Design Example*

Ivan Selesnick

Polytechnic University
Brooklyn, NY 11201, USA
selesi@poly.edu
718 260-3416

September 2, 2004

1 Introduction

The MATLAB programs listed here reproduce Example 1 in the paper

I. W. Selesnick and A. Farras Abdelnour. Symmetric wavelet tight frames with
two generators. *Applied and Computational Harmonic Analysis*, 17(2):211-225,
September 2004. (Special Issue: Frames in Harmonic Analysis, Part II.)

The next pages are obtained by running the program *DesignExample.m*. This uses subpro-
grams listed after.

*Research supported by ONR grant N000140310217*
The scaling function and wavelets

\[ \phi(t) \]

\[ \psi_1(t) \]

\[ \psi_2(t) \]
Filter Design for Symmetric Wavelet Tight Frames with Two Generators

This program reproduces the design in Example 1 of the paper: I. W. Selesnick and A. Farras Abdelnour, Symmetric wavelet tight frames with two generators, Applied and Computational Harmonic Analyanalysis, 17(2), 2004.

Ivan Selesnick, selesi@poly.edu, Polytechnic University, Brooklyn, NY

Contents

- Find alpha
- Find the scaling filter h0
- Verify that h0 satisfies Petukhov's condition
- Find the polynomial U(z)
- Find the polynomials A(z) and B(z)
- Verify that A(z) A(1/z) + B(z) B(1/z) = 1
- Find the wavelet filters h1 and h2
- Verify the perfect reconstruction conditions
- Find (anti-) symmetric wavelet filters h1, h2
- Plot the scaling function and wavelets

Find alpha

Find the scaling filter h0 as a linear combination of two symmetric maximally-flat filters.

```matlab
format
K0 = 5;    % K0: H0(z) will have (1+z)^K0 as a factor
VM = 2;    % VM: Number of vanishing moments

% "Symmetric Maximally-Flat" lowpass filters in Equation (46)
f0 = conv([-7 22 -7],binom(7,0:7))/2^10;
f1 = conv([0 -5 18 -5 0],binom(5,0:5))/2^8;

% Compute G0(z) and G1(z) in Equations (48) and (49)
g0 = sqrt(2)*f0(1:2:end);
g1 = sqrt(2)*f1(1:2:end);

% Compute the terms in Equation (50)
r0 = conv(g0,flip(g0));   % G0(z)G0(1/z)
r1 = conv(g1,flip(g1));   % G1(z)G1(1/z)
r01 = conv(g0,flip(g1))+conv(flip(g0),g1);   % G0(z)G1(1/z) + G0(1/z)G1(z)

% Compute the coefficients of alpha in Equation (50)
a = (-4:4==0) - 2*r0;
b = 4*r0 - 2*r01;
c = 2*r01 - 2*r0 - 2*r1;

% Compute the common factor, (-1/z + 2 + z)^2
s = [-1 2 -1]/4;
s = conv(s,s);

% For numerical accuracy, remove the common factor from each term
A = extractf(a,s);
B = extractf(b,s);
C = extractf(c,s);
```
% Perform the change of variables x = (-z + 2 - 1/z)/4
% to get the polynomials in x in the equation immediately
% before Equation (51)
q0 = z2x(A);
q1 = z2x(B);
q2 = z2x(C);

% It can be verified (in Maple, for example) that q0, q1, q2
% are exactly the following:
q0 = [-189  84  1008]/2^10;
q1 = [ 238  56  -448]/2^10;
q2 = [ -49   0     0]/2^10;

% Rearrange the coefficients to get P0(alpha), P1(alpha),
% P2(alpha) in Equation (51)
P = [q2; q1; q0];
p2 = P(:,1)';
p1 = P(:,2)';
p0 = P(:,3)';

% Compute the discriminant D(alpha)
discrim = conv(p1,p1) - 4*conv(p0,p2);

% The leading coefficient is 0, so let us remove it
discrim = discrim(2:end);

% It can be verified (in Maple, for example) that the
% discriminant is exactly
% discrim = [-112 800 -1644 981]*7^2/2^16
% so for numerical accuracy let us set
discrim = [-112 800 -1644 981];

% Compute the roots of the discriminant
rts = roots(discrim);

% The smallest of the roots gives the smoothest scaling function
alpha = min(rts)

alpha =

1.0720

Find the scaling filter h0

h0 = sqrt(2)*(alpha*f1 + (1-alpha)*f0)

% Compute smoothness coefficients (needs programs by Ojanen, for example)
% M = 2;
% K0 = 5;
% disp('SMOOTHNESS: ')
% sobolev(h0,K0,M)
% sobexp(h0,K0)
% holder(h0,K0,M)

h0 =

 0.0007  -0.0269  -0.0415   0.1906   0.5842   0.5842   0.1906  -0.0415  -0.0269
  0.0007
Verify that \( h_0 \) satisfies Petukhov’s condition

The scaling filter \( H_0(z) \) must satisfy Petukhov’s condition: that the roots of \( 2 - H_0(z) H_0(1/z) - H_0(-z) H_0(-1/z) \) are of even degree, or equivalently, that the roots of \( 1 - 2 H_0(z) H_0(1/z) \) are of even degree.

\[
\begin{align*}
rr &= \text{conv}(h0, \text{flip}(h0)); \\
rr2 &= rr; \quad rr(1:2:end) = -rr(1:2:end); \\
M &= \left(\text{length}(rr) - 1\right)/2;
\end{align*}
\]

\[
\begin{align*}
% \text{Find } 2 - H_0(z) H_0(1/z) - H(-z) H(-1/z) \\
\text{Chk} &= 2*\left((-M:M) == 0\right) - (rr + rr2); \\
% \text{Verify that all its roots are of even degree} \\
\text{ChkDbleRoots} &= \text{roots}(	ext{Chk})
\end{align*}
\]

\[
\begin{align*}
% \text{Find } 1 - 2 H_0(z) H_0(1/z) \\
h00 &= h0(1:2:end); \\
\text{Chk} &= \left(\{-4:4\} == 0\right) - 2*\text{conv}(h00, \text{flip}(h00)); \\
% \text{Verify that all its roots are of even degree} \\
\text{ChkDbleRoots} &= \text{roots}(	ext{Chk})
\end{align*}
\]

\[
\begin{align*}
\text{ChkDbleRoots} &= \\
0 &
5.5929 \\
5.5929 \\
-5.5929 \\
-5.5929 \\
1.0001 + 0.0001i \\
1.0001 - 0.0001i \\
0.9999 + 0.0001i \\
0.9999 - 0.0001i \\
-1.0001 + 0.0001i \\
-1.0001 - 0.0001i \\
-0.9999 + 0.0001i \\
-0.9999 - 0.0001i \\
0.1788 \\
0.1788 \\
-0.1788 \\
-0.1788 \\
\end{align*}
\]

\[
\begin{align*}
\text{ChkDbleRoots} &= \\
31.2802 \\
31.2802 \\
1.0002 + 0.0002i \\
1.0002 - 0.0002i \\
0.9998 + 0.0002i \\
0.9998 - 0.0002i \\
0.0320 \\
0.0320 \\
\end{align*}
\]

Find the polynomial \( U(z) \)

\[
\begin{align*}
% \text{Find the polyphase component } H00(z) \\
h00 &= h0(1:2:end); \\
N &= \text{length}(h0); \\
n &= 1-N/2:N/2-1;
\end{align*}
\]
% Find roots of $H_00(z)$
% \texttt{rts} = \texttt{roots}(h00); \quad \% \text{Values (52) in paper}

% Find $U(z)$ via spectral factorization of $1 - 2 H_00(z) H_00(1/z)$
% Note: $u$ should be a symmetric sequence.

% Find $U(z)^2$ from using Equation (20)
% \texttt{u2} = (n==0) - 2*\texttt{conv}(h00,\texttt{flip}(h00));

% For numerical accuracy, factor $(-1/z + 2 + z)^2$ out of $U(z)^2$
% \texttt{ff} = \texttt{extractf}(\texttt{u2},[1 -4 6 -4 1]);

% Find the roots (use 'dbleroots' function to improve numerical accuracy)
% \texttt{rts} = \texttt{dbleroots}(\texttt{ff});

% Form polynomial from the roots
% \texttt{u} = \texttt{poly}(\texttt{rts});

% Multiply with $(1/z - 2 + z)$
% \texttt{u} = \texttt{conv}(\texttt{u}, [1 -2 1]);

% Correctly normalize $U(z)$
% \texttt{u} = \texttt{u} \times \texttt{sqrt}(\texttt{u2}(1));

% Check that $U(z) U(1/z) = 1 - 2 H_00(z) H_00(1/z)$
% \texttt{ChkZeros} = \texttt{conv}(\texttt{u},\texttt{flip}(\texttt{u})) - \texttt{u2} \quad \% \text{this should be zero}

\texttt{ChkZeros} =
\begin{array}{cccccccc}
1.0e-12 & \times & \\
0.0000 & -0.0036 & 0.0847 & -0.3072 & 0.4523 & -0.3072 & 0.0847 & -0.0036 & 0.0000 \\
\end{array}

\textbf{Find the polynomials $A(z)$ and $B(z)$}

% Find $0.5 + 0.5 U(z)$ and $0.5 - 0.5 U(z)$
% \texttt{n} = (1-N/2)/2:(N/2-1)/2;
% \texttt{ra} = 0.5*(n==0) + 0.5*\texttt{u}; \quad \% \text{0.5 + 0.5 U(z)}
% \texttt{rb} = 0.5*(n==0) - 0.5*\texttt{u}; \quad \% \text{0.5 - 0.5 U(z)}

% Find roots of $0.5 + 0.5 U(z)$ and $0.5 - 0.5 U(z)$
% \texttt{rts} = \texttt{roots(\texttt{ra});}
% \% \text{Values (53) in paper}
% \texttt{rts} = \texttt{roots(\texttt{rb});}
% \% \text{Values (54) in paper}

% Determine the roots of $A(z)$ and $B(z)$ according to paper
% \texttt{rts} = [];
% \texttt{rts} = [];
% \texttt{k} = 1:4
% \texttt{[tmp1,k1] = min(abs(\texttt{rts} \times \texttt{h00}(k) - \texttt{rts});
% \texttt{[tmp2,k2] = min(abs(\texttt{rts} \times \texttt{h00}(k) - \texttt{rts});
% \texttt{if} \; \texttt{tmp1 < tmp2}
% \quad \texttt{rts} = \texttt{rts} \times \texttt{h00}(k);
% \texttt{else}
% \quad \texttt{rts} = \texttt{rts} \times \texttt{h00}(k);
% \texttt{end}
% \texttt{end}
% \texttt{a} = \texttt{poly(\texttt{rts});}
\[ b = \text{poly(rts}_b); \]
\[ a = a/\text{sum}(a)/\sqrt{2} \quad % \text{Normalize } A(z) \text{ so that } A(1) = 1/\sqrt{2} \]
\[ b = b/\text{sum}(b)/\sqrt{2} \quad % \text{Normalize } B(z) \text{ so that } B(1) = 1/\sqrt{2} \]

\[
a =
\begin{bmatrix}
  -0.0347 & 0.8300 & -0.0881 \\
\end{bmatrix}
\]

\[
b =
\begin{bmatrix}
  0.2160 & 0.5053 & -0.0142 \\
\end{bmatrix}
\]

Verify that \( A(z) A(1/z) + B(z) B(1/z) = 1 \)

\[
\text{ChkDelta} = \text{conv}(a,\text{flip}(a)) + \text{conv}(b,\text{flip}(b)) \quad % \text{should be delta}(n)
\]

\[
\text{ChkDelta} =
\begin{bmatrix}
  0.0000 & -0.0000 & 1.0000 & -0.0000 & 0.0000 \\
\end{bmatrix}
\]

Find the wavelet filters \( h1 \) and \( h2 \)

The filter \( h2 \) will be the time-reversed version of \( h1 \)

\[
\text{h1} = \text{flip(h1)};
\]

\[
\text{Table1} = [h0' \ h1' \ h2'] \quad % \text{Table 1 in paper}
\]

\[
\text{Table1} =
\begin{bmatrix}
  0.00069616789827 & 0.00120643067872 & -0.00020086099895 \\
  -0.02692519074183 & -0.04666026144290 & 0.00776855801988 \\
  -0.04145457368921 & -0.05765656504458 & 0.01432190717031 \\
  0.19056483888762 & 0.21828637525088 & -0.14630790303599 \\
  0.58422553883170 & 0.69498947938197 & -0.24917440947758 \\
  0.58422553883170 & 0.24917440947758 & -0.69498947938197 \\
  0.19056483888762 & -0.01432190717031 & -0.05765656504458 \\
  0.0069616789827 & -0.00020086099895 & 0.00120643067872 \\
\end{bmatrix}
\]

Verify the perfect reconstruction conditions

\[
\text{g0} = \text{flip(h0)};
\]
\[
\text{g1} = \text{flip(h1)};
\]
Find (anti-) symmetric wavelet filters $h_1, h_2$

The filters $h_1$ and $h_2$ are flips of one another, and neither are symmetric. Let us convert them to a symmetric and an anti-symmetric pair.

% Shift $h_1$ by 2 samples
h0 = [h0 0 0];
h1 = [0 0 h1];
h2 = [h2 0 0];

% Replace $h_1$ and $h_2$ by their sum and difference
tmp1 = (h1+h2)/sqrt(2);
tmp2 = (h1-h2)/sqrt(2);
h1 = tmp1;
h2 = tmp2;

% Display filter coefficients
Table2 = [h0' h1' h2']          % Table 2 in paper

Table2 =

0.00069616789827 -0.00014203017443 0.00014203017443
-0.02692519074183 0.00549320005590 -0.00549320005590
-0.04145457368921 0.01098019299360 -0.00927404236569
0.19056483888762 -0.13644909765614 0.07046152309972
0.58422553883170 0.33707999754377 -0.64578354990483
0.19056483888762 0.33707999754377 0.64578354990483
-0.04145457368921 -0.21696226276270 0.13542356651680
-0.02692519074183 -0.13644909765614 -0.07046152309972
Plot the scaling function and wavelets

```matlab
h0 = h0(1:10);
[s0,t0] = scalfn(h0); % Compute the scaling function
[w1,s,t] = wletfn(h0,h1); % Compute the first wavelet
[w2,s,t] = wletfn(h0,h2); % Compute the second wavelet

figure(1)
s1 = subplot(2,2,1);
ax1 = get(s1,'position');
s2 = subplot(2,2,2);
ax2 = get(s2,'position');
clf
s3 = subplot(2,2,1);
set(s3,'position',(ax1+ax2)/2);
plot(t0,s0);
title('\phi(t)')
axis([0 9 -0.5 1.5])

% subplot(2,2,3)
plot(t,w1);
axis([0 10 -1 1])
title('\psi_1(t)')

% subplot(2,2,4)
plot(t,w2)
axis([0 10 -1 1])
title('\psi_2(t)')
print -dpsc plots

% Make Figure 2 in paper (phase plot)
if 0
    [BA,w] = freqz(b,a);
    figure(2)
    subplot(2,2,1)
    plot(w/pi,angle(BA)/pi,w/pi,0.25*w/pi,':')
    xlabel('\omega/\pi')
    title('\angle{B(e^{j \omega})/A(e^{j \omega})}/\pi')
    axis square
    % print -dps phase
end
```
Program Listing
%% Filter Design for Symmetric Wavelet Tight Frames with Two Generators
% This program reproduces the design in Example 1 of the paper:
% I. W. Selesnick and A. Farras Abdelnour,
% Symmetric wavelet tight frames with two generators,
% Ivan Selesnick, selesi@poly.edu, Polytechnic University, Brooklyn, NY

%% Find alpha
% Find the scaling filter h0 as a linear combination of two
% symmetric maximally-flat filters.

format
K0 = 5; % K0: H0(z) will have (1+z)^K0 as a factor
VM = 2; % VM: Number of vanishing moments

"Symmetric Maximally-Flat" lowpass filters in Equation (46)
f0 = conv([-7 22 -7],binom(7,0:7))/2^10;
f1 = conv([0 -5 18 -5 0],binom(5,0:5))/2^8;

% Compute G0(z) and G1(z) in Equations (48) and (49)
g0 = sqrt(2)*f0(1:2:end);
g1 = sqrt(2)*f1(1:2:end);

% Compute the terms in Equation (50)
r0 = conv(g0,flip(g0)); % G0(z)G0(1/z)
r1 = conv(g1,flip(g1)); % G1(z)G1(1/z)
r01 = conv(g0,flip(g1))+conv(flip(g0),g1); % G0(z)G1(1/z) + G0(1/z)G1(z)

% Compute the coefficients of alpha in Equation (50)
a = (-4:4==0) - 2*r0;
b = 4*r0 - 2*r01;
c = 2*r01 - 2*r0 - 2*r1;

% Compute the common factor, (-1/z + 2 + z)^2
s = [-1 2 -1]/4;
s = conv(s,s);

% For numerical accuracy, remove the common factor from each term
A = extractf(a,s);
B = extractf(b,s);
C = extractf(c,s);

% Perform the change of variables x = (-z + 2 - 1/z)/4
% to get the polynomials in x in the equation immediately
% before Equation (51)
q0 = z2x(A);
q1 = z2x(B);
q2 = z2x(C);

% It can be verified (in Maple, for example) that q0, q1, q2
% are exactly the following:
q0 = [-189 84 1008]/2^10;
q1 = [ 238 56 -448]/2^10;
q2 = [ -49 0 0]/2^10;

% Rearrange the coefficients to get P0(alpha), P1(alpha),
% P2(alpha) in Equation (51)
P = [q2; q1; q0];
p2 = P(:,1)';
p1 = P(:,2)';
p0 = P(:,3)';

% Compute the discriminant D(alpha)
discrim = conv(p1,p1) - 4*conv(p0,p2);

% The leading coefficient is 0, so let us remove it
discrim = discrim(2:end);

% It can be verified (in Maple, for example) that the
% discriminant is exactly
% discrim = [-112 800 -1644 981]*7^2/2^16
% so for numerical accuracy let us set
discrim = [-112 800 -1644 981];

% Compute the roots of the discriminant
rts = roots(discrim);

% The smallest of the roots gives the smoothest scaling function
alpha = min(rts)

%%%% Find the scaling filter h0
h0 = sqrt(2)*(alpha*f1 + (1-alpha)*f0)

% Compute smoothness coefficients (needs programs by Ojanen, for example)
% M = 2;
% K0 = 5;
% disp('SMOOTHNESS: ')
% sobolev(h0,K0,M)
% sobexp(h0,K0)
% holder(h0,K0,M)

%%% Verify that h0 satisfies Petukhov’s condition
% The scaling filter H0(z) must satisfy Petukhov’s condition: 
% that the roots of 2 - H0(z) H0(1/z) - H0(-z) H0(-1/z) are of 
% even degree, or equivalently, that the roots of 
% 1 - 2 H00(z) H00(1/z) are of even degree.

rr = conv(h0,flip(h0));
rr2 = rr; rr(1:2:end) = -rr(1:2:end);
M = (length(rr)-1)/2;

% Find 2 - H0(z) H0(1/z) - H0(-z) H0(-1/z)
Chk = 2*((-M:M) == 0) - (rr + rr2);
% Verify that all its roots are of even degree
ChkDbleRoots = roots(Chk)

% Find 1 - 2 H00(z) H00(1/z)
hs0 = h0(1:2:end);
Chk = ((-4:4) == 0) - 2*conv(hs0,flip(hs0));
% Verify that all its roots are of even degree
ChkDbleRoots = roots(Chk)

%%% Find the polynomial U(z)

% Find the polyphase component H00(z)
hs0 = h0(1:2:end);
N = length(h0);
n = 1-N/2:N/2-1;

% Find roots of H00(z)
rts_h00 = roots(hs0); % Values (52) in paper

% Find U(z) via spectral factorization of 1 - 2 H00(z) H00(1/z)
Note: u should be a symmetric sequence.

Find U(z)^2 from using Equation (20)
\[ u_2 = (n==0) - 2*\text{conv}(h00,\text{flip}(h00)); \]

For numerical accuracy, factor \((-1/z + 2 + z)^2\) out of U(z)^2
\[ ff = \text{extractf}(u2,[1 -4 6 -4 1]); \]

Find the roots (use 'dbleroots' function to improve numerical accuracy)
\[ \text{rts}_f = \text{dbleroots}(ff); \]

Form polynomial from the roots
\[ u = \text{poly}([\text{rts}_f]); \]

Multiply with \((1/z - 2 + z)\)
\[ u = \text{conv}(u, [1 -2 1]); \]

Correctly normalize U(z)
\[ u = u*\text{sqrt}(u2(1)); \]

Check that U(z) U(1/z) = 1 - 2 H00(z) H00(1/z)
\[ \text{ChkZeros} = \text{conv}(u,\text{flip}(u)) - u2 \quad \text{this should be zero} \]

Find the polynomials A(z) and B(z)

Find 0.5 + 0.5 U(z) and 0.5 - 0.5 U(z)
\[ n = (1-N/2)/2:(N/2-1)/2; \]
\[ ra = 0.5*(n==0) + 0.5*u; \]
\[ rb = 0.5*(n==0) - 0.5*u; \]

Find roots of 0.5 + 0.5 U(z) and 0.5 - 0.5 U(z)
\[ \text{rts}_ra = \text{roots}(ra); \quad \text{Values (53) in paper} \]
\[ \text{rts}_rb = \text{roots}(rb); \quad \text{Values (54) in paper} \]

Determine the roots of A(z) and B(z) according to paper
\[ \text{rts}_a = []; \]
\[ \text{rts}_b = []; \]
\[ \text{for} \ k = 1:4 \]
\[ \quad [\text{tmp1,k1}] = \text{min}(\text{abs}(\text{rts}_h00(k)-\text{rts}_ra)); \]
\[ \quad [\text{tmp2,k2}] = \text{min}(\text{abs}(\text{rts}_h00(k)-\text{rts}_rb)); \]
\[ \quad \text{if} \ \text{tmp1} < \text{tmp2} \]
\[ \quad \quad \text{rts}_a = [\text{rts}_a \ \text{rts}_h00(k)]; \]
else
  rts_b = [rts_b 1/rts_h00(k)];
end

% Find A(z) and B(z)
a = poly(rts_a);
b = poly(rts_b);
a = a/sum(a)/sqrt(2) % Normalize A(z) so that A(1) = 1/sqrt(2)
b = b/sum(b)/sqrt(2) % Normalize B(z) so that B(1) = 1/sqrt(2)

%% Verify that A(z) A(1/z) + B(z) B(1/z) = 1
ChkDelta = conv(a,flip(a)) + conv(b,flip(b)) % should be delta(n)

%% Find the wavelet filters h1 and h2
% The filter h2 will be the time-reversed version of h1

% Determine H10(z) and H11(z)
h10 = conv(a,a); % H10(z) = A^2(z)
h11 = -conv(b,b); % H11(z) = -B^2(z)

% Determine H1(z) and H2(z)
h1 = [h10; h11]; h1 = h1(:)';
h2 = flip(h1);

% Display filter coefficients
format long
Table1 = [h0' h1' h2'] % Table 1 in paper

%% Verify the perfect reconstruction conditions

g0 = flip(h0);
g1 = flip(h1);
g2 = flip(h2);
pr1 = conv(h0,g0) + conv(h1,g1) + conv(h2,g2);
N = length(h0);
s = (-1).^(0:N-1);
pr2 = conv(h0.*s,g0) + conv(h1.*s,g1) + conv(h2.*s,g2);
CheckPR = [pr1' pr2']

%% Find (anti-) symmetric wavelet filters h1, h2
% The filters h1 and h2 are flips of one another, and neither are symmetric.
% Let us convert them to a symmetric and an anti-symmetric pair.

% Shift h1 by 2 samples
h0 = [h0 0 0];
h1 = [0 0 h1];
h2 = [h2 0 0];

% Replace h1 and h2 by their sum and difference
tmp1 = (h1+h2)/sqrt(2);
tmp2 = (h1-h2)/sqrt(2);
h1 = tmp1;
h2 = tmp2;

% Display filter coefficients
Table2 = [h0' h1' h2']
% Table 2 in paper

%% Plot the scaling function and wavelets

h0 = h0(1:10);
[s0,t0] = scalfn(h0); % Compute the scaling function
[w1,s,t] = wletfn(h0,h1); % Compute the first wavelet
[w2,s,t] = wletfn(h0,h2); % Compute the second wavelet

figure(1)
s1 = subplot(2,2,1);
a1 = get(s1,'position');
s2 = subplot(2,2,2);
a2 = get(s2,'position');
c1
s3 = subplot(2,2,1);
set(s3,'position',(a1+a2)/2);
plot(t0,s0);
title('\phi(t)')
axis([0 9 -0.5 1.5])

subplot(2,2,3)
plot(t,w1);
axis([0 10 -1 1])
title('\psi_1(t)')
subplot(2,2,4)
plot(t,w2)
axis([0 10 -1 1])
title('
psi_2(t)')
print -depsc plots

% Make Figure 2 in paper (phase plot)
if 0
    [BA,w] = freqz(b,a);
    figure(2)
    subplot(2,2,1)
    plot(w/pi,angle(BA)/pi,w/pi,0.25*w/pi,:')
    xlabel('
\omega/\pi')
    title('
\angle\{B(e^{j \omega})/A(e^{-j \omega})\}/\pi')
    axis square
    % print -depsc phase
end
function a = binom(n,k)
%
% a = binom(n,k)
% BINOMIAL COEFFICIENTS
%
% allowable inputs:
%    n : integer, k : integer
%    n : integer vector, k : integer
%    n : integer, k : integer vector
%    n : integer vector, k : integer vector (of equal dimension)
%
% Ivan Selesnick
% selesi@poly.edu
% Polytechnic University
% Brooklyn, NY, USA

nv = n;
kv = k;
if (length(nv) == 1) & (length(kv) > 1)  
    nv = nv * ones(size(kv));
elseif (length(nv) > 1) & (length(kv) == 1)  
    kv = kv * ones(size(nv));
end
a = nv;
for i = 1:length(nv)  
    n = nv(i);
    k = kv(i);
    if n >= 0
        if k >= 0
            if n >= k  
                c = prod(1:n)/(prod(1:k)*prod(1:n-k));
            else
                c = 0;
            end
        else
            c = 0;
        end
    else
        c = 0;
    end
end
\[
c = (-1)^k \frac{\text{prod}(1:k-n-1)}{\text{prod}(1:k) \cdot \text{prod}(1:-n-1)};
\]
else
  if \( n \geq k \)
    \[
c = (-1)^{n-k} \frac{\text{prod}(1:-k-1)}{\text{prod}(1:n-k) \cdot \text{prod}(1:-n-1)};
\]
  else
    \[
c = 0;
\]
  end
end
end
a(i) = c;
end
function rts = dbleroots(p)

% Find roots of a polynomial with double roots:
% P(z) = Q(z)^2.
% Find roots of Q(z).
% Proceed by taking derivative of P(z) to improve
% the numerical accuracy of the root computation.

% Ivan Selesnick
% selesi@poly.edu
% Polytechnic University
% Brooklyn, NY, USA

N = length(p)-1;
p = p(:)';    % ensure p is a row vector
pdiff = p(1:N).* (N:-1:1);

rts_p = roots(p);
rts_pdiff = roots(pdiff);

rts = rts_p;
for k = 1:N
    [tmp, i] = min(abs(rts(k)-rts_pdiff));
    rts(k) = rts_pdiff(i);
end

trim = zeros(N/2,1);
for i = 1:N/2
    trim(i) = rts(1);
    rts(1) = [];
    [tmp,k] = min(abs(trim(i)-rts));
    rts(k) = [];
end

rts = trim;
function f = extractf(h,p);

% f = extractf(h,p)
% find f such that h = conv(f,p)
% When such an f exists, this function has better
% numerical accuracy that the "deconv" command

% Ivan Selesnick
% selesi@poly.edu
% Polytechnic University
% Brooklyn, NY, USA

p = p(:);
h = h(:);
Np = length(p);
Nh = length(h);

C = convmtx(p,Nh-Np+1);

f = C\h;

% check accuracy of result:

SN = 0.000001;   % Small Number
e = max(abs(C*f - h));
% disp(e)
if e > SN
    disp('there is a problem in extractf')
    keyboard
end

f = f';
function b = flip(a)

b = a(end:-1:1);
function [s,t] = scalfn(h,J)
% [s,t] = scalfn(h,J);
% Scaling function obtained by dyadic expansion
% input
% h : scaling filter
% output
% s : samples of the scaling function phi(t)
% for t = k/2^J, k=0,1,2,...
% % Example:
% h = [1+sqrt(3) 3+sqrt(3) 3-sqrt(3) 1-sqrt(3)]/(4*sqrt(2));
% [s,t] = scalfn(h);
% plot(t,s)

% Ivan Selesnick
% selesi@poly.edu
% Polytechnic University
% Brooklyn, NY, USA

if nargin < 2
    J = 5;
end

N = length(h);
h = h(:).'; % form a row vector

% check sum rules
n = 0:N-1;
e0 = sum(h) - sqrt(2);
e1 = sum((-1).^n.*h);
if abs(e0) > 0.0001
    disp(' need: sum(h(n)) = sqrt(2)')
    return
end
if abs(e1) > 0.0001
    disp(' need: sum((-1)^n h(n)) = 0')
    return
end

% Make convolution matrix
H = toeplitz([h zeros(1,N-1)],[h(1) zeros(1,N-1)]);
% or: H = convmtx(h(:),N);

% Make P matrix
P = sqrt(2)*H(1:2:2*N-1,:);

% Solve for vector
s = [P-eye(N); ones(1,N)] \ [zeros(N,1); 1];
L = N;        % length of phi vector

% phi at integers
s = s.';

% Loop through scales
for k = 0:J-1
    s = sqrt(2)*conv(h,s);
    L = 2*L-1;
    s = s(1:L);
    h = up(h,2);
end

% Time axis
t = (0:L-1)*(N-1)/(L-1);
function y = up(x,M)
% y = up(x,M)
% M-fold up-sampling of a 1-D signal

[r,c] = size(x);
if r > c
    y = zeros(M*r,1);
else
    y = zeros(1,M*c);
end
y(1:M:end) = x;
function [w,s,t] = wletfn(h0,h1,K);

% [w,s,t] = wletfn(h0,h1,K);
% % Computes the scaling function and wavelet
% % Ivan Selesnick
% % selesi@poly.edu
% % Polytechnic University
% % Brooklyn, NY, USA

if nargin < 3
    K = 7;
end

N0 = length(h0);
N1 = length(h1);

[s,t] = scalfn(h0,K);

L = length(s);

w = sqrt(2)*conv(up(h1,2^(K-1)),s(1:2:L));

L = (N0-1)/2 + (N1-1)/2;

t = [0:2^K*L]/2^K;

w = w(1:(2^K*L+1));
function p = z2x(h)
% p = z2x(h)
% Implements the change of variables
% x = (-z + 2 - 1/z)/4
% where h(z) is a odd-length symmetric filter

% Ivan Selesnick
% selesi@poly.edu
% Polytechnic University
% Brooklyn, NY, USA

N = length(h);
M = (N-1)/2;
p = [];
g = 1;
for k = 1:M
    g = conv(g,[-1 2 -1]/4);
end
for k = 0:M
    [q,r] = deconv(h,g);
    p(M+1-k) = q;
    h = r(2:end-1);
    g = deconv(g,[-1 2 -1]/4);
end
p = p(end:-1:1);