Deuternaopia Combined Filter Analysis

Table of Contents

This analysis is based on assuming the filter characteristics to be ideal. This is not true in the real world scenario. The graph will have infinite number of poles and infinite number of zeros to get the ideal characteristics

Clearing previous values

```
clear all
clc
```

For Expanded Transfer function

```
disp ('Expanded Transfer Functions');
 num = [1 1+i];
 den = [1 1];
 n_roots = roots(num);
 d_roots = roots(den);
```

```
Expanded Transfer Functions
```

Zeros of the transfer functions

```
n_roots = [0.9*cos(105*pi/180)+i*0.9*sin(105*pi/180)
 cos(124*pi/180)+i*sin(124*pi/180)
 0.9*cos(148.96*pi/180)+i*0.9*sin(148.96*pi/180)]
```

```
n_roots =

  -0.2329 + 0.8693i  -0.5592 + 0.8290i  -0.7711 + 0.4641i
```

Poles of the transfer functions

```
d_roots = [ 0.8*cos(55*pi/180)+i*0.8*sin(55*pi/180)
 0.9*cos(74*pi/180)+i*0.9*sin(74*pi/180)
 0.8*cos(86.9*pi/180)+i*0.8*sin(86.9*pi/180)]
```

```
d_roots =

   0.4589 + 0.6553i   0.2481 + 0.8651i   0.0433 + 0.7988i
```

Commented for complex analysis

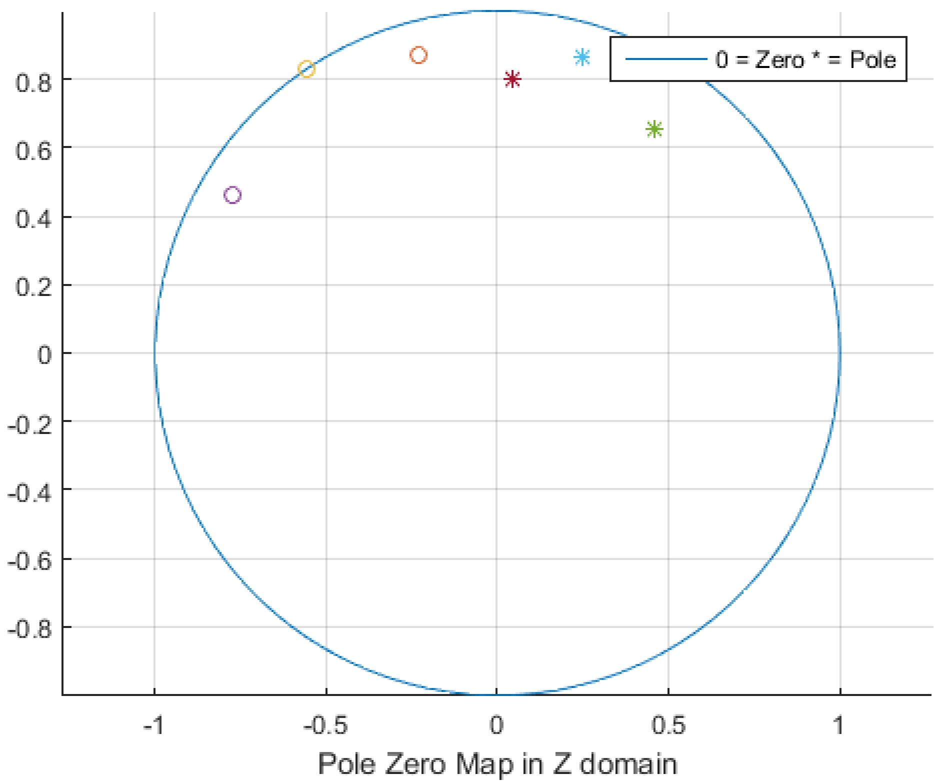
```
% [loop1, ~] = size(n_roots);
% [loop2, ~] = size(d_roots);
%
% for i = 1:loop1
%
% a = real(n_roots(i));
% b = imag(n_roots(i));
%
% r = sqrt(a^2 + b^2);
%
% theta = atan(b/a);
%
% zeros_mag(i) = r;
% zeros_angle(i) = theta;
%
% end
%
% for j = 1:loop2
%
% c = real(n_roots(j));
% d = imag(n_roots(j));
%
% r = sqrt(c^2 + d^2);
%
% theta = atan(d/c);
%
% poles_mag(j) = r;
% poles_angle(j) = theta;
% end
```

Plotting PZ map in Z domain

```
r = 1;
xc = 0;
yc = 0;
grid on
hold on
theta = linspace(0,2*pi);
xlabel('Pole Zero Map in Z domain')
x = r*cos(theta) + xc;
y = r*sin(theta) + yc;
plot(x,y)
legend('boxon')
legend('0 = Zero * = Pole');
```

```
[l1, m1] = size(n_roots);

[l2, m2] = size(d_roots);

for i=1:m1

    disp('KTB')

plot(real(n_roots(i)),imag(n_roots(i)),'o');

end



for i=1:m2

plot(real(d_roots(i)),imag(d_roots(i)),'*');

end

axis equal
```

KTB
KTB
KTB



For phase and magnitude plot

```
equity = 0;

equity_num = 0;
equity_den = 0;

%syms z;

w = 1:180;

x = cos((w*pi)/180) + i*sin((w*pi)/180);

z  = x;


for k = 1:(l1)
%disp('KTB');
equity_num = equity_num + z.^k * num(l1 - k + 1);

end

equity_num = equity_num + num(l1+1);


for m = 1:(l2)

equity_den = equity_den + z.^m * den(l2 - m + 1);

end


equity_den = equity_den + den(l2+1);


equity = equity_num / equity_den;
```

Published with MATLAB® R2015a