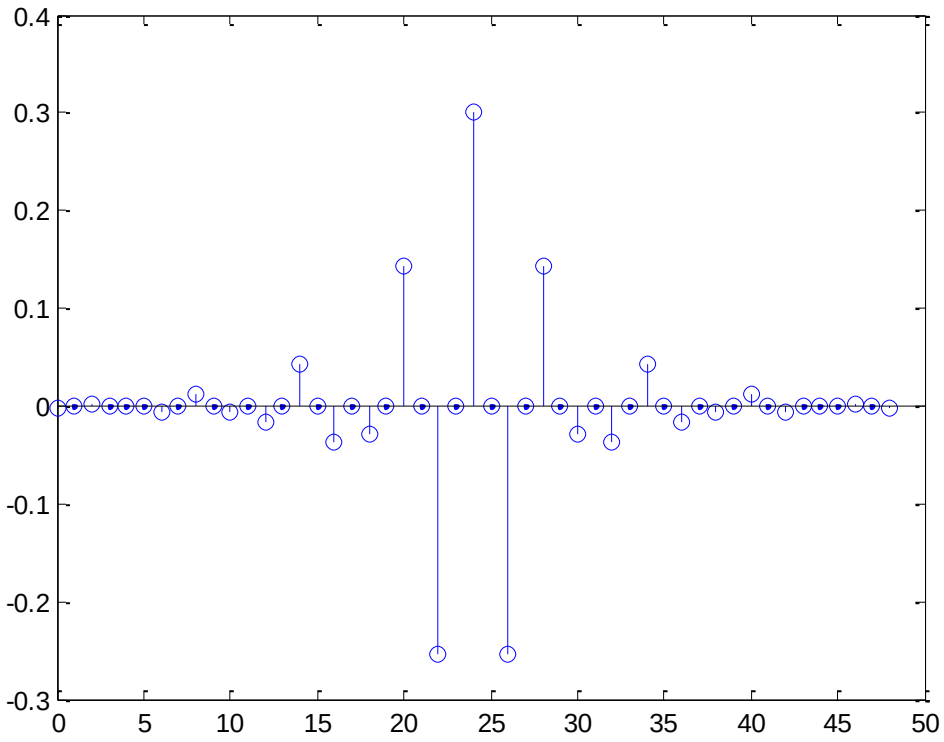
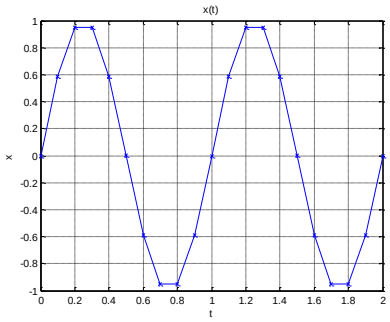
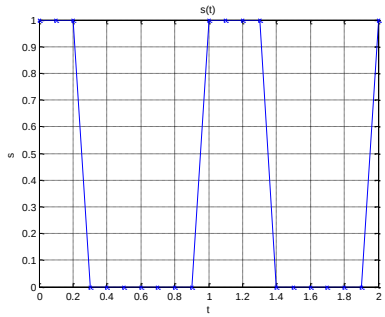
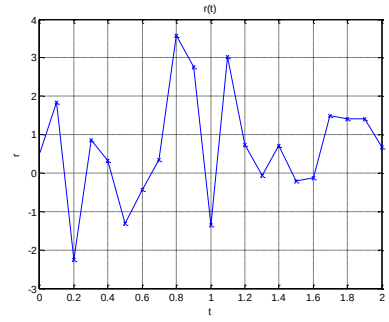


Рис. 3. Пример Импульсной Переходной Характеристики (ИПХ). >>[a,b] = fir1(48,[0.35 0.65]); [h,t] = impz(a,b); stem(t,h).

Таблица 1. Примеры моделирования векторов входного сигнала.

t=0:1/Fd:T; x=sin(2*pi*f0*t);	x(t) – гармонический сигнал частотой f0, длительностью T, с частотой дискретизации Fd	 Для T = 2; Fd = 10; f0 = 1.

t=0:1/Fd:T; s=(square(2*pi*f0*t, f0*tau*100)+1)/2;	s(t) - прямоугольные импульсы единичной высоты, частотой следования f0 , скважностью tau (< периода сигнала = 1/f0), дискретизированные частотой Fd. **Примечание:** Коэффициент 100 в вычислении **s** переводит произведение f0*tau в проценты.	 Для T = 2; Fd = 10; f0 = 1; tau = 0.3;
r=randn(1,length(t));	r(t) - нормальный белый шум длительности T, дискретизированного частотой Fd, первый параметр функции – размерность вектора (1 - скаляр).	 Для T = length(t) = 2;

Реакцию y КИХ фильтра b на входное воздействие x можно вычислить с помощью функции свертки (перемножения полиномов):

y=conv(x, b).

Отклик y КИХ и БИХ фильтров на входное воздействие x можно вычислить с помощью функции

y=filter(b, a, x),

где b и a – коэффициенты числителя и знаменателя передаточной функции фильтра.

Примечание.

- БИХ (IIR) – рекурсивный фильтр (с бесконечной импульсной характеристикой в котором используется обратная связь).
- КИХ (FIR) – нерекурсивный фильтр (с конечной импульсной характеристикой, с отсутствием обратной связи). С какого-то момента времени импульсная характеристика нерекурсивного фильтра становится равной нулю. Главным достоинством КИХ фильтров является то, что они позволяют корректировать АЧХ сигнала, не влияя на его фазу.

Например, фильтрация сигнала x скользящим средним из 5 элементов записывается следующим образом.

windowSize = 5;

y = filter(ones(1,windowSize)/windowSize,1,x)

Пример построения `Filter_PSD` - фильтра низких частот (апериодического звена) в аналитической форме.

```
fs       = param.fmax/param.points;
fs_nrm   = fs/param.bps;
f_nrm    = fs_nrm .* (-param.points/2:param.points/2-1);
s        = f_nrm .* j;
Filter_PSD = (1./(1 + s/FilterParam(1)));
```

где `fmax` – максимальная частота; `points` – длина вектора (количество точек отсчета); `FilterParam(1)` – постоянная времени фильтра.

Расчет фильтрации с использованием функций прямого и обратного преобразования Фурье можно выполнять в следующей последовательности.

1. Перевод вектора входного сигнала из временной области в частотную.
2. Перемножение входного сигнала в частотной области на частотную функцию фильтра
3. Перевод результирующей функции полученной на втором этапе во временную область с целью получения выходного сигнала фильтра ! реакции фильтра на входное воздействие.

Примечание. Пример такого расчета дан ниже в задании 1 и задании 2.

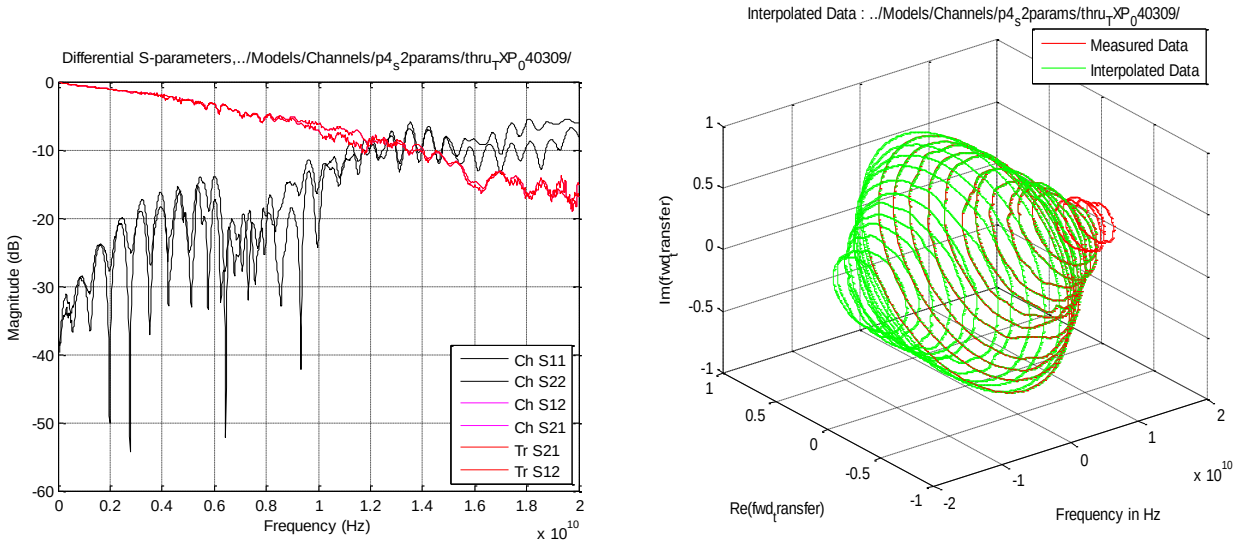


Рис. 4. Пример АЧХ и 3-х мерной АФЧХ в MatLAB

РЕЖЕКТОРНЫЙ ФИЛЬТР

Режекторный фильтр (фильтр пробка от англ. ***notch filter***) не пропускает колебания в узкой полосе частот. Он хорош для подавления резонансов. Фазовая характеристика за пределами полосы подавления фильтра не растет – стремится к нулю с увеличением частоты. Полоса фильтра и величина подавления зависят от ξ:

$$W_{tf}(s) = \frac{1 + 2\xi\omega_0 + {\omega_0}^2}{1 + 2\omega_0 + {\omega_0}^2}$$

Пример амплитудно-частотной характеристики для ω_0 155.9 и ξ = 0.001 и ξ = 0.1 показан на рисунке 5.

```
figure(1)
clf('reset');
w0 = 155.9;  % частота в Гц
xi = 0.1;  % частота в Гц
sys_notch = tf([1 2*xi*w0 w0^2],[1 2*w0 w0^2]) % ПФ фильтра
h = bodeplot(sys_notch);
setoptions(h,'FreqUnits','Hz','Grid','on');
title(sprintf('Notch filter (xi = %4.3f)',xi));
```

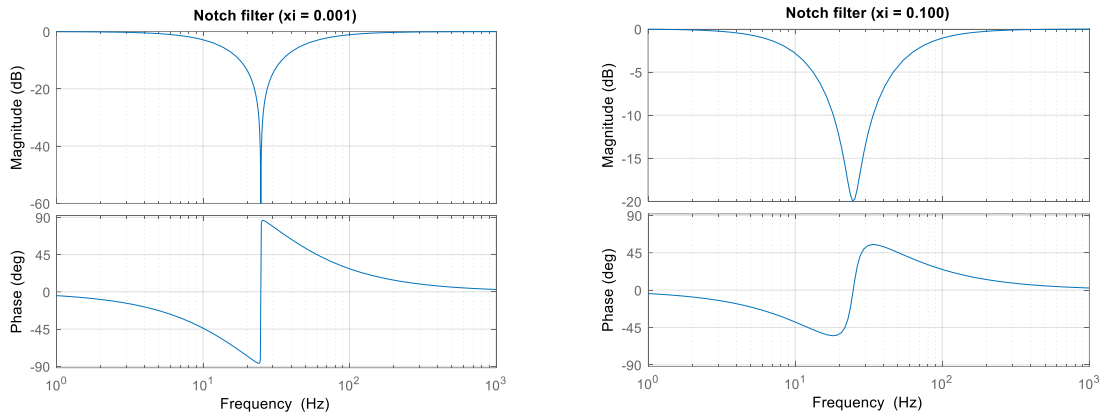


Рис. 5. Амплитудно-частотная характеристика режекторного фильтра $W_{tf}(s) = \frac{1+2\xi\omega_0+{\omega_0}^2}{1+2\omega_0+{\omega_0}^2}$ для ω_0 =155.9 Гц и ξ = 0.001 (слева) и ξ = 0.1 (справа).

РАСЧЕТ ФИЛЬТРА С ПОМОЩЬЮ ПАКЕТА SPTOOL

SPTool – это графический интерфейс пользователя (GUI), который управляет четырьмя другими графическими интерфейсами:

- Signal Browser,
- Filter Design and Analysis Tool,
- FVTool, и
- Spectrum Viewer.

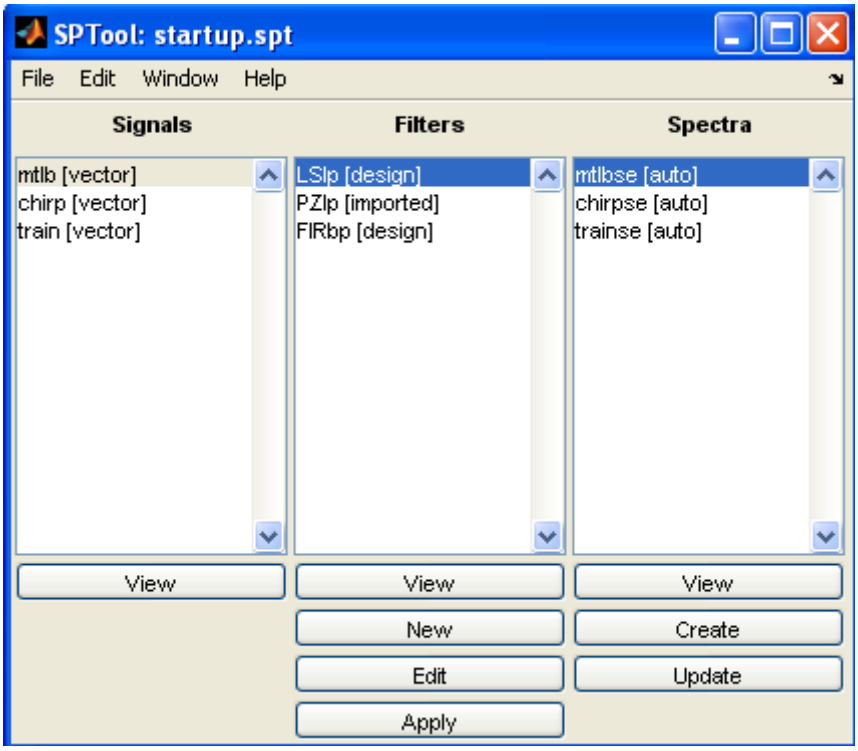


Рис. 6. Внешний вид окна SPTool которое открывается командой sptool в командном окне.

Эти графические интерфейсы обеспечивают доступ ко многим сигналам, фильтрам и функциям спектрального анализа в toolbox.

При помощи SPTool можно

- Анализировать сигналы списка Signals
- Проектировать или редактировать фильтры при помощи Filter Design and Analysis Tool (включая редактор нулей и полюсов Pole/Zero Editor)
- Анализировать реакцию фильтров, входящих в список Filters при помощи FVTool
- Добавлять сигналы и фильтры в соответствующие списки
- Создавать и анализировать спектры сигналов при помощи Spectrum Viewer

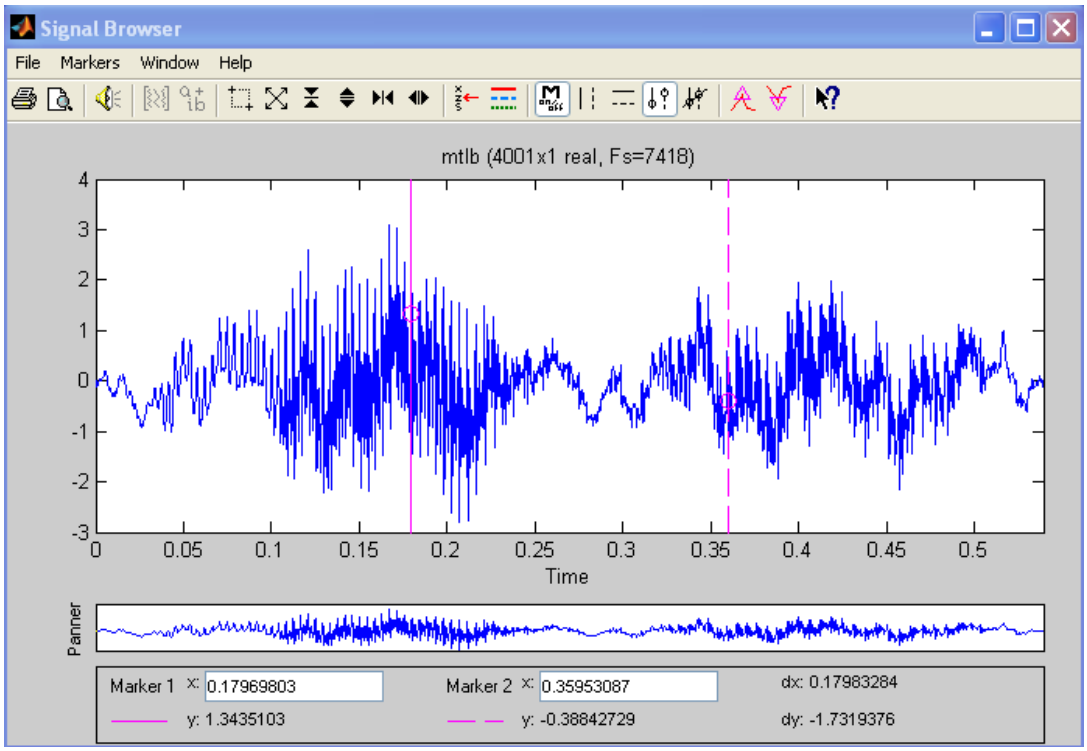


Рис. 7. Окно Signal Browser открывается командой View колонки Signals

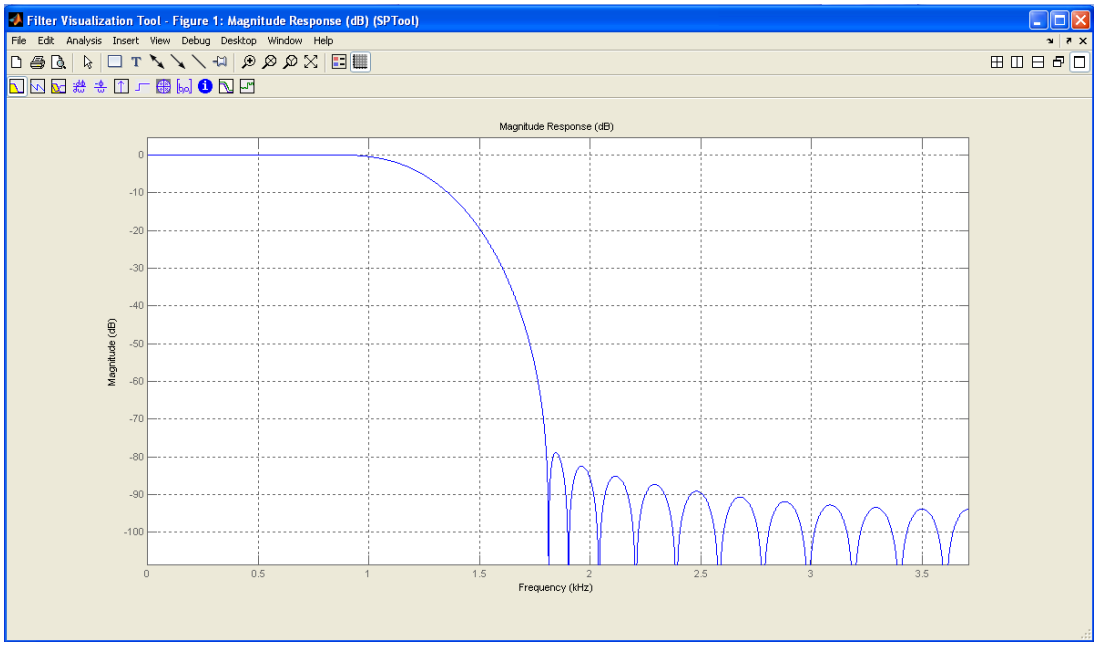


Рис. 8. Окно Filter Visualization Tool открывается командой View колонки Filters. FVTool позволяет просматривать характеристики проектируемого или импортированного фильтра, включая его АЧХ, фазовую характеристику, задержки, расположения полюсов и нулей, реакции на импульсные и скачкообразные воздействия.

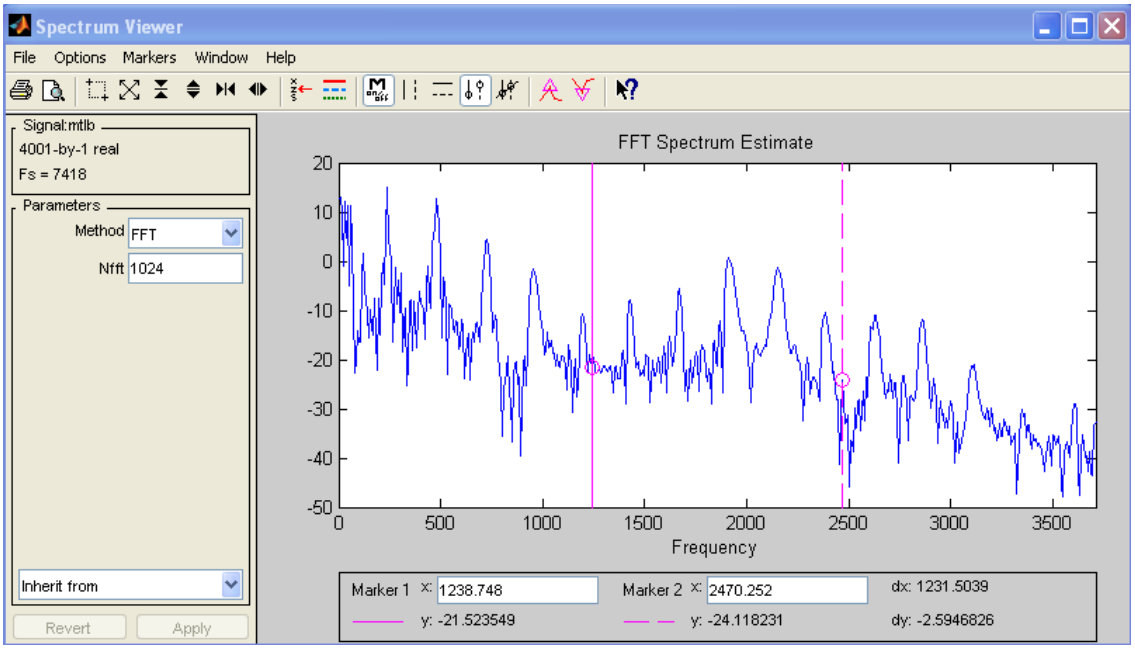


Рис. 9. Окно Spectrum Viewer открывается командой View списка Spectra

Пакет Filter Design and Analysis вызывается нажатием кнопок New или Edit колонки Filters окна пакета SPTool.

Пакет sptool позволяет моделировать процесс фильтрации с помощью рассчитанного фильтра. Для этого в среду пакета sptool нужно импортировать входной сигнал (из WorkSpace), созданный в рабочем окне программы Matlab. Фильтрация происходит после нажатия кнопки Apply.

РАСЧЕТ ФИЛЬТРА С ПОМОЩЬЮ ПАКЕТА FDATOOL

Пакет Filter Design and Analysis (FDATOOL) позволяет выполнять

- Расчет цифровых фильтров
- Анализ фильтров
- Редактирование фильтров
- Экспортировать параметры фильтра в Simulink модель

Пакет Filter Design and Analysis (FDATOOL) вызывается командой fdatool через командное окно или из пакета SPTool кнопками New или Edit колонки Filters окна пакета SPTool.

Параметры фильтра задаются в окне «Filter Designer», например, как показано на **Рис. 10**.

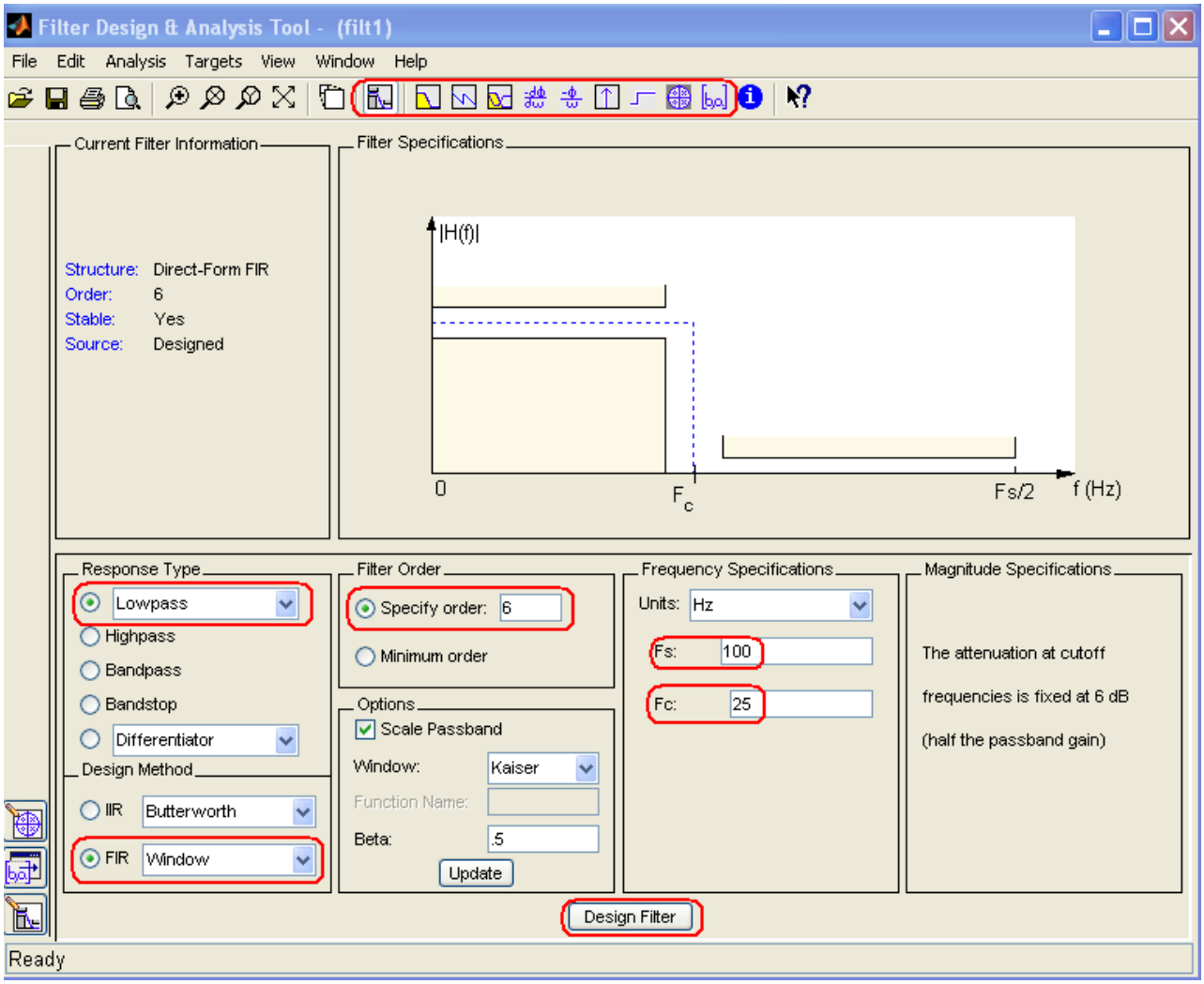


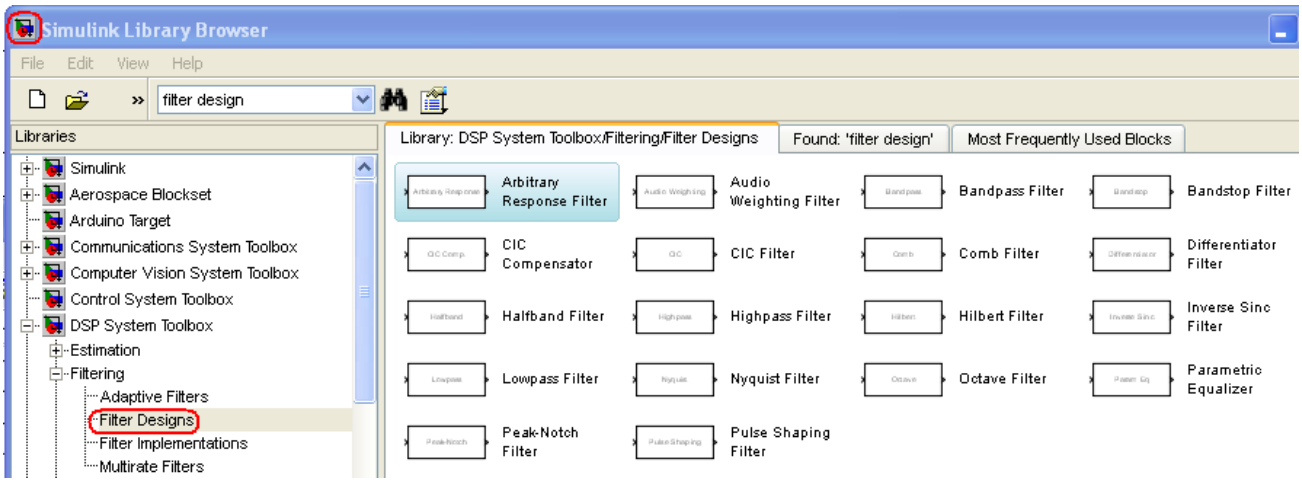
Рис. 10. Пример набора входных параметров фильтра.

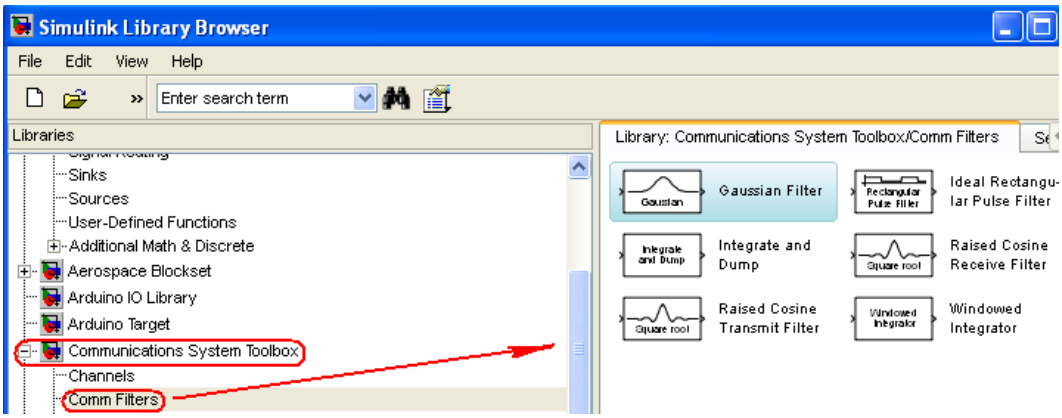
Командами Menu > View > Filter Visualization Tool выводятся АЧХ, ФЧХ, ИПХ, задержка, импульсная характеристика, нули и полюса в Z-плоскости, значения коэффициентов фильтра

РАСЧЕТ ФИЛЬТРА СРЕДСТВАМИ SIMULINK

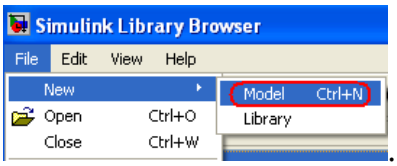
Simulink предлагает следующие средства проектирования фильтров

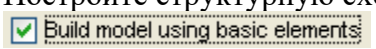
Модули раздела Filter Design можно найти браузером библиотеки Simulink





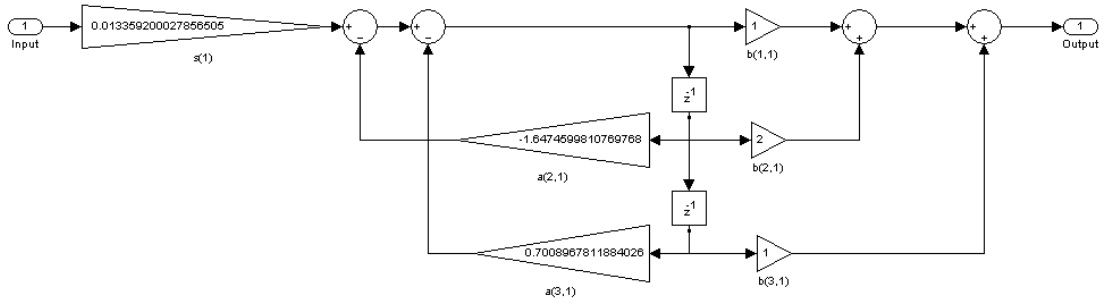
Для оперативного построения структуры цифрового фильтра в Simulink и его программной реализации сделайте следующее.

1. Загрузите MatLAB (проверено в R2007a, version 7.4.).
2. Настройте MatLAB на свой рабочий каталог.
3. В командном окне запустите Simulink.
4. Откройте редактор Simulink 

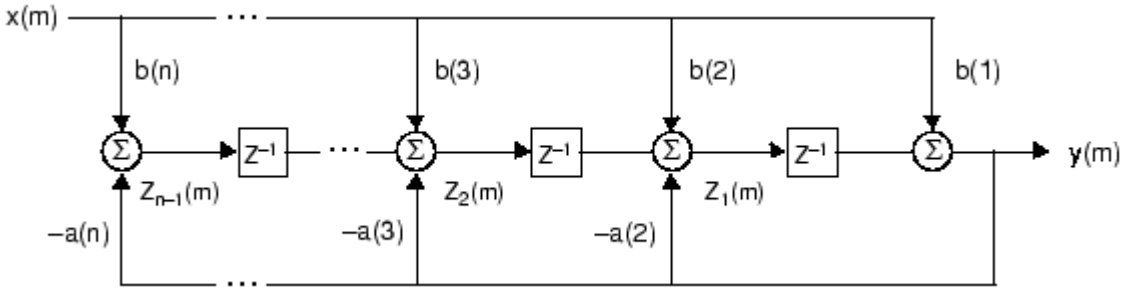
.
5. Скопируйте в рабочее окно модели блок: Communications Blockset > Comm Filters > Filter Designs Library Link > Filter Design > ***Digital Filter Design.***
6. Раскройте блок и постройте требуемый фильтр.
7. Постройте структурную схему фильтра: Меню > File > Export to Simulink model > 

> Realize Model.

8. Раскройте новый блок появившийся в окне модели. Он содержит схему фильтра с информацией достаточной для программирования фильтра на любом языке программирования, например,



Функция построенного фильтра представляет собой структуру



или выражение

y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb) - a(2)*y(n-1) - ... - a(na+1)*y(n-na),

в котором n-1 - порядок фильтра.

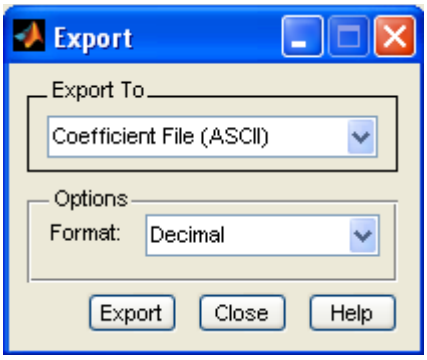
Во временной области, передаточная функция описывается системой уравнений:

$$
\begin{aligned}
y(m) &= b(1)x(m) + z_1(m-1) \\
z_1(m) &= b(2)x(m) + z_2(m-1) - a(2)y(m) \\
\vdots \quad &= \quad \vdots \qquad\qquad \vdots \\
z_{n-2}(m) &= b(n-1)x(m) + z_{n-1}(m-1) - a(n-1)y(m) \\
z_{n-1}(m) &= b(n)x(m) - a(n)y(m)
\end{aligned}
$$

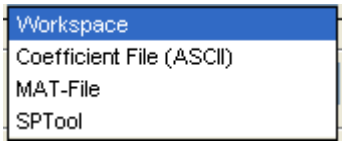
Через отношение входа к выходу передаточная функция выглядит как

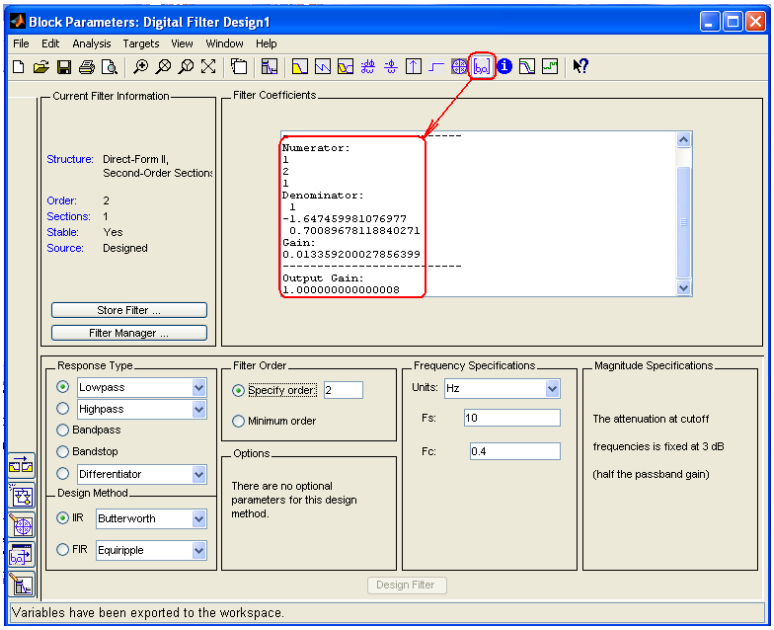
$$Y(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(nb+1)z^{-nb}}{1 + a(2)z^{-1} + \ldots + a(na+1)z^{-na}} X(z)$$

Примечание. Коэффициенты спроектированного фильтра можно получить следующим образом

- Меню Digital Filter Design > File > Export > 


```
SOS matrix:
1   2   1   1   -1.647459981076977   0.70089678118840271

Scale Values:
0.013359200027856399
1.000000000000008
```

- Меню Digital Filter Design > File > Export > 

- Из экспортированной в Simulink model
- Из Digital Filter Design: 

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1. Построение фильтров с использованием языка программирования MATLAB.

1. Используя следующий код постройте амплитудные характеристики фильтров 'ideal'; 'singlepole'; 'twopole'; 'raisedcosine'; 'bessel'; 'gaussian'; и трехмерную амплитудно!фазовую характеристику.

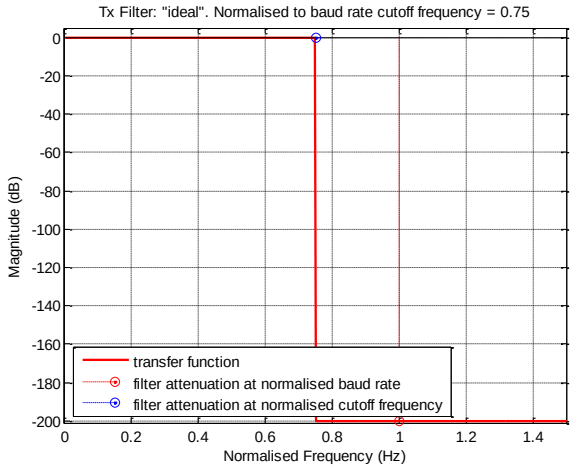
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% filter.m      v1.0a
% Matlab 7.4.(R2007a), R2012(7.14)
% Bob Davidov
% 16 September 2013
% %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
param.points = 2^13;
param.bps = 1.03125e10;
param.fmax = 3.0938e+010
FilterParam = [0.7500  0.7500];
name = 'Tx' % name of filter
Filter_type = 'ideal';
Filter_type = 'singlepole';
 Filter_type = 'twopole';
 Filter_type = 'raisedcosine';
 Filter_type = 'bessel';
 %Filter_type = 'gaussian';

fs      = param.fmax/param.points;
fs_nrm  = fs/param.bps;
f_nrm   = fs_nrm .* (-param.points/2:param.points/2-1);
s       = f_nrm .* j;

if strcmp(Filter_type, 'ideal')

   N = size(find(abs(f_nrm)<=FilterParam(1)),2);

   if N == param.points
      Filter_PSD(1:param.points) = 1;
   else
      % attenuation
      idata.abs(1:param.points) = 10^(-200/20);
      idata.abs(find(abs(f_nrm)<=FilterParam(1))) = 1;

      % phase shift from edge to edge of pi radians over a frequency shift
      NN = param.points/2 - round(N/2) + 1;

      idata.phase(1:NN) = pi/2;
      d_phase = pi/(N+1);
      for i = 1:N
         idata.phase(NN + i) = idata.phase(NN + i -1) - d_phase;
      end
      idata.phase(NN+i+1:param.points) = -pi/2;

      idata.s = idata.abs .* exp(+j.*idata.phase);
      Filter_PSD = real(idata.s) + j*imag(idata.s);
   end
```

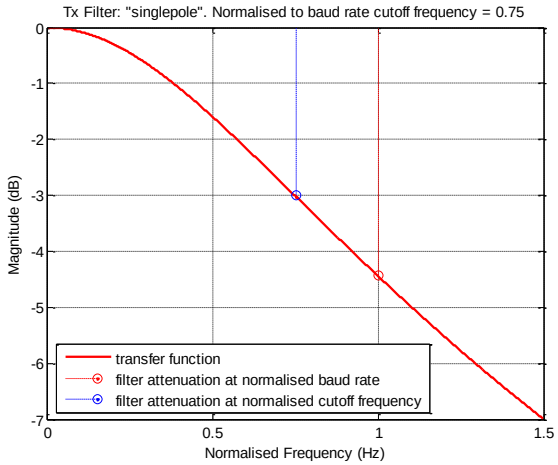
```
end

if strcmp(Filter_type, 'singlepole')
   Filter_PSD = (1./(1 + s/FilterParam(1)));
end

if strcmp(Filter_type, 'twopole')
   Filter_PSD = (1 ./(1 + s/FilterParam(1)) ./ (1 + s/FilterParam(2)) );
end

if  strcmp(Filter_type, 'raisedcosine')
   % the roll-off factor "alpha" is defined by param.TxFilter_RollOffFactor
   % and param.RxFilter_RollOffFactor
   alpha = 0.35; % optimised?
   T     = 1;
   r1 = (abs(f_nrm ) <= (1-alpha)/(2*T));
   r2 = (abs(f_nrm ) >  (1-alpha)/(2*T)) .* (abs(f_nrm )<(1+alpha)/(2*T));
   Filter_PSD =              r1 * T;
   Filter_PSD = Filter_PSD + r2 .* T/2.*(1 + cos(pi.*T./alpha.*(abs(f_nrm)-(1-alpha)/(2*T))));
   Filter_PSD = Filter_PSD + 10^(-200/20);
end

if strcmp(Filter_type, 'bessel')
   BesselFilterOrder = 4;
   n = 4; % Bessel Filter Order, optimised?;
   w = FilterParam(1);
   [b,a] = besself(n, w);
   h = freqs(b,a,f_nrm(param.points/2 + 1:param.points));
   idata.abs   = interp1(f_nrm(param.points/2 + 1:param.points), abs(h),   abs(f_nrm), 'linear', 'extrap');
   idata.phase = interp1(f_nrm(param.points/2 + 1:param.points), unwrap(angle(h)), abs(f_nrm), 'linear', 'extrap');
   % correct for negative frequencies
   idata.s = idata.abs .* exp(+j.*idata.phase);
   Filter_PSD = real(idata.s) + j*imag(idata.s) .* sign(f_nrm);
end

if strcmp(Filter_type, 'gaussian')
   % Gaussian filter is defined to have 50% transmission at the cutoff frequency
   Filter_PSD = exp(log(2) .*(s ./FilterParam(1)).^2 );
end

if 1
   figure
   plot_handles_Filter = plot(f_nrm(param.points/2 + 1:param.points), 20*log10(abs(Filter_PSD(param.points/2 + 1:param.points))), 'r',
'linewidth', 2);
   hold on
   if strcmp(Filter_type, 'ideal') || strcmp(Filter_type, 'singlepole') || strcmp(Filter_type, 'bessel')
      stem_handles_br = stem(1, 20*log10(abs(Filter_PSD(max(find(f_nrm < 1))))), '-.ro');
      hold on
      if (FilterParam(1) <= 1)
         stem_handles_c = stem(FilterParam(1), 20*log10(abs(Filter_PSD(max(find(f_nrm < FilterParam(1)))))), '-.bo');
         legend_handles = [plot_handles_Filter, stem_handles_br(1), stem_handles_c(1)];
         legend(legend_handles, 'transfer function', 'filter attenuation at normalised baud rate', 'filter attenuation at normalised cutoff
frequency', 3);
      else
         legend_handles = [plot_handles_Filter, stem_handles_br(1)];
         legend(legend_handles, 'transfer function', 'filter attenuation at normalised baud rate', 3);
```

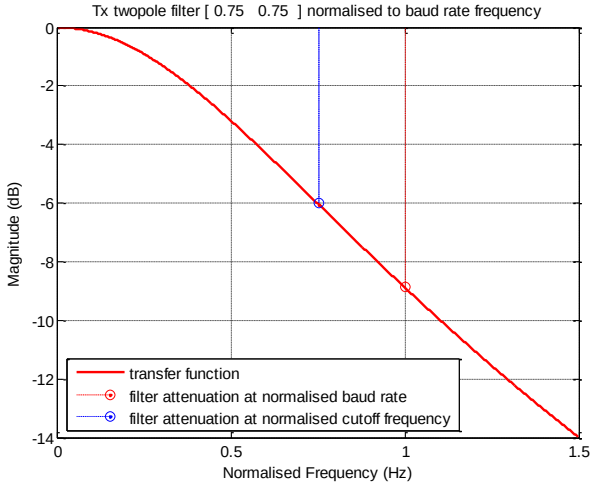
```
        end

        if strcmp(Filter_type, 'ideal') || strcmp(Filter_type, 'singlepole')
            title(sprintf('%s Filter: "%s". Normalised to baud rate cutoff frequency = %3.2f', name, Filter_type, FilterParam(1)));
        else
            title(sprintf('%s Filter: "%s, %d order". Normalised to baud rate cutoff frequency = %3.2f', name, Filter_type, BesselFilterOrder,
FilterParam(1)));
        end
    end
    if strcmp(Filter_type, 'twopole')
        stem_handles_br = stem(1, 20*log10(abs(Filter_PSD(max(find(f_nrm < 1))))), '-.ro');
        hold on
        if (FilterParam(1) <= 2) &&  (FilterParam(2) <= 2)
            stem_handles_c = stem(FilterParam, [20*log10(abs(Filter_PSD(max(find(f_nrm < FilterParam(1)))))
20*log10(abs(Filter_PSD(max(find(f_nrm < FilterParam(2))))))], '-.bo');
            title([sprintf('%s twopole filter [', name)  sprintf(' %3.2f  ', FilterParam) '] normalised to baud rate frequency']);
            legend_handles = [plot_handles_Filter, stem_handles_br(1), stem_handles_c(1)];
            legend(legend_handles, 'transfer function', 'filter attenuation at normalised baud rate', 'filter attenuation at normalised cutoff
frequency', 3);
        else
            title([sprintf('%s twopole filter [', name)  sprintf(' %3.2f  ', FilterParam) '] normalised to baud rate frequency']);
            legend_handles = [plot_handles_Filter, stem_handles_br(1)];
            legend(legend_handles, 'transfer function', 'filter attenuation at normalised baud rate', 3);
        end
    end

    if strcmp(Filter_type, 'raisedcosine')
        stem_handles_6dB = stem(0.5, 20*log10(abs(Filter_PSD(max(find(f_nrm < 0.5))))), '-.bo');
        stem_handles_br = stem(1, 20*log10(abs(Filter_PSD(max(find(f_nrm < 1))))), '-.ro');
        hold on
        title(sprintf('%s Filter: "%s". Normalised to baud rate cutoff frequency = 1', name, Filter_type));
        legend_handles = [plot_handles_Filter, stem_handles_6dB(1), stem_handles_br(1)];
        legend(legend_handles, 'transfer function', '6dB attenuation at half normalised cutoff frequency', 'filter attenuation at normalised
cutoff frequency', 3);
    end

    if strcmp(Filter_type, 'gaussian')
        stem_handles_6dB = stem(FilterParam(1), 20*log10(abs(Filter_PSD(max(find(f_nrm < FilterParam(1)))))), '-.bo');
        stem_handles_br = stem(1, 20*log10(abs(Filter_PSD(max(find(f_nrm < 1))))), '-.ro');
        hold on
        title(sprintf('%s Filter: "%s". Normalised to baud rate cutoff frequency = 1', name, Filter_type));
        legend_handles = [plot_handles_Filter, stem_handles_6dB(1), stem_handles_br(1)];
        legend(legend_handles, 'transfer function', '6dB attenuation at cutoff freq: xxFilterParam(1)', 'filter attenuation at normalised
cutoff frequency', 3);
    end

    xlabel('Normalised Frequency (Hz)');
    ylabel('Magnitude (dB)');

    if strcmp(Filter_type, 'ideal')
        axis([0 max(f_nrm) -201 5]);
    end
    grid
end

% plot resulting real vs imaginary plots to check data
```

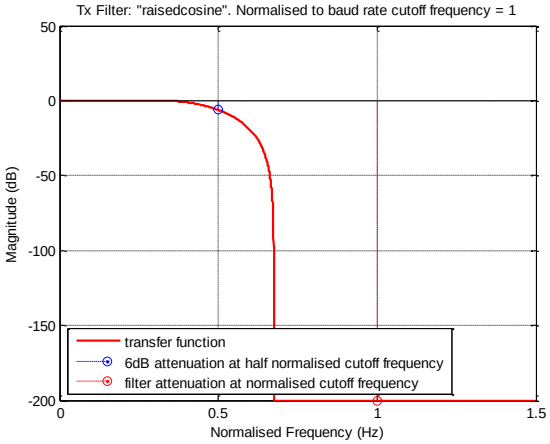
```
if 1
    figure
    plot3(f_nrm, real(Filter_PSD), imag(Filter_PSD),'r');
    hold on
    grid on
    xlabel('Normalised Frequency');
    ylabel('Re(FTF)');
    zlabel('Im(FTF)');
    title(['3d FTF']);
end

%End of filter.m
```

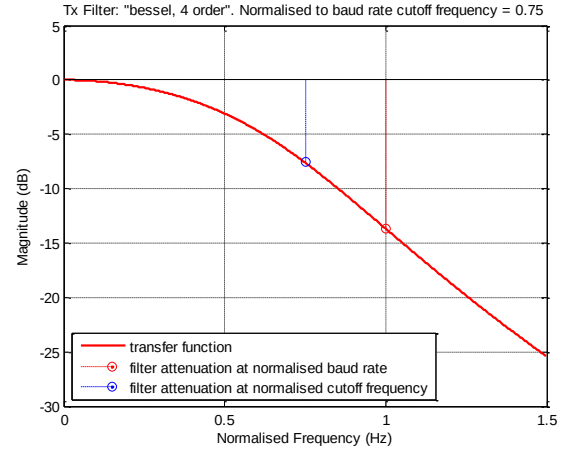
2. Сопоставьте полученные результаты со следующими характеристиками фильтров.

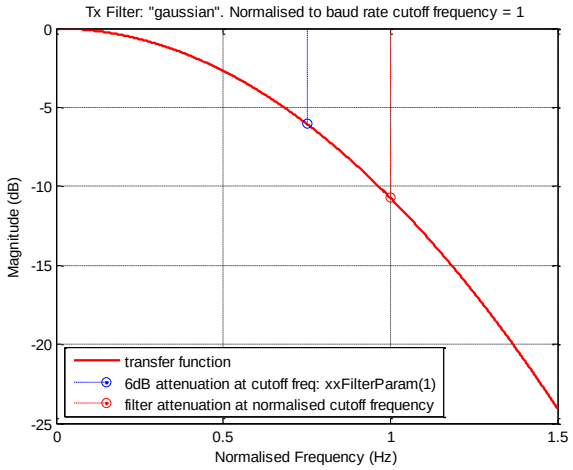


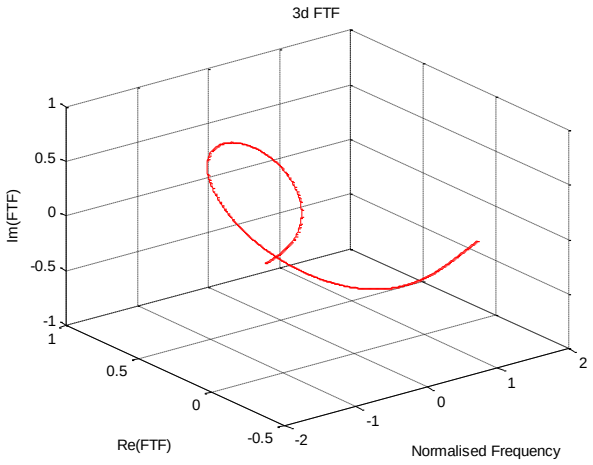


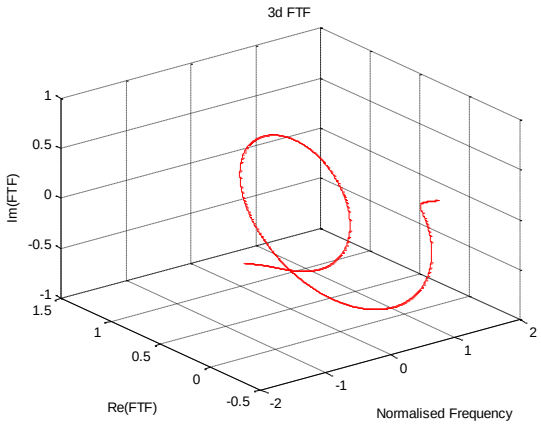












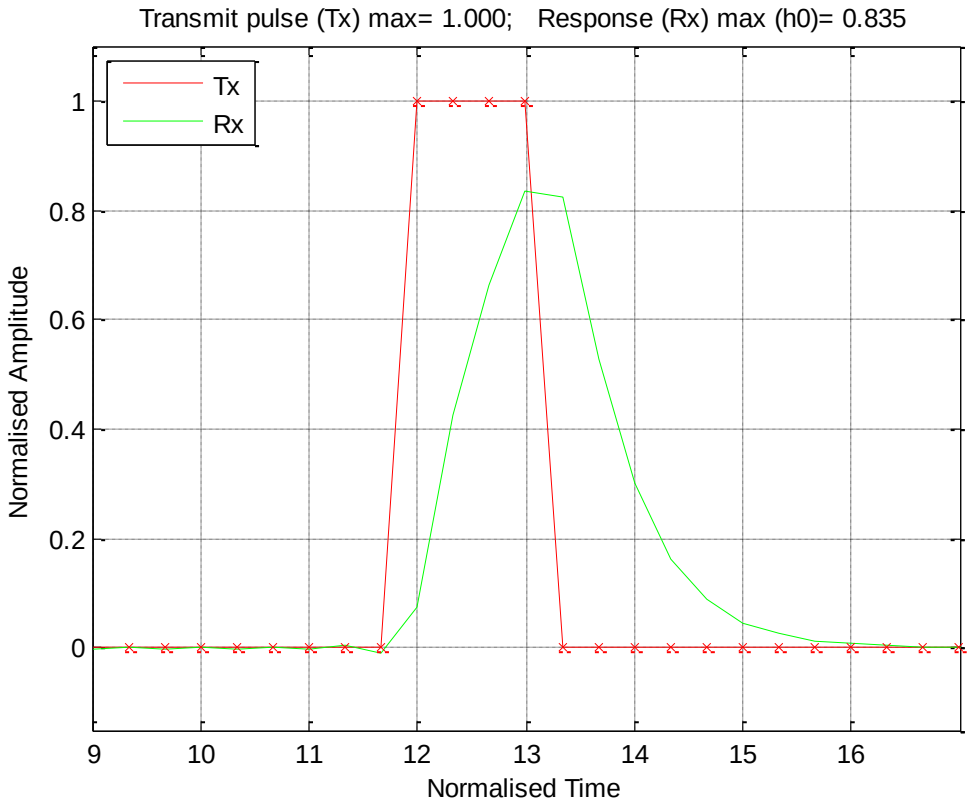
Задание 2. Фильтрация сигнала с использованием языка программирования MatLAB.

1. Используя следующий код постройте вектор импульсного сигнала, затем переведите сигнал в частотную область, найдите совместную частотную характеристику сигнала и фильтра, переведите результирующий сигнал во временную область.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% filter_response.m      v1.0a
% Matlab 7.4.(R2007a), R2012(7.14)
% Bob Davidov
% 16 September 2013
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all

param.bps = 1.03125e10;
param.fmax = 3.0938e+010;
param.points = 2^13;
Ts_nrm = param.bps /param.fmax;
time = Ts_nrm .* (1:param.points);
```

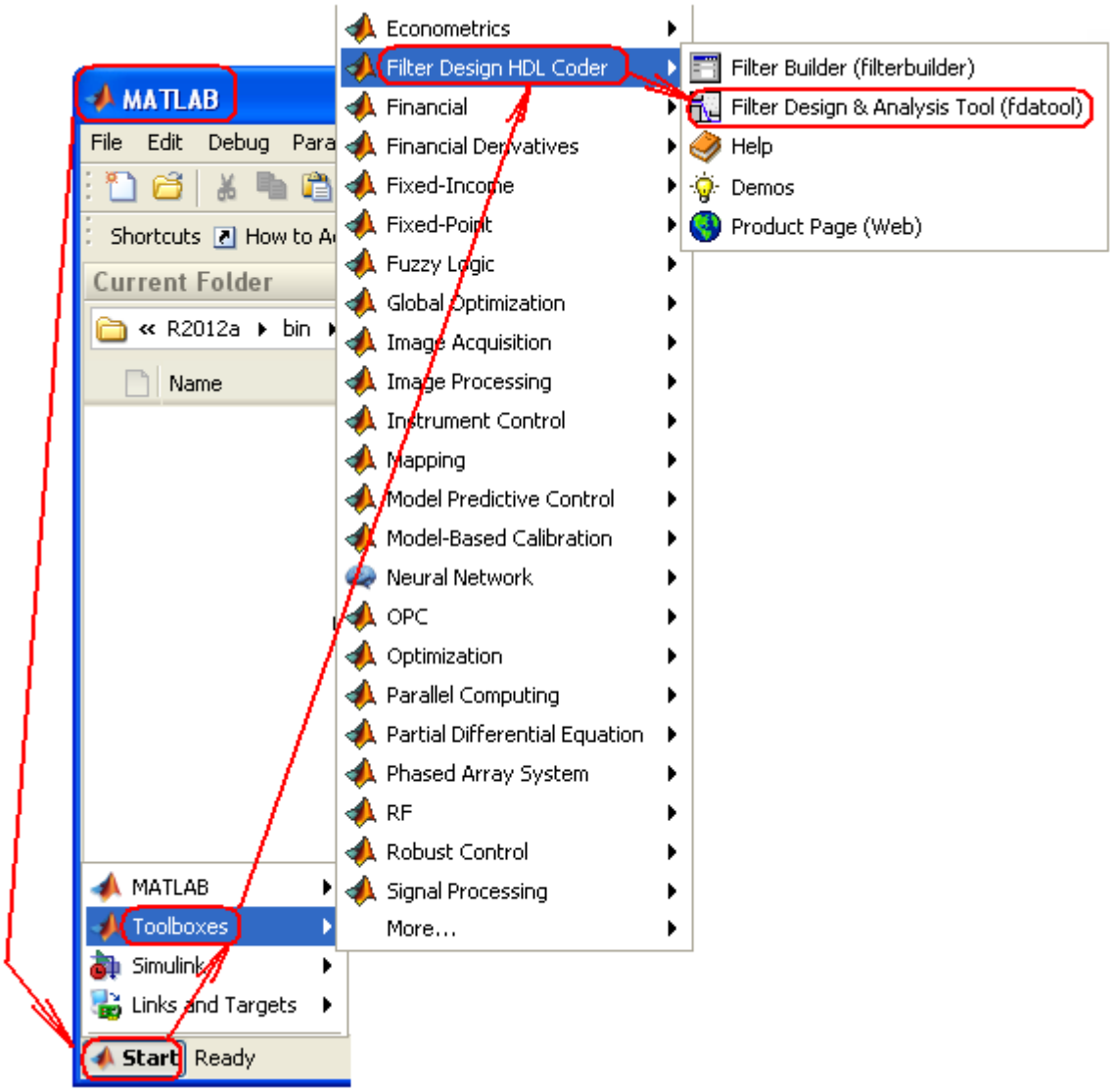
```
% create ideal transmit pulse
iSignal.Tx(1:size(time,2)) = 0;
t0 =    max(find(time<=12.0001));
t1  =   max(find(time<12.9999));
iSignal.Tx(t0:t1)  = 1.0;

% transmit input pule from time into frequency domain
TransFunction.shiftedPSD = fft(iSignal.Tx);
TransFunction.PSD        = fftshift(TransFunction.shiftedPSD);

% set two poles filter;
FilterParam = [0.7500  0.300];
fs     = param.fmax/param.points;
fs_nrm = fs/param.bps;
f_nrm  = fs_nrm .* (-param.points/2:param.points/2-1);
s      = f_nrm .* j;
Filter.PSD_Tx = (1 ./(1 + s/FilterParam(1)) ./ (1 + s/FilterParam(2)));

% filter response in freq. domain
TransFunction.PSD        = TransFunction.PSD  .* Filter.PSD_Tx;
TransFunction.shiftedPSD  = ifftshift(TransFunction.PSD);

% filter response in time domain
iSignal.Rx = real(ifft(TransFunction.shiftedPSD));

% display input and pulse response
figure
plot(time, iSignal.Tx,'r');
hold on
[max_Tx, time_maxTx] = max(iSignal.Tx);
[min_Tx, time_minTx] = min(iSignal.Tx);
[max_Rx, time_maxRx] = max(iSignal.Rx);
dtime_p5= round((time_maxRx - time_maxTx)*time(1) -1);
plot(time - dtime_p5, iSignal.Rx,'g');
hold on
plot(time, iSignal.Tx,'rx');
axis([(time_maxTx*time(1) - 3) (time_maxTx*time(1) + 5)  (min_Tx-0.15) (max_Tx+0.1)])
grid on
legend('Tx','Rx', 2);
xlabel('Normalised Time');
ylabel('Normalised Amplitude');
title(sprintf('Transmit pulse (Tx) max= %4.3f;   Response (Rx) max (h0)= %4.3f', max(iSignal.Tx), max(iSignal.Rx)));

% End of filter_response.m
```

2. Сопоставьте полученные результаты со следующими графиками.

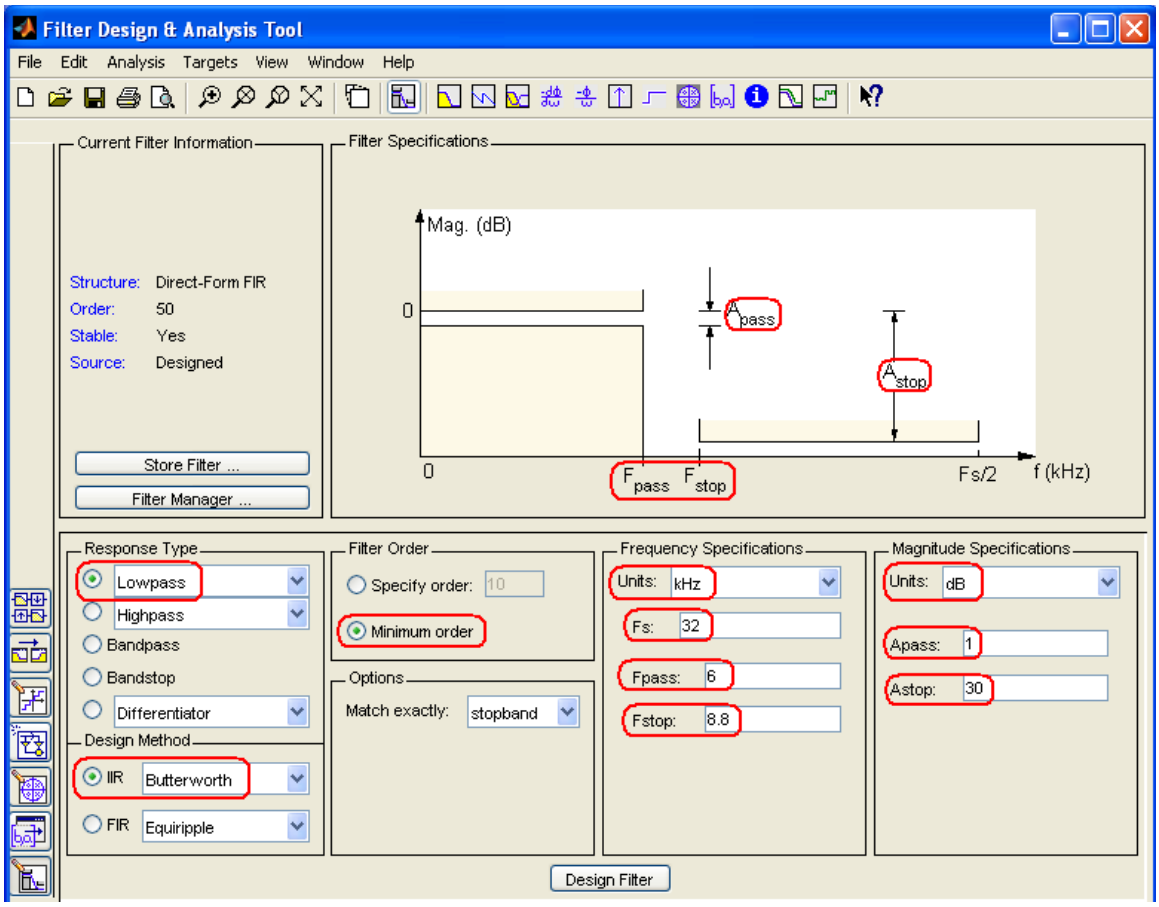


Задание 3. Синтезировать БИХ фильтр нижних частот.

1. Запустите FDAToolbox:



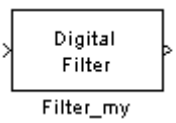
2. Выберите тип фильтра – Lowpass;
3. Выберите метод синтеза – IIR, в качестве прототипа - фильтр Butterworth;
4. Выберите порядок фильтра: Minimum order (минимальный порядок);
5. Задайте следующие параметры фильтра.
   - Частота дискретизации Fs= 32 кГц
   - Частота среза Fpass = 6 кГц
   - Частота полосы задержки = 8,8 кГц
   - Пульсации (неравномерность) в полосе пропускания Apass= 1 дБ
   - Ослабление в полосе подавления Astop= 30 дБ



6. Нажмите кнопку Design Filter  .
7. Используя средства анализа fdatool получите: АЧХ  и ЛАЧХ  , ФЧХ  , Импульсную характеристику  , переходную характеристику  , нули и полюса  .
8. Является ли фильтр устойчивым ?

Задание 4. Получение структурной схемы фильтра.

1. Нажмите на кнопку Realise Model  . Данная функция позволяет синтезировать имитационную модель фильтра в Simulink
2. Задайте требуемые параметры
3. Нажмите кнопку Realise Model  .

В результате в окне Simulink создается блок с параметрами фильтра 

Задание 5. Фильтрация речевого сигнала в среде Simulink

1. Постройте модель, показанную на рисунке. Модель принимает сигнал из WorkSpace и выводит сигнал на звуковую плату компьютера.
2. Получите качественный звуковой сигнал меняя характеристики фильтра и сравнивая звучание оригинального (входного) и отфильтрованного сигналов.

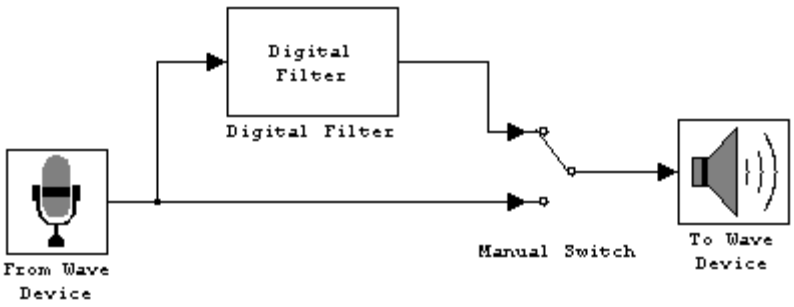
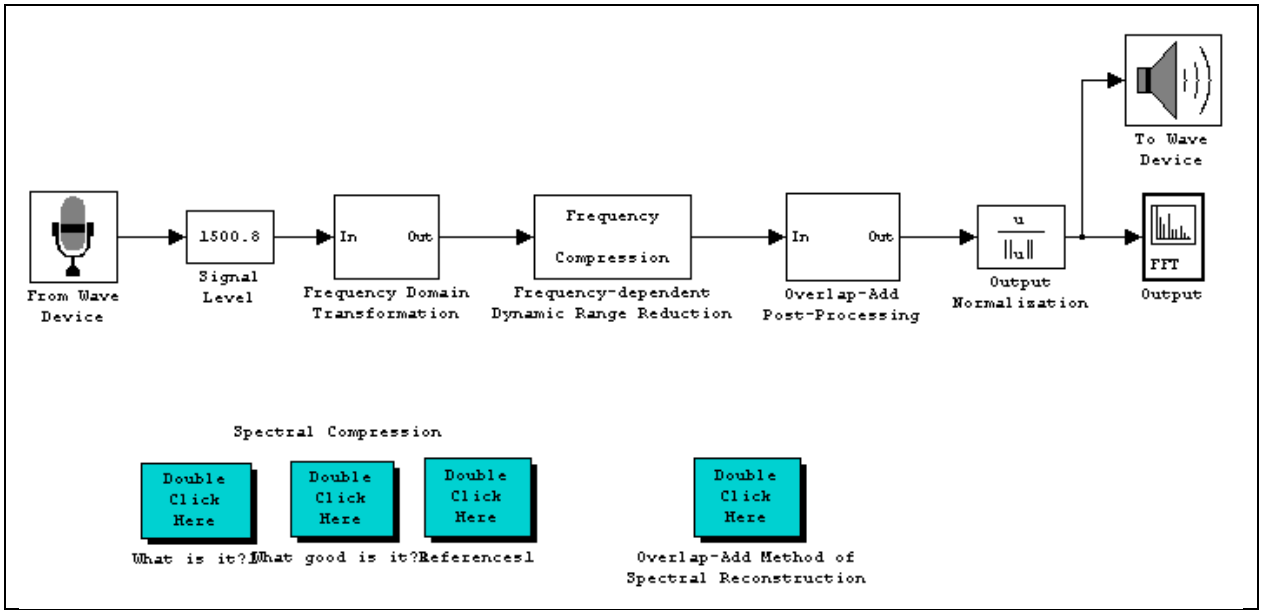


Рис. 11. Модель simulink.

3. Добавьте блоки библиотеки simulink.
4. Рассмотрите влияние новых блоков на выходной сигнал.



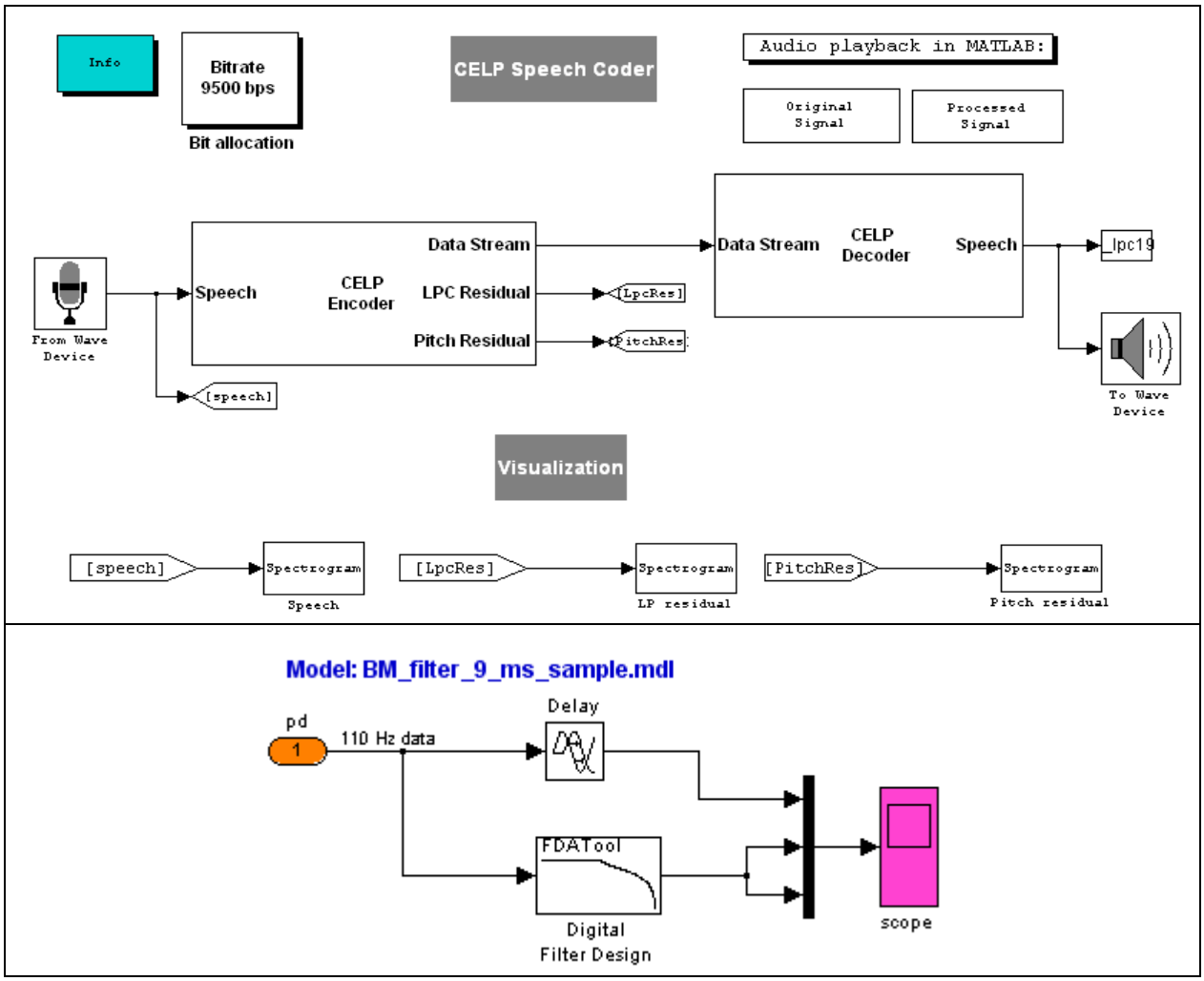


Рис. 12. Блоки моделирования канала фильтрации.

5. Постройте следующую модель.
6. Рассмотрите эффект реверберации.

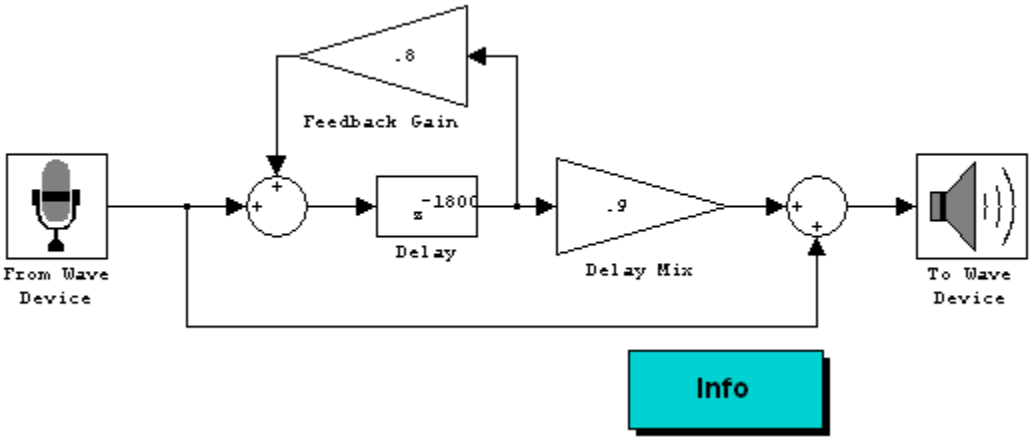


Рис. 13. Пример построения эффекта реверберации.

7. Включите в модель блок Time-Domain Filter.
8. Определите изменения звукового сигнала пропуская входной сигнал через низкочастотный и высокочастотный фильтры.

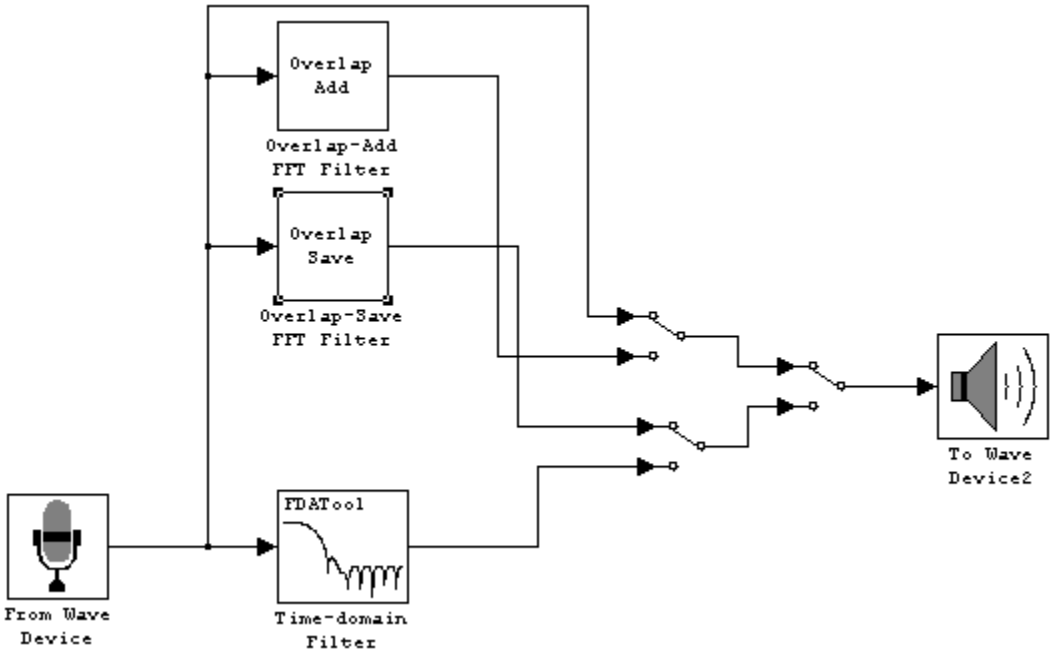


Рис. 14. Модель с блоком Time-domain Filter.

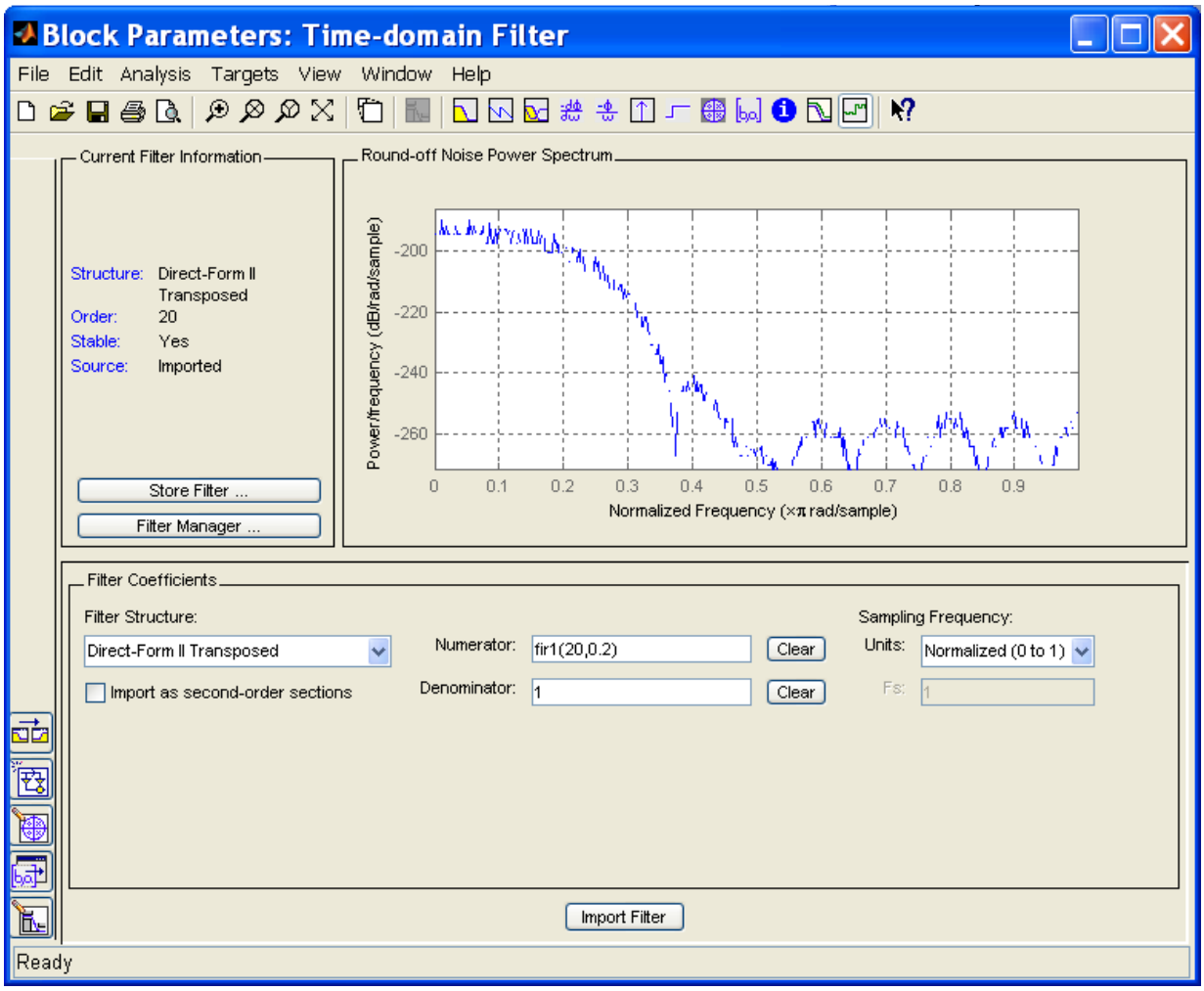
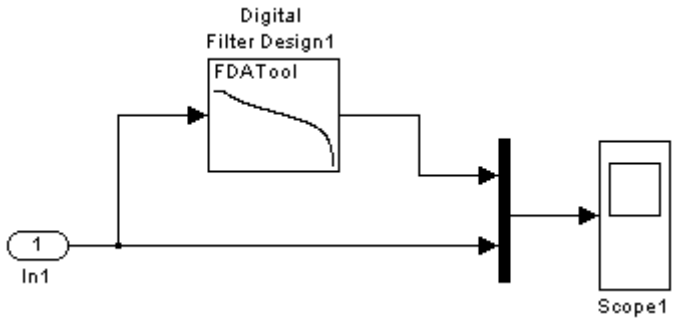


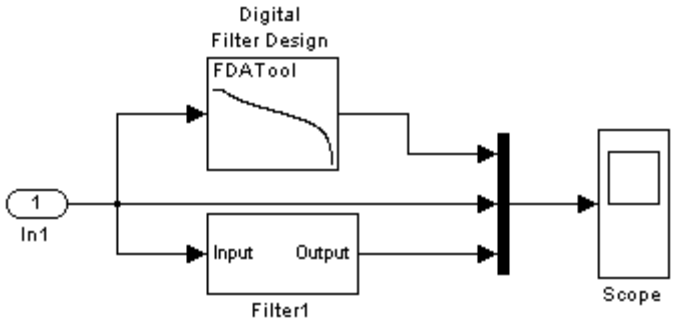
Рис. 15. Интерфейс библиотечной программы построения фильтров во временной области “Time Domain Filter

Задание 6. Построение программной реализации дискретного фильтра

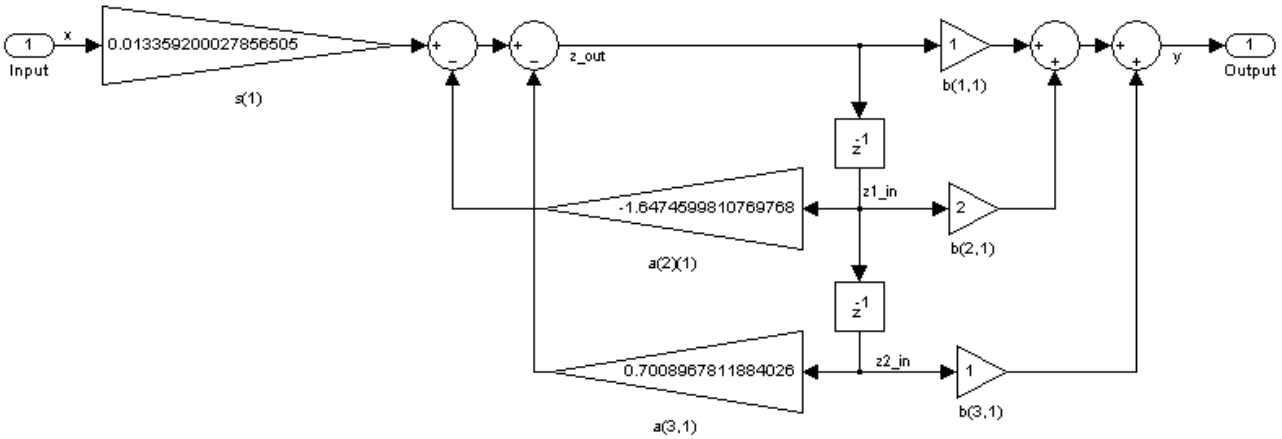
1. Постройте и введите зашумленный сигнал в модель Simulunk. Проверьте отображение сигнала при помощи блока Scope.
2. Постройте модель, показанную на рисунке.



3. Раскройте блок и постройте требуемый фильтр.
4. Постройте блок фильтра с детальной структурной схемой: Меню > File > Export to Simulink model. Убедитесь, что сигналы обоих блоков совпадают.



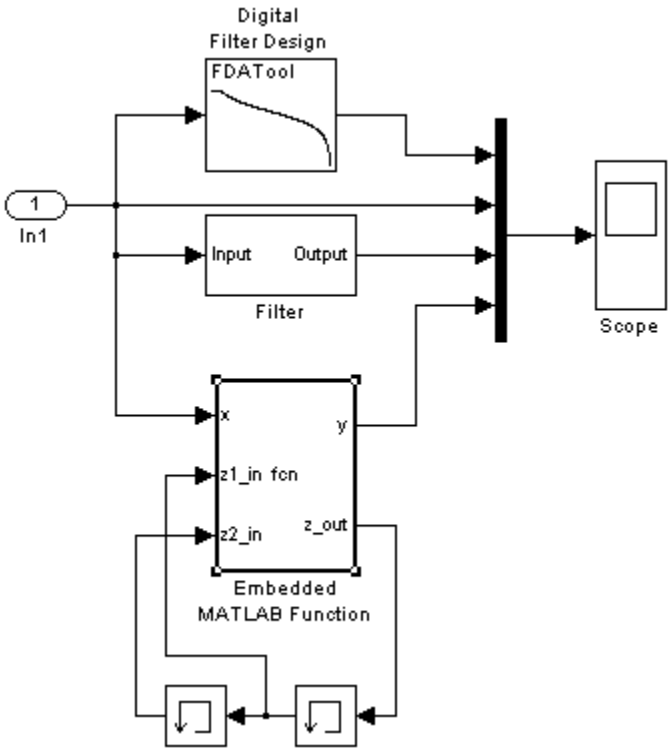
5. Раскройте появившийся новый блок в окне модели. Он содержит схему фильтра с информацией достаточной для программирования фильтра, например,



6. Постройте Embedded MatLAB функцию фильтра, например:

```
function [y, z_out] = fcn(x, z1_in, z2_in)
z_out = x * 0.013359200027856505 + 1.6474599810769768 …
      * z1_in - 0.7008967811884026 * z2_in;
y = z_out + 2 * z1_in + z2_in;
```

7. Убедитесь, что выходы всех трех реализаций фильтра Simulink совпадают.



8. Постройте программный код фильтра на любом языке программирования, например:

```
% BATTERWORT discrete-time filter

% Specify order: 2;     Fs: 10;      Fc:    0.4;

for i = 1:length(x)

    z_out = x(i) * 0.013359200027856505 + 1.6474599810769768 ...

        * z1_in - 0.7008967811884026 * z2_in;

    y (i) = z_out + 2 * z1_in + z2_in;

    z2_in = z1_in;

    z1_in = z_out;

end
```

9. Постройте выходной сигнал фильтра и сравните его с откликами фильтров модели Simulink.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какими средствами фильтрации сигналов располагает MATLAB?
2. Что такое частота среза фильтра?
3. Какими свойствами обладает режекторный фильтр?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. MatLAB Help.
2. Dr. Bob Davidov. Компьютерные технологии управления в технических системах http://portalnp.ru/author/bobdavidov