PART IV: Finite wordlength effects in practice

- This part can be divided into the following four topics:
- How to easily quantize the coefficients of direct-form FIR filters.
- How to easily quantize the coefficients of IIR filters implemented as a cascade of second- and first-order blocks.
- How to easily quantize the coefficients of IIR filters being implementable as a parallel connection of two allpass filters.
- Validity of the commonly used noise model for estimating the output noise due to the multiplication roundoff errors.

QUANTIZATION OF FIR FILTER COEFFICIENTS

- Consider a Type I or Type II FIR filter with transfer function of the form

$$H(z) = \sum_{n=0}^{N} h(n) z^{-n}, \quad h(N-n) = h(n).$$

- The zero-phase frequency response of this filter is given by (see the lecture notes on DIGITAL FILTERING II)

$$H(\omega) = \begin{cases} h(\frac{N}{2}) + 2\sum_{n=1}^{N/2} h(\frac{N}{2} - n)\cos n\omega, & N \text{ even} \\ 2\sum_{n=0}^{(N-1)/2} h(\frac{N-1}{2} - n)\cos((n+\frac{1}{2})\omega), & N \text{ odd.} \end{cases}$$

- After rounding the coefficients to b fractional bits, we get

$$H_b(z) = \sum_{n=0}^{N} h_b(n) z^{-n}, \quad h_b(N-n) = h_b(n)$$

and

$$H_b(\omega) = \begin{cases} h_b(\frac{N}{2}) + 2\sum_{n=1}^{N/2} h_b(\frac{N}{2} - n)\cos n\omega, & N \text{ even} \\ 2\sum_{n=0}^{(N-1)/2} h_b(\frac{N-1}{2} - n)\cos((n+\frac{1}{2})\omega), & N \text{ odd,} \end{cases}$$

where

$$h_b(n) = \text{round}(2^b h(n))/2^b, \quad n = 0, 1, \cdots, N$$

with round(x) standing for rounding x to the nearest integer.

- Alternatively, we can write

$$H_b(z) = H(z) + E_b(z)$$

$$H_b(\omega) = H(\omega) + E_b(\omega),$$

where

$$E_b(z) = \sum_{n=0}^{N} e_b(n) z^{-n}, \quad e_b(N-n) = e_b(n)$$

and

$$E_b(\omega) = \begin{cases} e_b(\frac{N}{2}) + 2\sum_{n=1}^{N/2} e_b(\frac{N}{2} - n)\cos n\omega, & N \text{ even} \\ 2\sum_{n=0}^{(N-1)/2} e_b(\frac{N-1}{2} - n)\cos((n+\frac{1}{2})\omega), & N \text{ odd} \end{cases}$$

with

$$e_b(n) = h_b(n) - h(n) \quad n = 0, 1, \cdots, N.$$

- Based on the above formulas, the filter transfer function $H_b(z)$ can be interpreted as a parallel connection of the ideal transfer function $H(z)$ with

infinite-precision coefficients and an error transfer function $E_b(z)$.

- Similarly, $H_b(\omega)$ is the sum of $H(\omega)$, the zero-phase frequency response of the ideal transfer function $H(z)$, and $E_b(\omega)$, the zero-phase frequency response of the error transfer function $E_b(z)$.
- In order for $H_b(z)$ to meet the given citeria, it is required that $H(z)$ be designed to stay well within the given limits such that $H_b(\omega) = H(\omega) + E_b(\omega)$ still meets the criteria.
- For

$$\delta_e = \max_{\omega \in [0,\ \pi]} |E_b(\omega)|,$$

the following three estimates have been developed in the literature:

$$\delta_e = (N+1)2^{-(b+1)} \qquad (A)$$

$$\delta_e = 2^{-(b+1)}[(N+1)/3]^{1/2} \qquad (B)$$

$$\delta_e = 2^{-(b+1)}[(N+1)\log_e(N+1)/3]^{1/2}. \qquad (C)$$

- Among these formulas, Eq. (C) gives generally the best estimate especially for long filters.

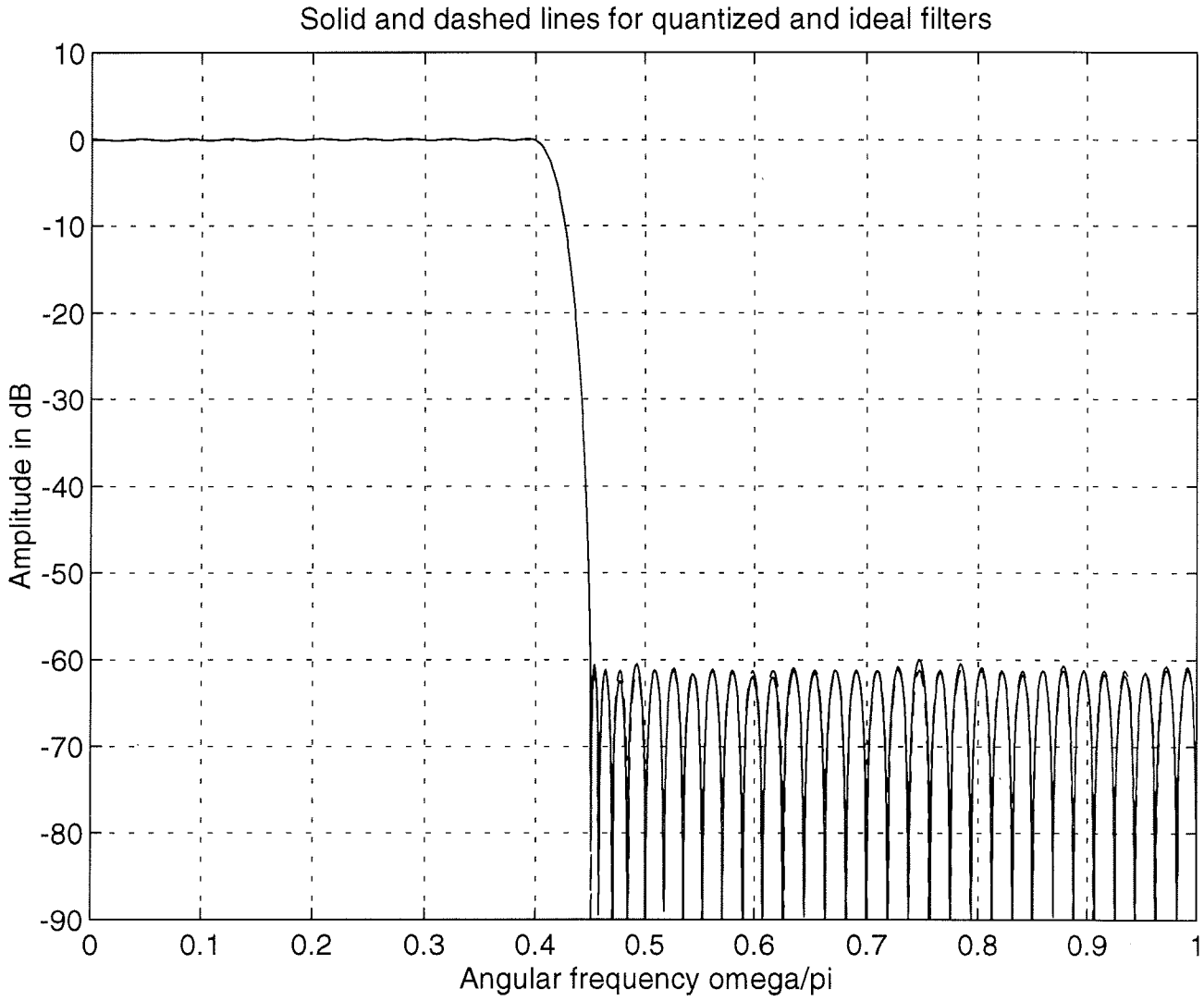
- The role of δ_e is the following. If the infinite-precision lowpass filter has been designed such that its passband and stopband ripples are δ_p and δ_s, respectively, then the corresponding ripples for the quantized filter are approximately $\delta_p + \delta_e$ and $\delta_s + \delta_e$, respectively.
- The following example illustrates the use of δ_e in designing FIR filters with quantized coefficients.
- In the end of this pile of lecture notes you can find a Matlab-file firquan.m for analysing filters with quantized coefficient values. The Matlab-file firgen.m considered in DIGITAL FILTERING II can be used for automatically determining the minimum FIR filter order to meet the given criteria.

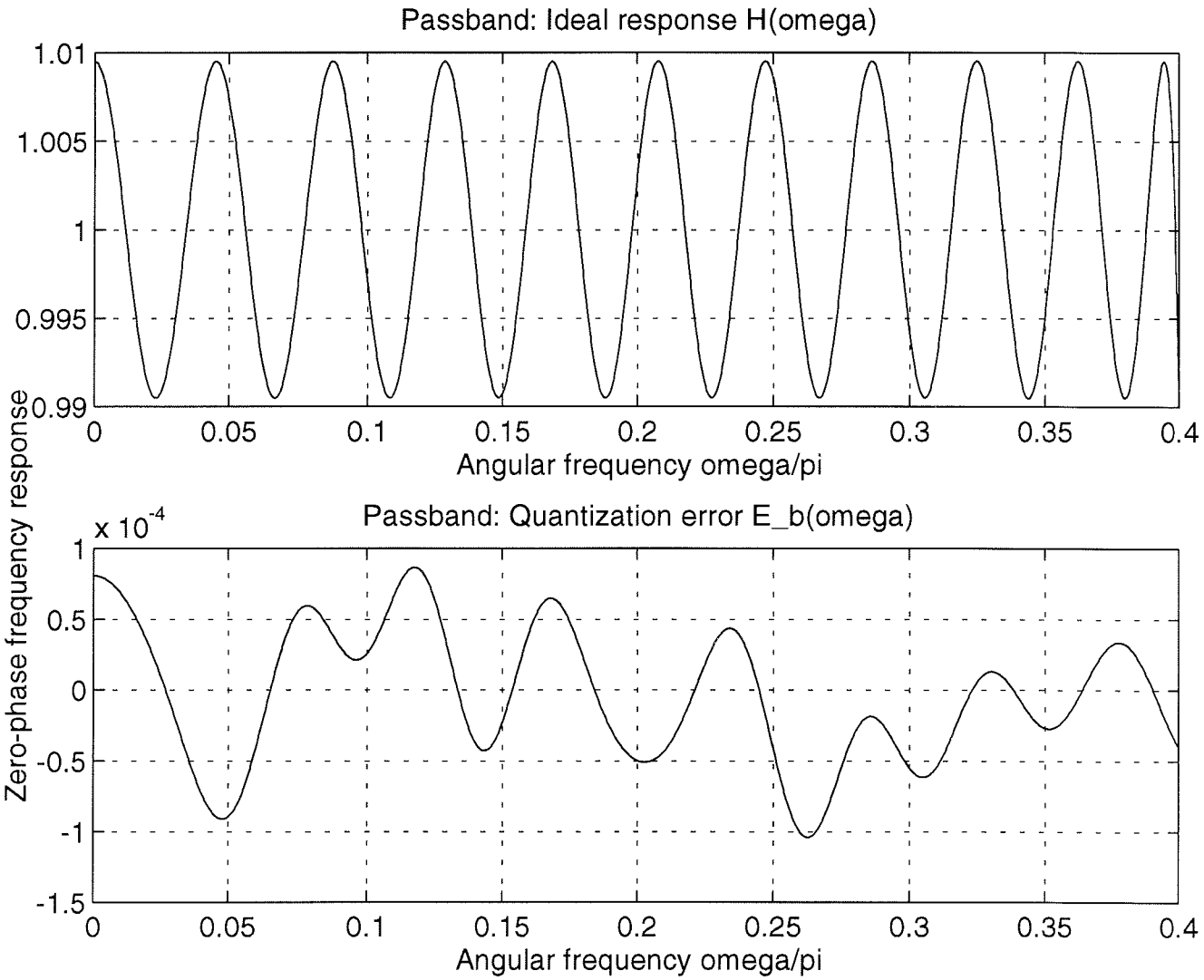
EXAMPLE

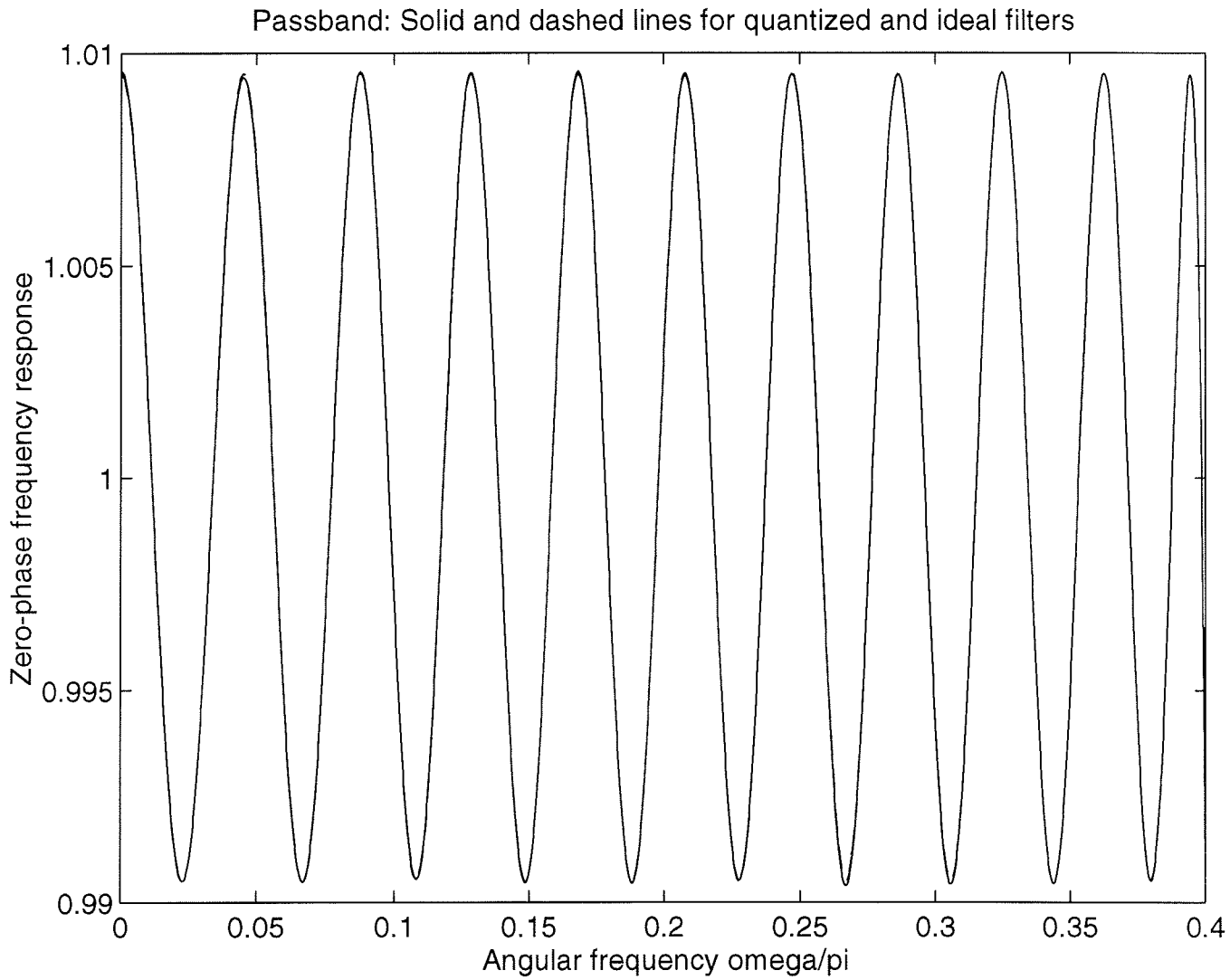
- It is desired to design a lowpass filter with edges at $\omega_p = 0.4\pi$ and $\omega_s = 0.45\pi$ and ripples of $\delta_p = 0.01$ and $\delta_s = 0.001$.
- The minimum order to meet these criteria is $N = 104$.
- For $b = 16$ fractional bits and $N = 104$, Eq. (C) gives $\delta_e = 0.0000973 \approx 0.0001$.
- Based on this, we first design a filter with ripples $\delta_p - \delta_e = 0.0099$ and $\delta_s - \delta_e = 0.0009$. The minimum order is $N = 105$.
- When the coefficients are rounded, 16 bits is in fact the minimum number of fractional bits needed for the zero-phase response to stay within the limits $1 \pm \delta_p = 1 \pm 0.01$ in the passband and within the limits $\pm\delta_s = \pm 0.001$ in the stopband.
- In the following there are five figures giving the responses $H(\omega)$, $H_b(\omega)$, and $E_b(\omega)$. It is seen that the peak absolute value of $E_b(\omega)$, occuring in the stopband region, is approximately 0.00013, which is very

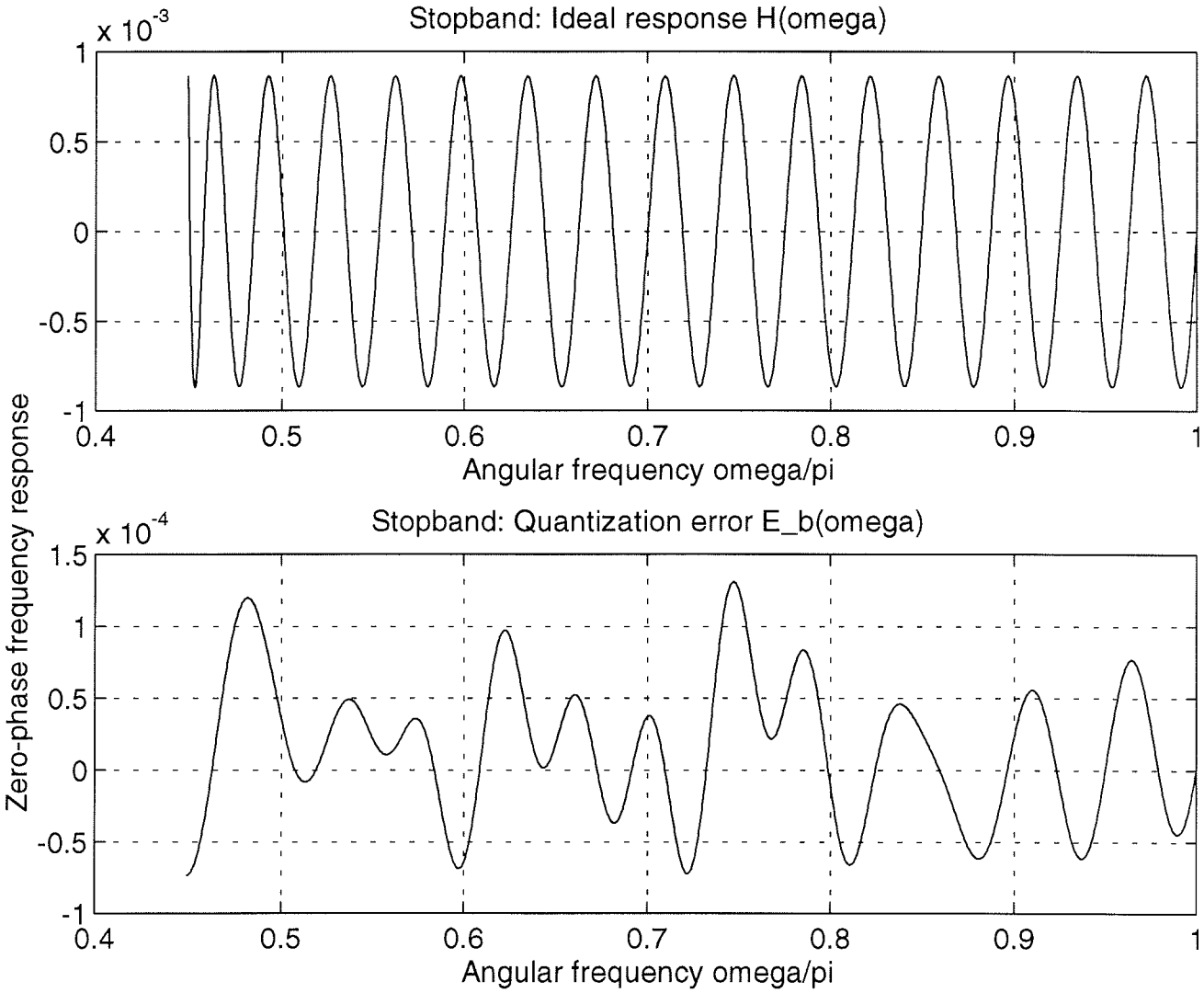
is very close to the estimated value of 0.0001.

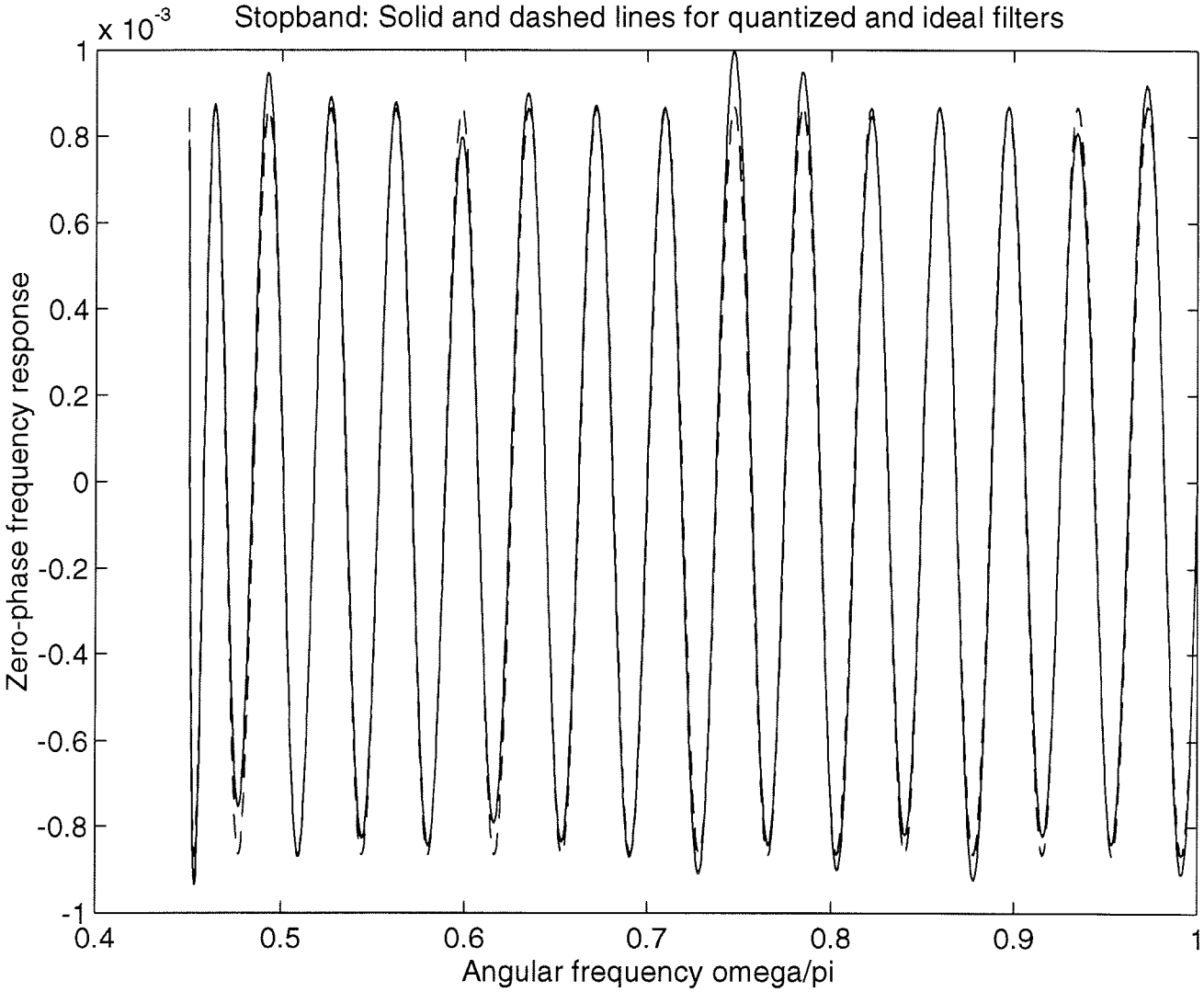
- However, the quantized filter still meets the given criteria, If not, then the filter order should be increased by one or two.


Solid and dashed lines for quantized and ideal filters
Amplitude in dB
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Passband: Ideal response H(omega)
1.01
1.005
1
0.995
0.99
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi
Zero-phase frequency response
Passband: Quantization error E_b(omega)
x 10^-4
1
0.5
0
-0.5
-1
-1.5
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi

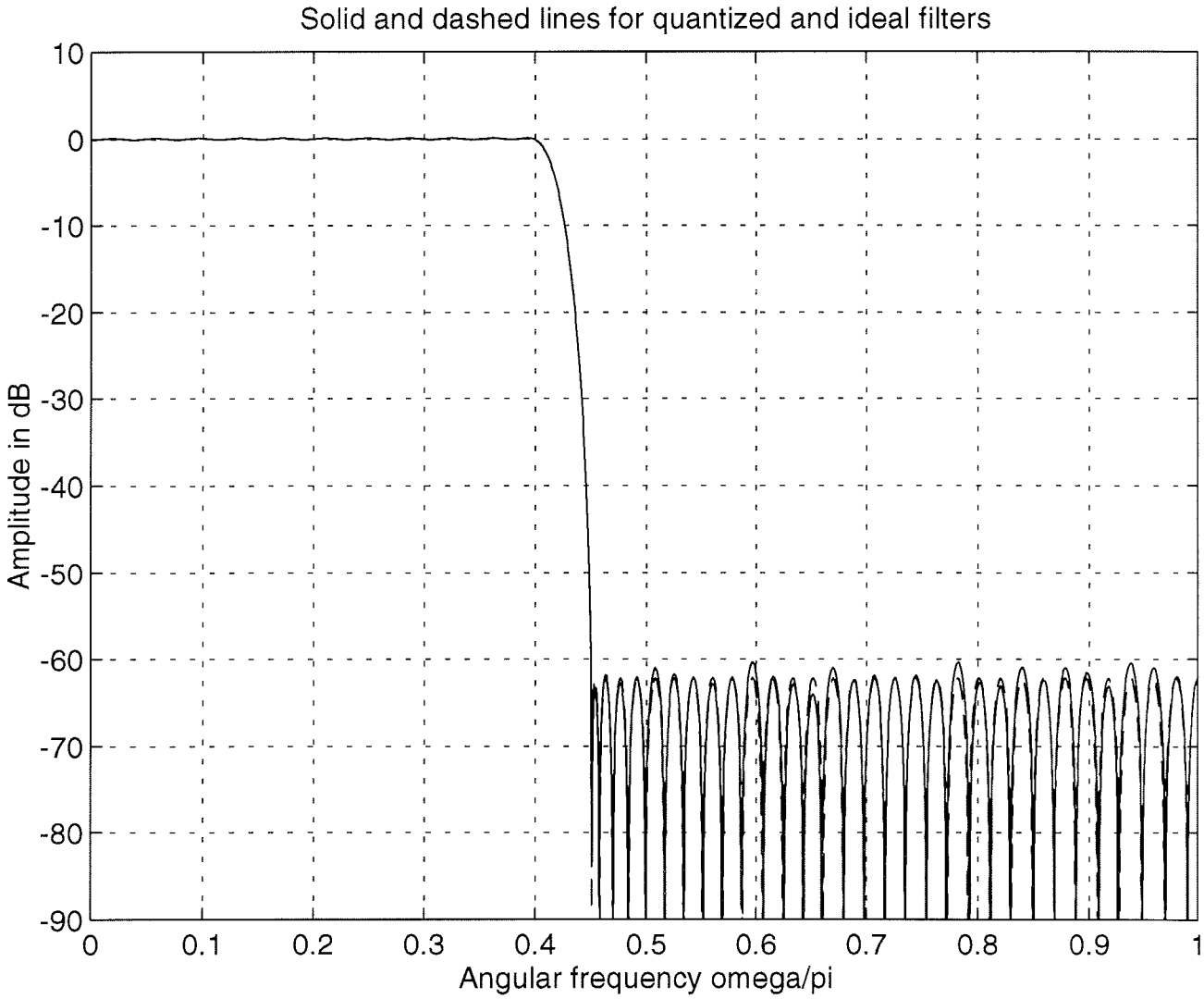

Passband: Solid and dashed lines for quantized and ideal filters
1.01
1.005
1
0.995
0.99
Zero-phase frequency response
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi

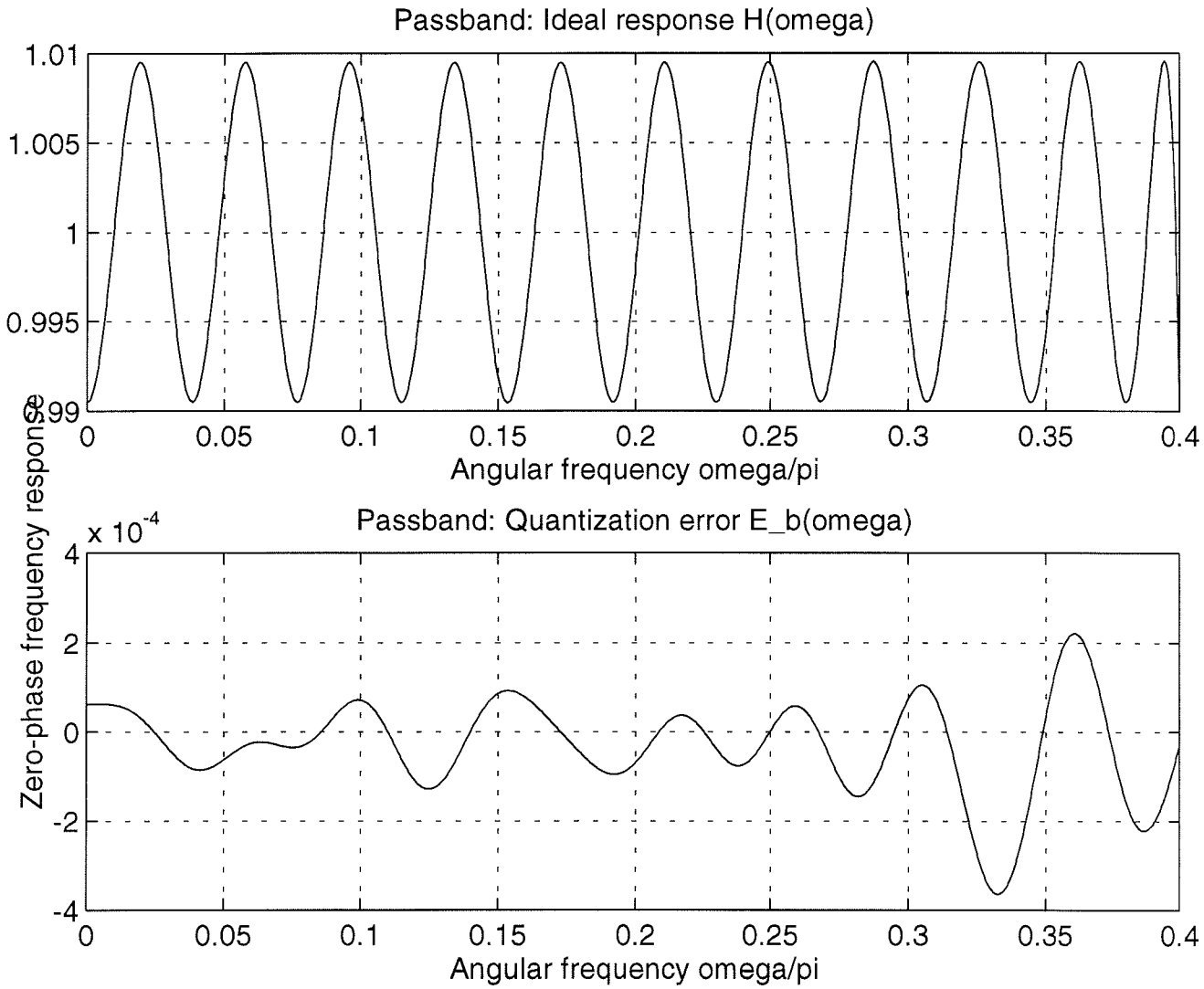

Stopband: Ideal response H(omega)
x 10^-3
1
0.5
0
-0.5
-1
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi
Stopband: Quantization error E_b(omega)
x 10^-4
1.5
1
0.5
0
-0.5
-1
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi
Zero-phase frequency response

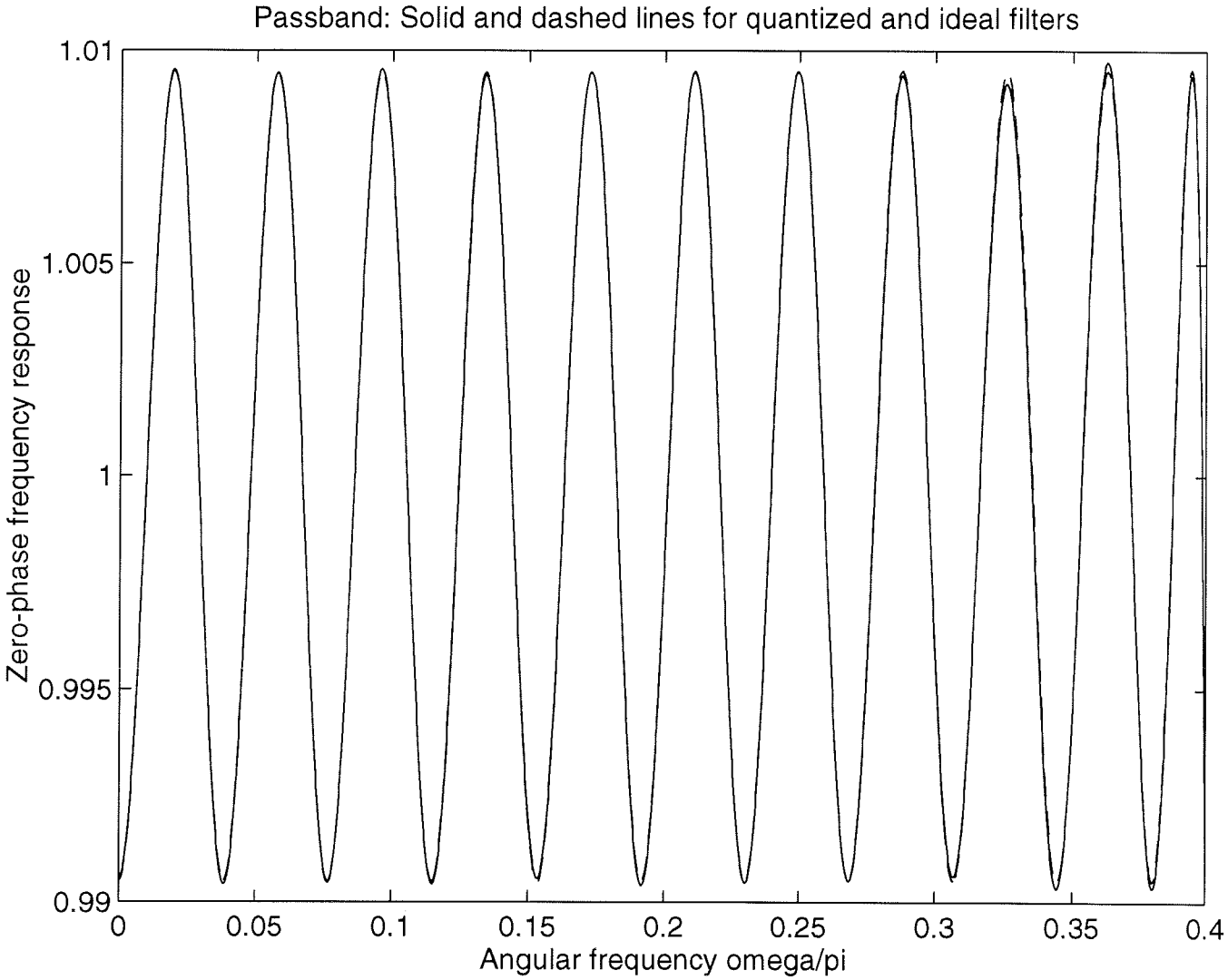

Stopband: Solid and dashed lines for quantized and ideal filters
x 10^-3
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Zero-phase frequency response
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

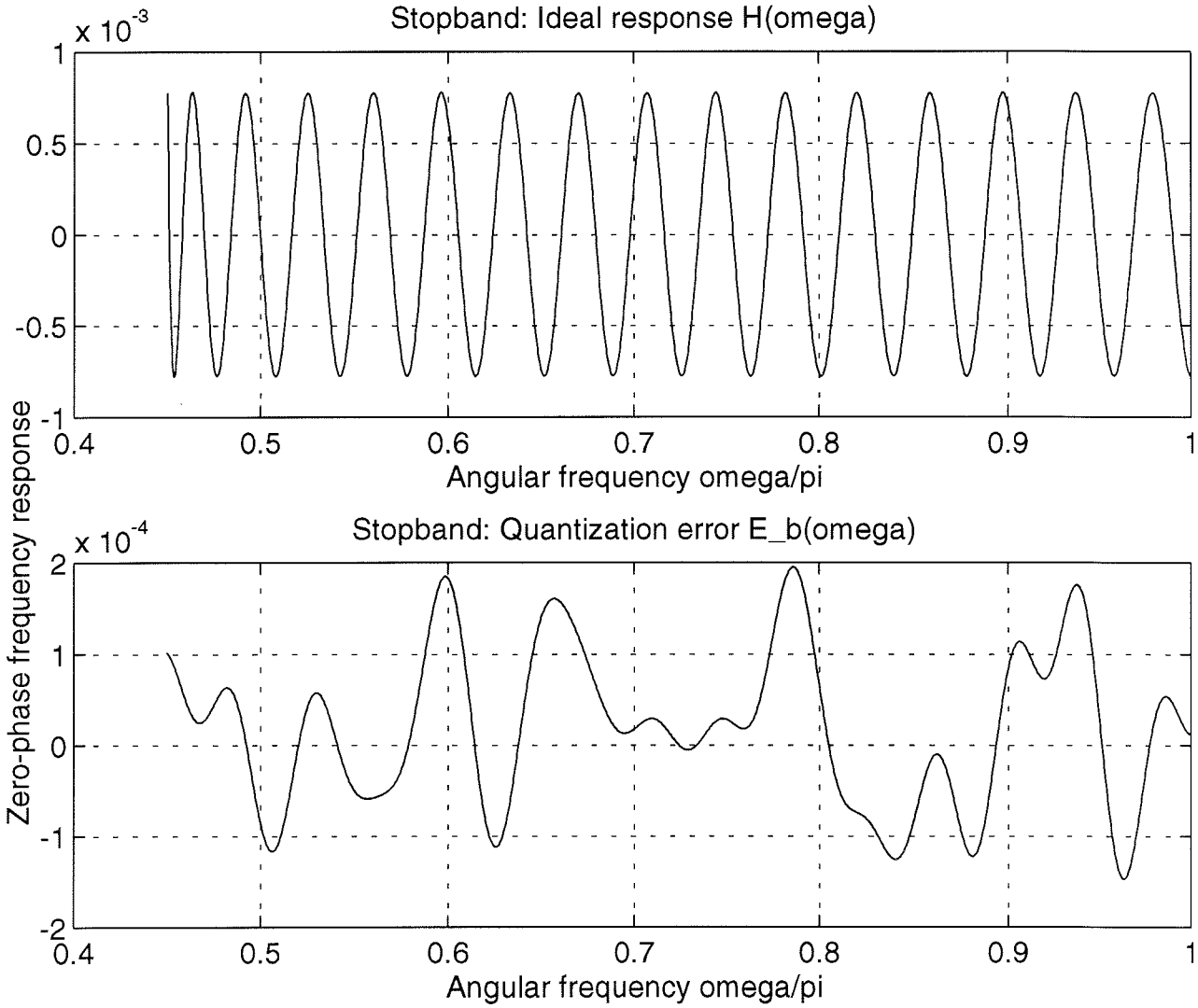
b =15 fractional bits

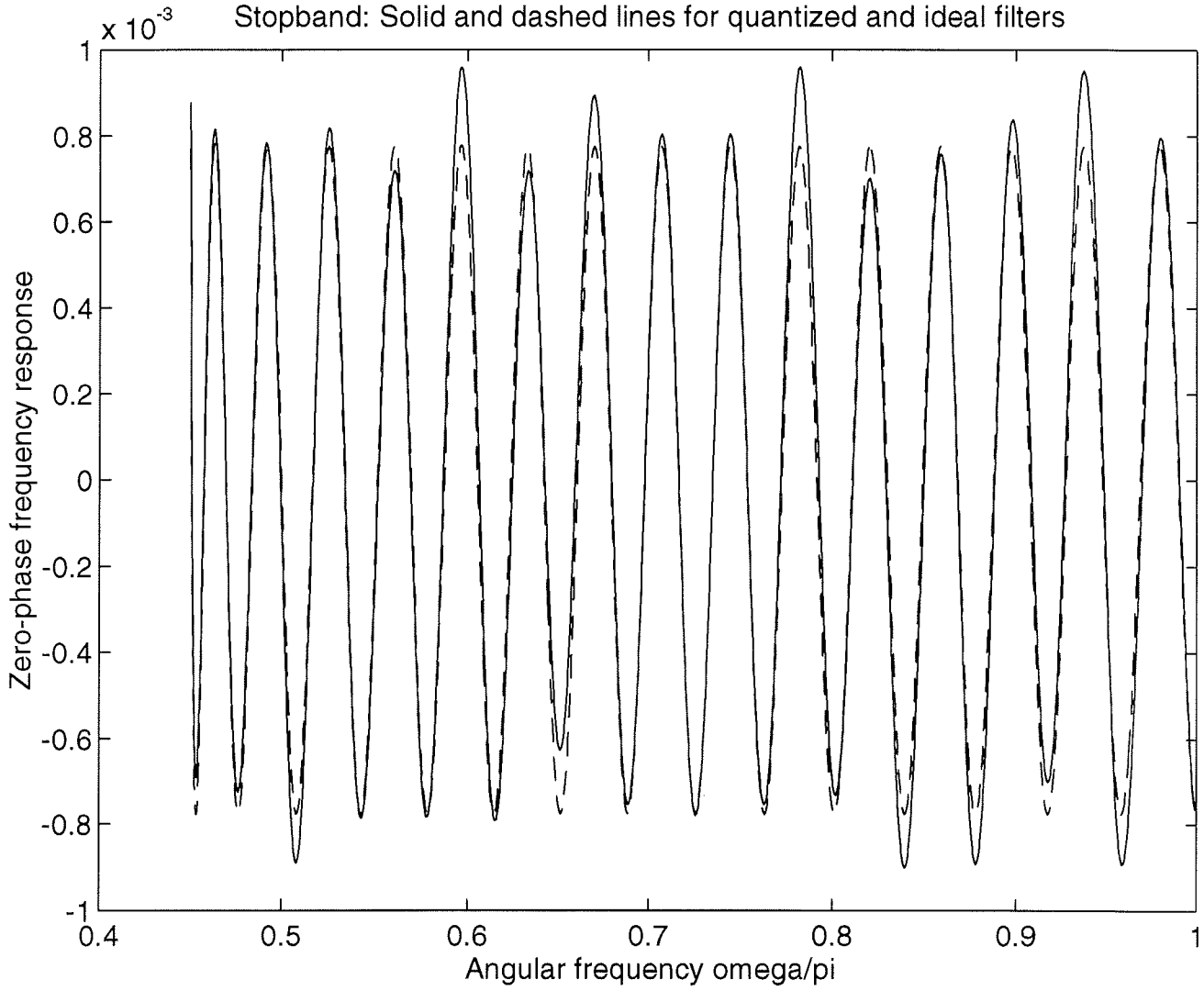
- For $b = 15$ fractional bits and $N = 104$, Eq. (C) gives $\delta_e \approx 0.0002$.
- Based on this, we first design a filter with ripples $\delta_p - \delta_e = 0.0098$ and $\delta_s - \delta_e = 0.0008$. The minimum order is $N = 106$.
- When the coefficients are rounded, 15 bits is in fact the minimum number of fractional bits needed for the zero-phase response to stay within the limits $1 \pm \delta_p = 1 \pm 0.01$ in the passband and within the limits $\pm\delta_s = \pm 0.001$ in the stopband.
- In the following there are again five figures giving the responses $H(\omega)$, $H_b(\omega)$, and $E_b(\omega)$. It is seen that the peak absolute value of $E_b(\omega)$, occuring in the passband region, is approximately 0.00035, which is larger than the estimated value of 0.0002. However, the filter with $b = 15$ fractional bits meets the criteria.


Solid and dashed lines for quantized and ideal filters
Amplitude in dB
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Passband: Ideal response H(omega)
1.01
1.005
1
0.995
0.99
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi
Zero-phase frequency response
Passband: Quantization error E_b(omega)
x 10^-4
4
2
0
-2
-4
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi

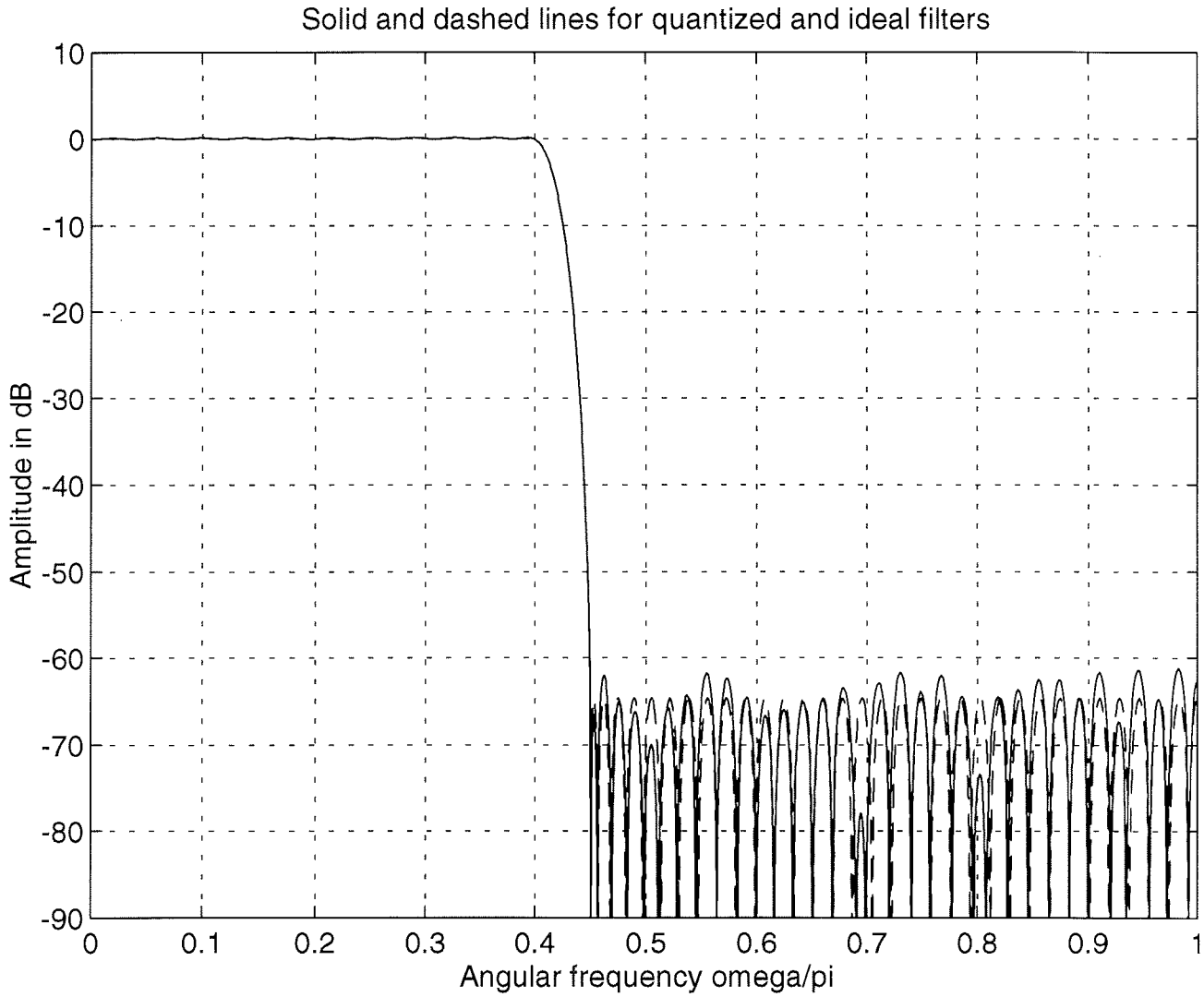

Passband: Solid and dashed lines for quantized and ideal filters
1.01
1.005
1
0.995
0.99
Zero-phase frequency response
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi

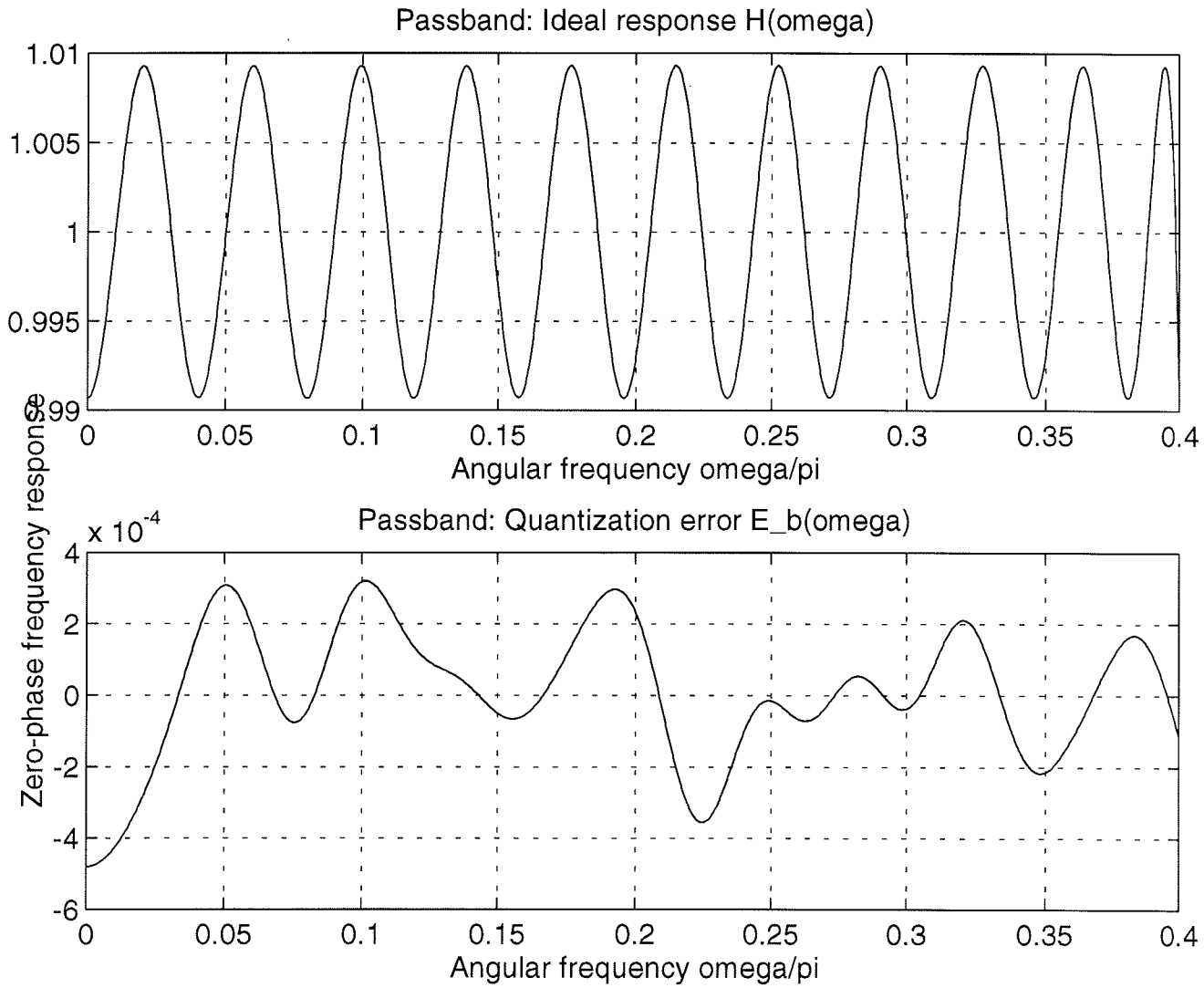

Stopband: Ideal response H(omega)
x 10^-3
1
0.5
0
-0.5
-1
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi
Stopband: Quantization error E_b(omega)
x 10^-4
2
1
0
-1
-2
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi
Zero-phase frequency response

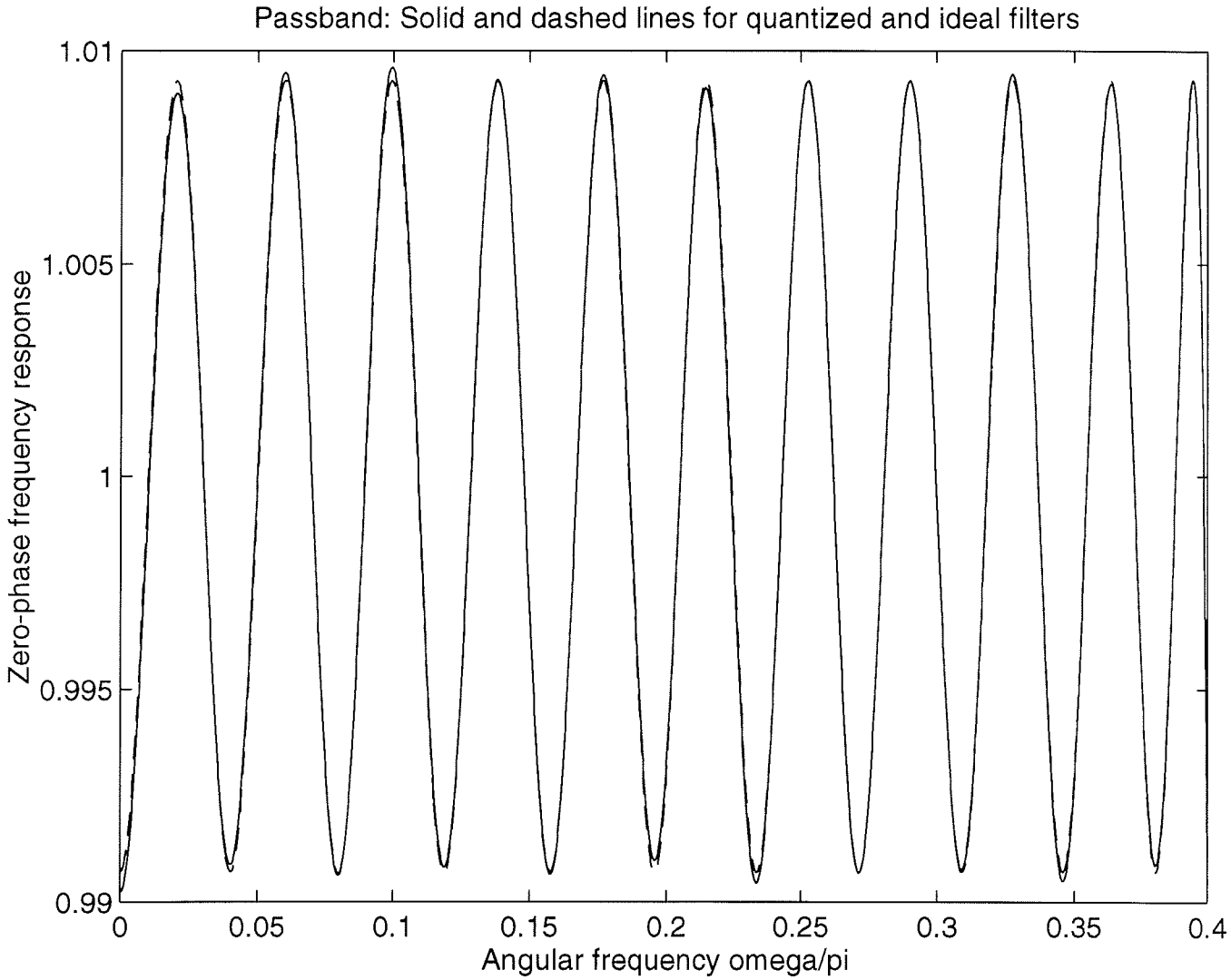

Stopband: Solid and dashed lines for quantized and ideal filters
x 10^-3
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Zero-phase frequency response
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

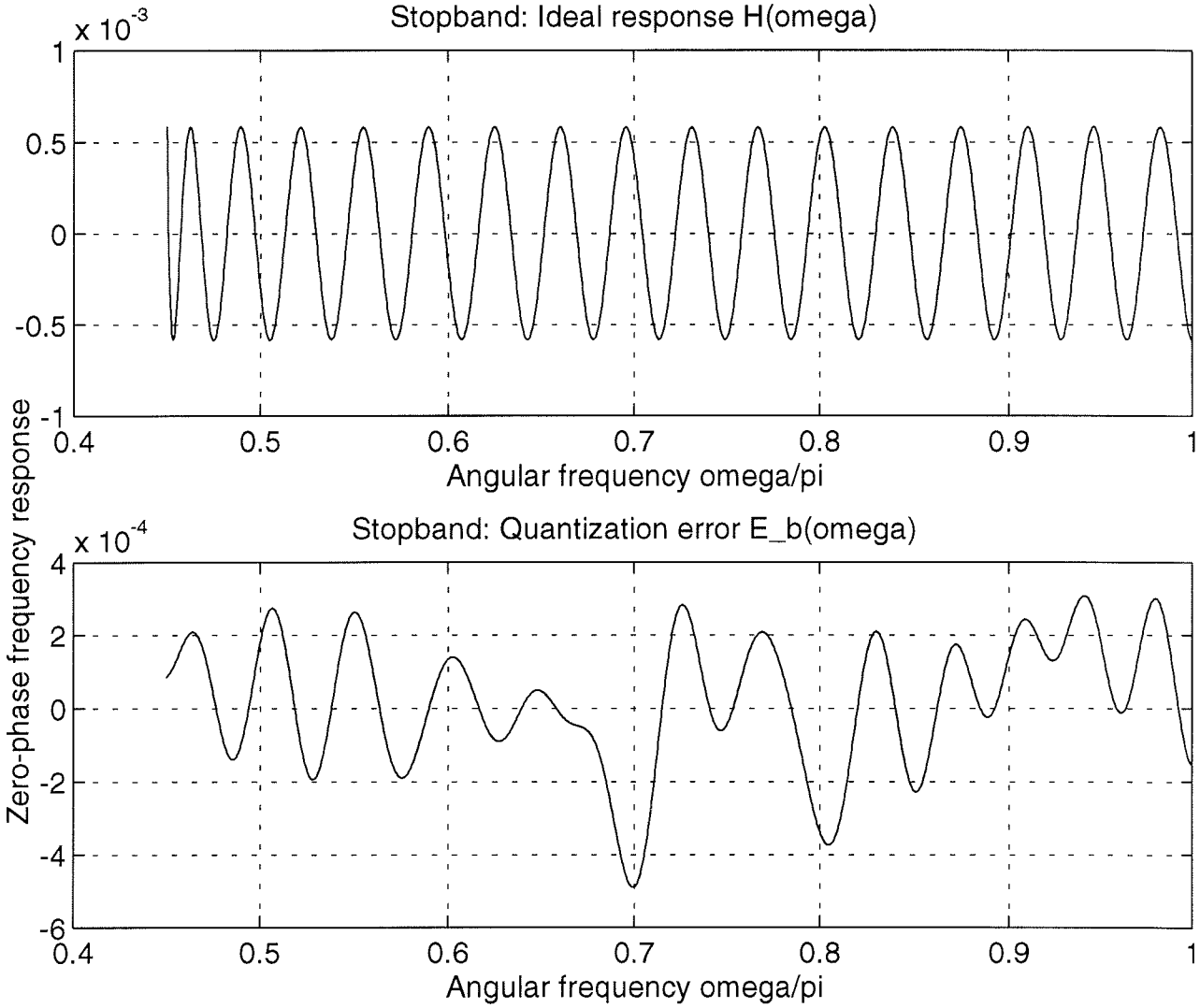
b =14 fractional bits

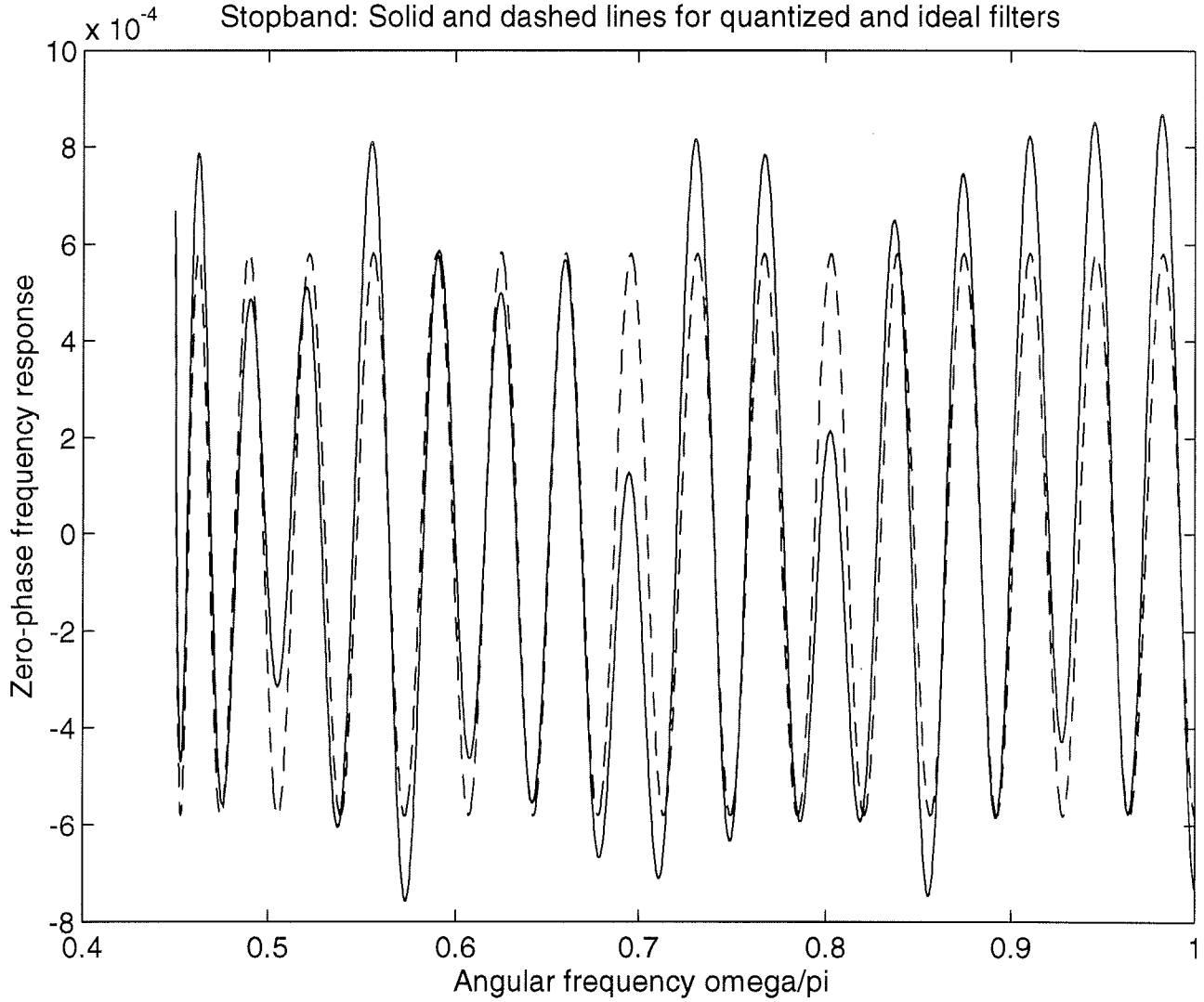
- For $b = 14$ fractional bits and $N = 104$, Eq. (C) gives $\delta_e \approx 0.0004$.
- Based on this, we first design a filter with ripples $\delta_p - \delta_e = 0.0096$ and $\delta_s - \delta_e = 0.0006$. The minimum order is $N = 110$.
- When the coefficients are rounded, 14 bits is in fact the minimum number of fractional bits needed for the zero-phase response to stay within the limits $1 \pm \delta_p = 1 \pm 0.01$ in the passband and within the limits $\pm\delta_s = \pm 0.001$ in the stopband.
- Again, there are five figures giving the responses $H(\omega)$, $H_b(\omega)$, and $E_b(\omega)$. It is seen that the peak absolute value of $E_b(\omega)$, occuring in the stopband region, is approximately 0.0005, which is very close to the estimated value of 0.0004.


Solid and dashed lines for quantized and ideal filters
Amplitude in dB
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Passband: Ideal response H(omega)
1.01
1.005
1
0.995
0.99
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi
Passband: Quantization error E_b(omega)
x 10^-4
4
2
0
-2
-4
-6
Zero-phase frequency response
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi

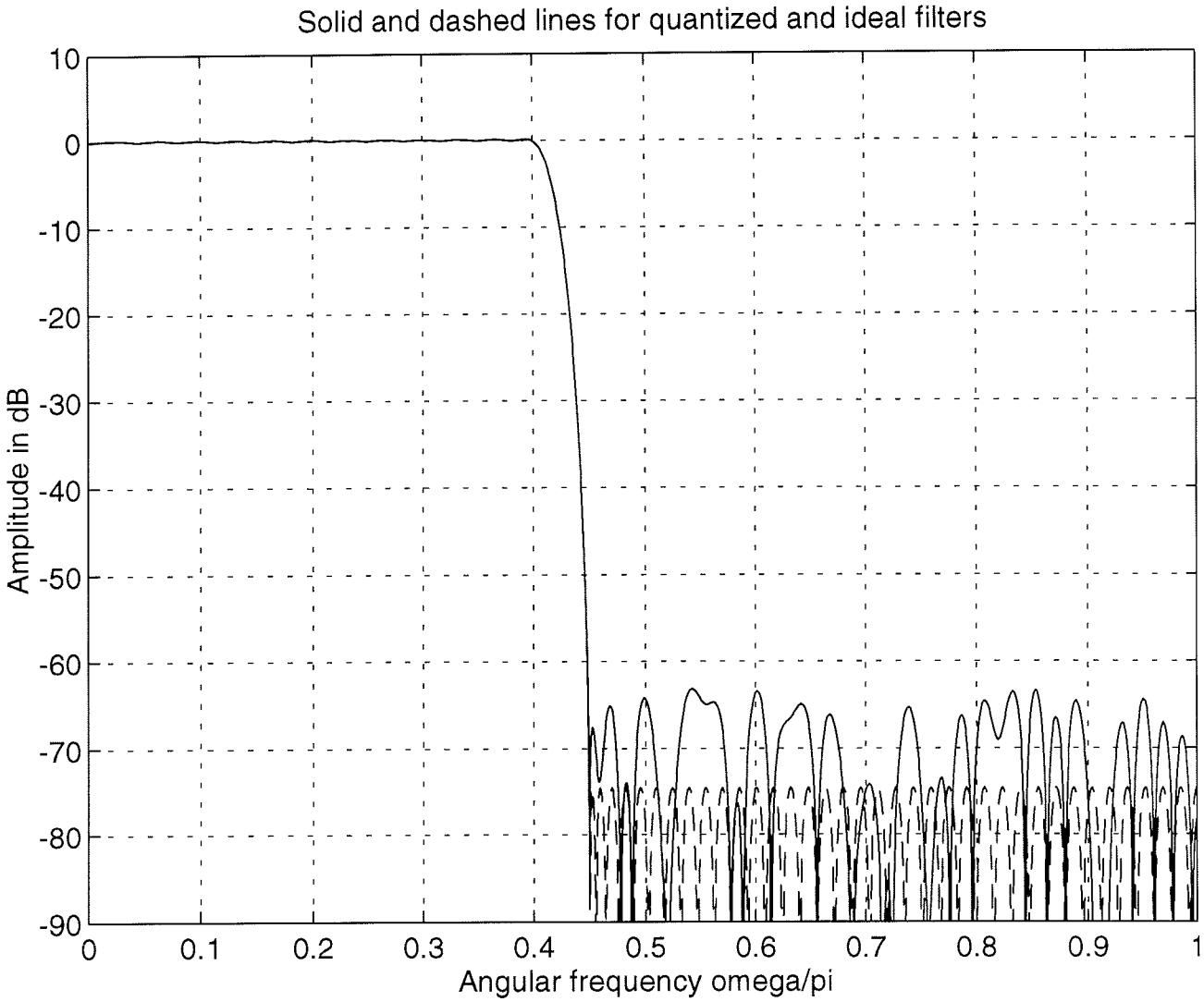

Passband: Solid and dashed lines for quantized and ideal filters
Zero-phase frequency response
1.01
1.005
1
0.995
0.99
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi

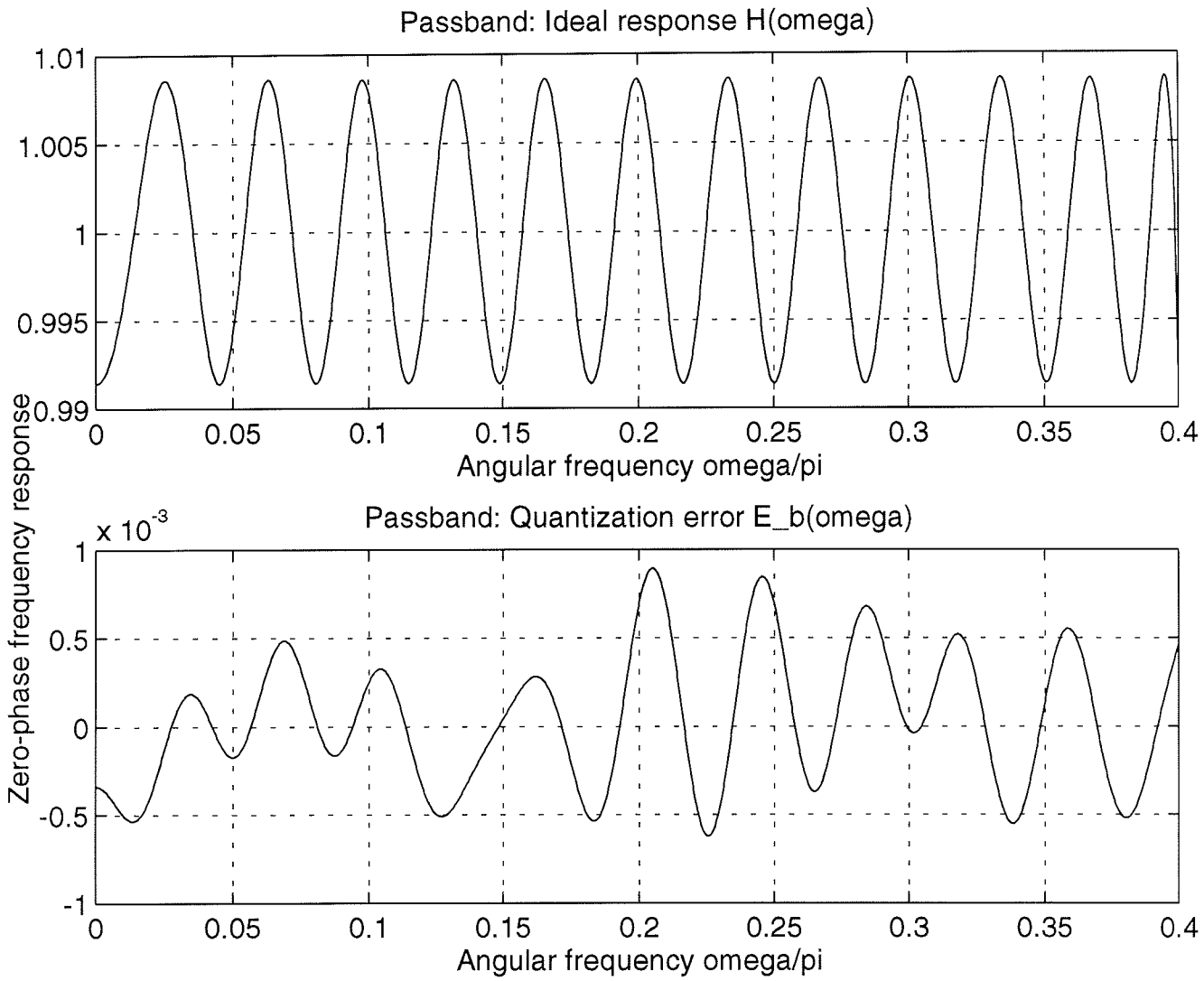

Stopband: Ideal response H(omega)
x 10^-3
1
0.5
0
-0.5
-1
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi
Stopband: Quantization error E_b(omega)
x 10^-4
Zero-phase frequency response
4
2
0
-2
-4
-6
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

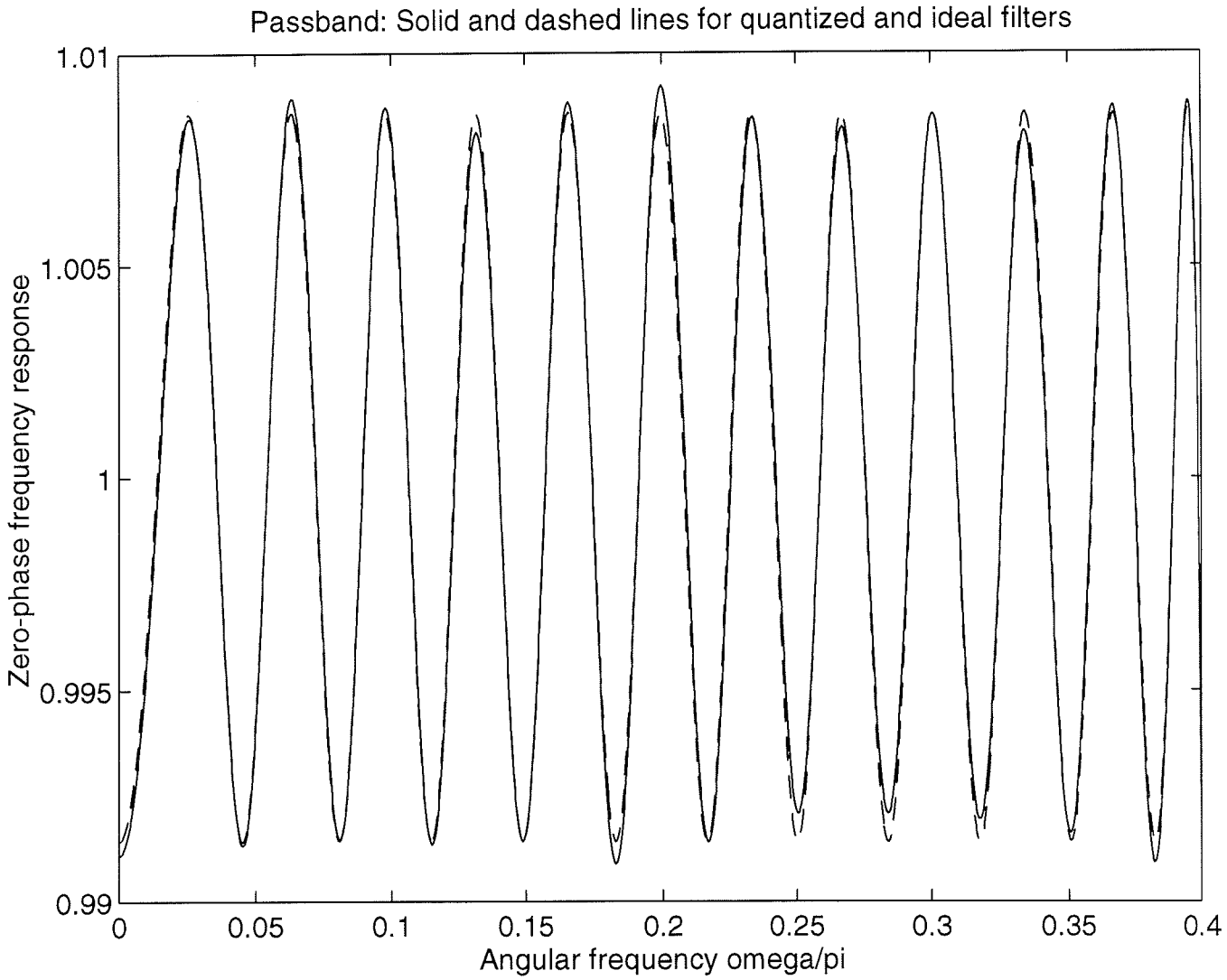

Stopband: Solid and dashed lines for quantized and ideal filters
x 10^-4
10
8
6
4
2
0
-2
-4
-6
-8
Zero-phase frequency response
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

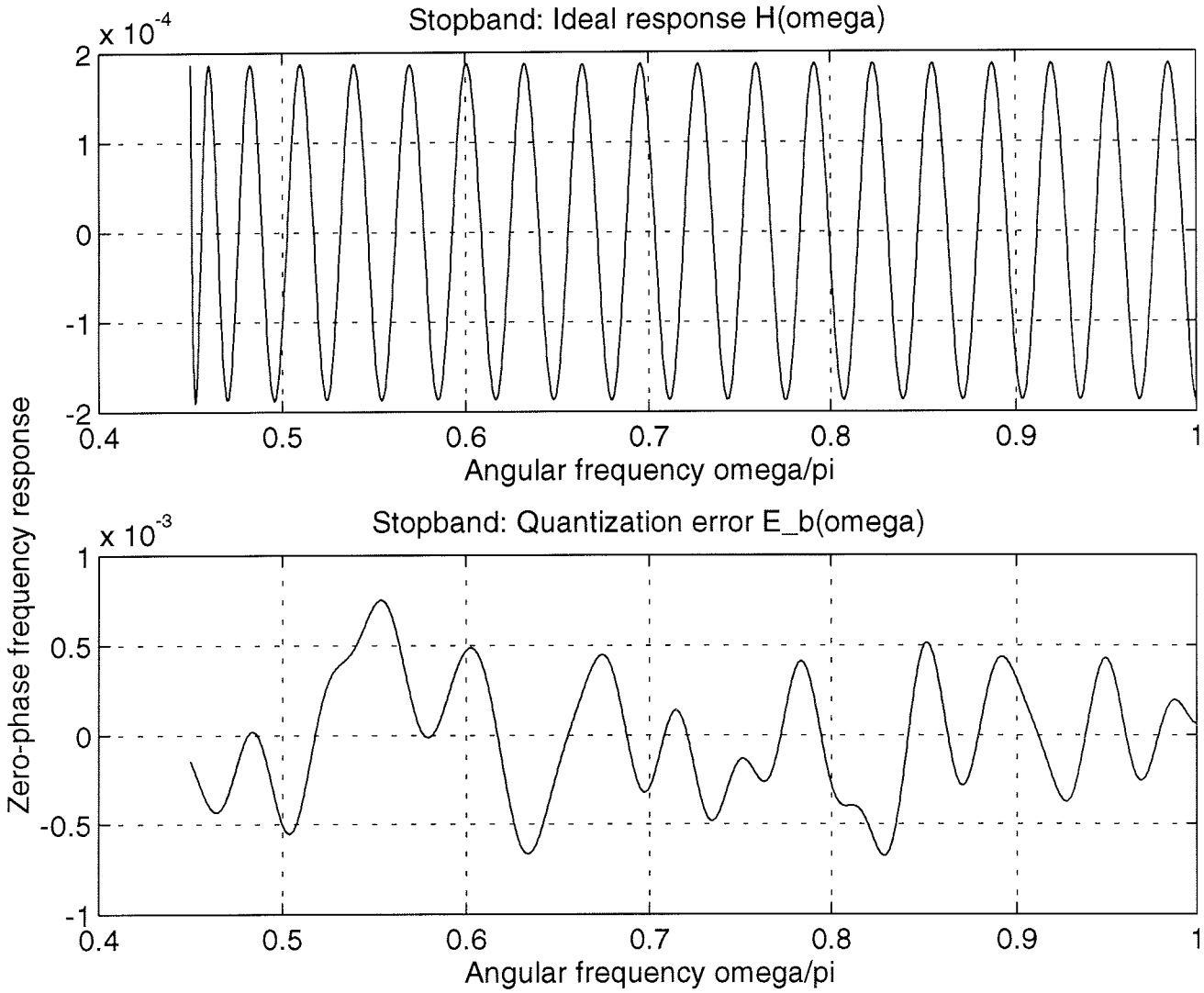
b =13 fractional bits

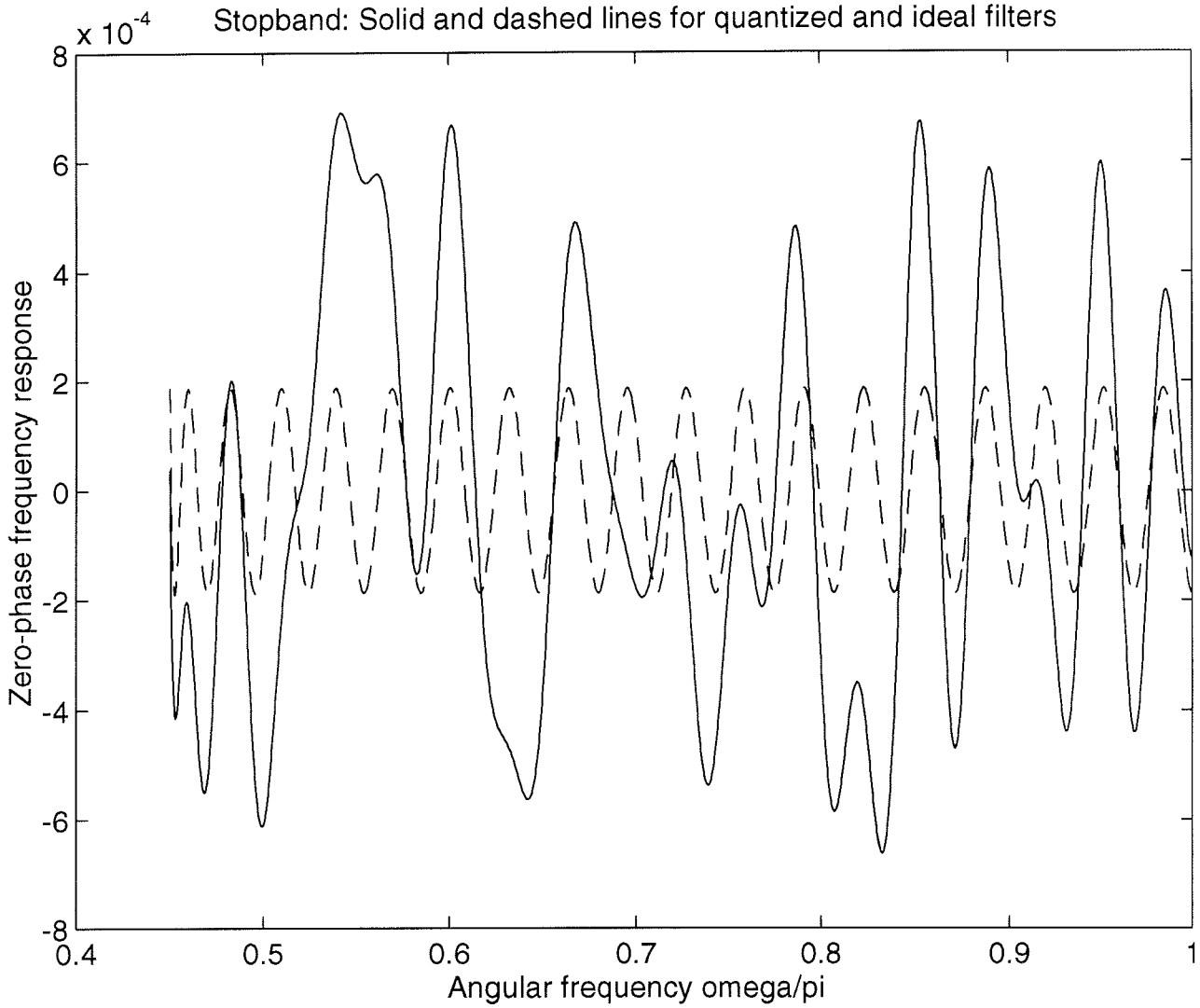
- For $b = 13$ fractional bits and $N = 104$, Eq. (C) gives $\delta_e \approx 0.0008$.
- Based on this, we first design a filter with ripples $\delta_p - \delta_e = 0.0092$ and $\delta_s - \delta_e = 0.0002$. The minimum order is $N = 122$.
- When the coefficients are rounded, 13 bits is in fact the minimum number of fractional bits needed for the zero-phase response to stay within the limits $1 \pm \delta_p = 1 \pm 0.01$ in the passband and within the limits $\pm\delta_s = \pm 0.001$ in the stopband.
- Again, there are five figures giving the responses $H(\omega)$, $H_b(\omega)$, and $E_b(\omega)$. It is seen that the peak absolute value of $E_b(\omega)$ is approximately the estimated value of 0.0008.


Solid and dashed lines for quantized and ideal filters
Amplitude in dB
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Passband: Ideal response H(omega)
1.01
1.005
1
0.995
0.99
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi
Passband: Quantization error E_b(omega)
x 10^-3
1
0.5
0
-0.5
-1
Zero-phase frequency response
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi


Passband: Solid and dashed lines for quantized and ideal filters
Zero-phase frequency response
1.01
1.005
1
0.995
0.99
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Angular frequency omega/pi


Stopband: Ideal response H(omega)
x 10^-4
2
1
0
-1
-2
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi
Stopband: Quantization error E_b(omega)
x 10^-3
1
0.5
0
-0.5
-1
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi
Zero-phase frequency response


Stopband: Solid and dashed lines for quantized and ideal filters
x 10^-4
8
6
4
2
0
-2
-4
-6
-8
Zero-phase frequency response
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

b =12 fractional bits

- For $b = 12$ fractional bits and $N = 104$, Eq. (C) gives $\delta_e \approx 0.0016$.
- $\delta_e > \delta_s = 0.001$. This shows that it is unlikely to meet the given criteria with $b = 12$ fractional bits.

COMMENTS

- In the above, we gave a simple procedure for finding FIR filters with quantized coefficients. The results followed probably too well the theory. This is not always the case.
- In practice, you can try several values of δ_e, not just those obtained by using Eq. (C).
- Finally, it should be noted that there exist several optimization routines for quantizing the FIR filter coefficients. These routines give better results than direct rounding.
- If you are interested in these algorithms, please contact the lecturer.

```
% Matlab m-file (firquan.m)
% for designing a multiband  FIR filter using the
% the Remez algorithm and then quantizing the
% coefficients to the given number of fractional
% bits
% can be found in ~ts/matlab/sldsp
%
% Tapio Saramaki  16.10.95

clear all
close all
disp('I am a program for designing a multiband FIR')
disp('filter with the aid of the Remez routine.')
disp('I assume that the desired function as well')
disp('as the permissible error in each band are')
disp('constants')
nbands=input('Number of bands: ');
for k=1:nbands
  disp(' ')
  fprintf('Band number %g.\',k)
  disp(' ')
  wc(2*k-1)=input('Lower edge as a fraction of pi: ');
  wc(2*k)=input('Upper edge as a fraction of pi: ');
  m(k)=input('Desired constant in the band: ');
  d(k)=input('Permissible deviation in the band: ');
end
  for k=1:2*nbands-2
  f(k)=wc(k+1);end
  dev=d;
  ma=m;
  end
[N,fo,mo,w] = remezord(f,ma,dev);
fprintf('Estimated order is %g.\',N)
for k=1:nbands
w(k)=1/dev(k);
end
l=0;
while l < 1
N=input('Your  selection for N:  ')
h = remez(N,fo,mo,w);
nbit=input('Number of bits: ');
hs=round(h*2^nbit)/(2^nbit);
```

```
figure(1)
[H,W]=zeroam(h,.0,1.,2000);
[H1,W1]=zeroam(hs,.0,1.,2000);
plot(W/pi,20*log10(abs(H)),'- -',W/pi,20*log10(abs(H1)));
axis([0 1 -90 10]);grid;
ylabel('Amplitude in dB'); xlabel('Angular frequency omega/pi');
title('Solid and dashed lines for quantized and ideal filters');
[H,W]=zeroam(h,fo(1),fo(2),2000);
[H1,W1]=zeroam(hs,fo(1),fo(2),2000);
figure(2)
subplot(211)
plot(W/pi,H);grid
xlabel('Angular frequency omega/pi');
title('Passband: Ideal response H(omega)');
subplot(212)
plot(W/pi,H1-H);
grid;
ylabel('                    Zero-phase frequency response');
xlabel('Angular frequency omega/pi');
title('Passband: Quantization error E_b(omega)');
figure(3)
plot(W/pi,H,'- -',W/pi,H1);
title('Passband: Solid and dashed lines for quantized and ideal
filters');
ylabel('Zero-phase frequency response');
xlabel('Angular frequency omega/pi');
figure(4)
[H,W]=zeroam(h,fo(3),fo(4),2000);
[H1,W1]=zeroam(hs,fo(3),fo(4),2000);
subplot(211)
plot(W/pi,H);grid
xlabel('Angular frequency omega/pi');
title('Stopband: Ideal response H(omega)');
subplot(212)
plot(W/pi,H1-H);
grid;
ylabel('                    Zero-phase frequency response');
xlabel('Angular frequency omega/pi');
title('Stopband: Quantization error E_b(omega)');
figure(5)
plot(W/pi,H,'- -',W/pi,H1);
```

```
title('Stopband: Solid and dashed lines for quantized and ideal
filters');
ylabel('Zero-phase frequency response');
xlabel('Angular frequency omega/pi');
l=input('0 for continue');
end
```

FINITE WORDLENGTH EFFECTS IN IIR FILTERS

- In this pile of transparencies we consider the following topics:

1) The effect of coefficient rounding for cascaded-form IIR filters consisting of direct-form II blocks.

2) The effect of coefficient rounding for IIR filters implemented as a parallel connection of two allpass sections.

3) The validity of the model developed for estimating the output noise performance due to the multiplication roundoff errors.

- We refer several times to the material in the course DIGITAL FILTERING II especially to the lectures notes FINITE WORDLENGTH EFFECTS IN DIGITAL FILTERS and DESIGN OF RECURSIVE FILTERS USING ALLPASS FILTERS AS BUILDING BLOCKS.

- If this material is not available, please contact the lecturer.

EFFECTS OF COEFFICIENT ROUNDING FOR CACADED-FORM IIR FILTERS

- We consider these effects in terms of the following problem:
- What is the minimum number of fractional bits required for a sixth-order IIR filter to meet the criteria?:
  - In the passband $[0, 0.2\pi]$ the amplitude response stays between 0 dB and 0.5 dB.
  - In the stopband $[0.3\pi, \pi]$ the minimum attenuation is at least 60 dB.
- When solving this problem the following assumptions are made:

1) We start with a sixth-order filter which is properly designed (to be explained later.)

2) The filter is implemented in the form of page 45 in the lecture notes FINITE WORDLENGTH EFFECTS IN DIGITAL FILTERS and the L_∞-norm scaling is used.

3) The second-order blocks are constructed according to the rules of pages 77 and 78 in the same lecture notes. The block with the pole closest to the unit circle is implemented last.

4) All the coefficients are rounded to the given number of fractional bits, denoted by b later.

5) When plotting the resulting response, it is scaled such that the maximum value in the passband is 0 dB.

- It practice, it is not so important what is the actual level in the passband.

- In the end of this pile of lecture notes you can find a Matlab-file iircoef.m, which is automatically taking care of assumptions 2)–5). It can be used also for scaling the filter in other ways. This file can be found in the end of this pile of lecture notes.
- What is now left is to find a proper starting point elliptic filter.
- This is because when designing an elliptic filter using a standard Matlab-file, it automatically makes

the stopband edge as small as possible such that the given passband and stopband ripples are just met.

- This means that there is no space for an error caused by coefficient quantization.
- To get around this problem, the lecturer has generated a Matlab-file ordel.m, which will considered next in more details and can also be found in this pile.

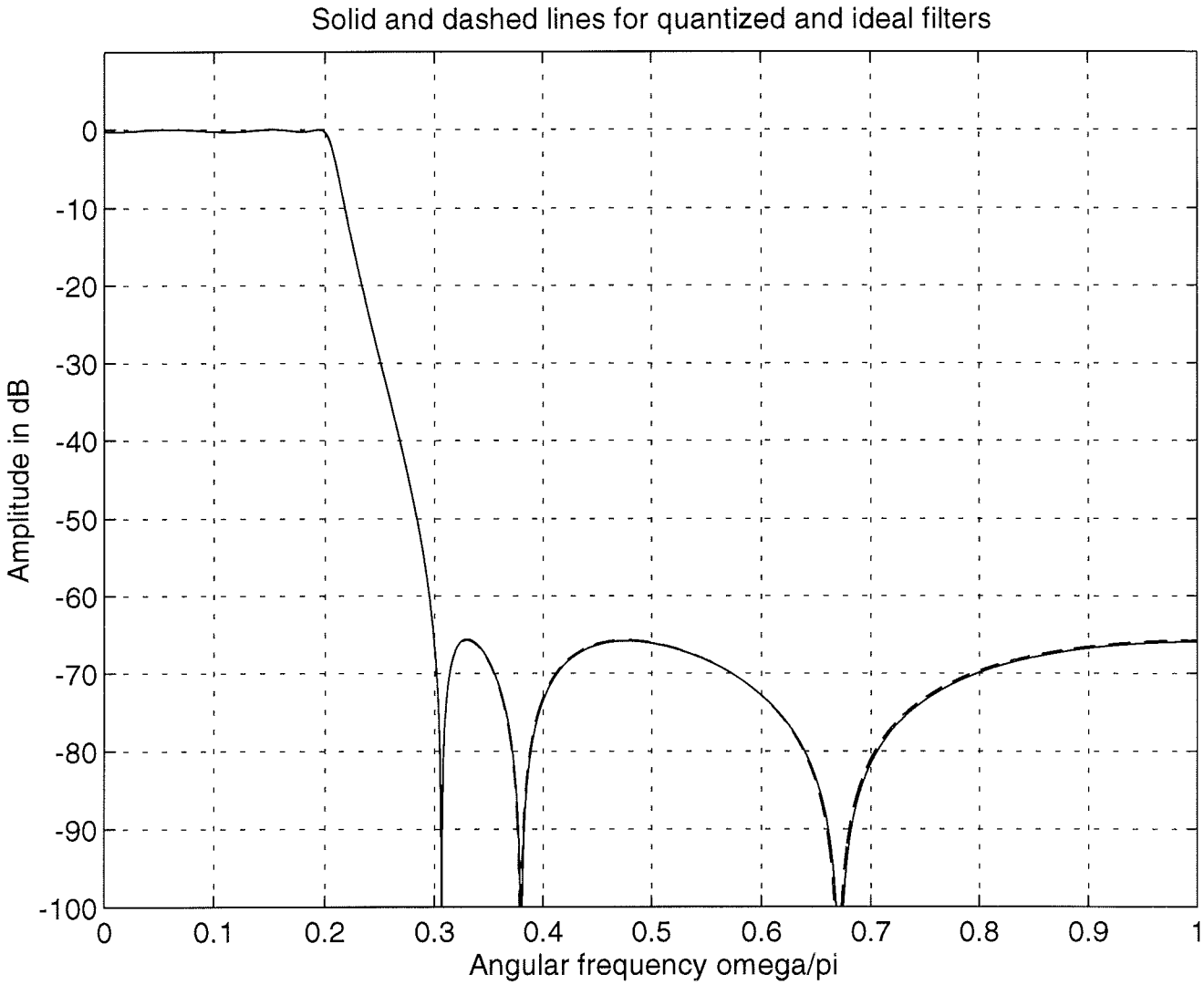
Matlab-file ordel.m for analysing the performance of elliptic filters

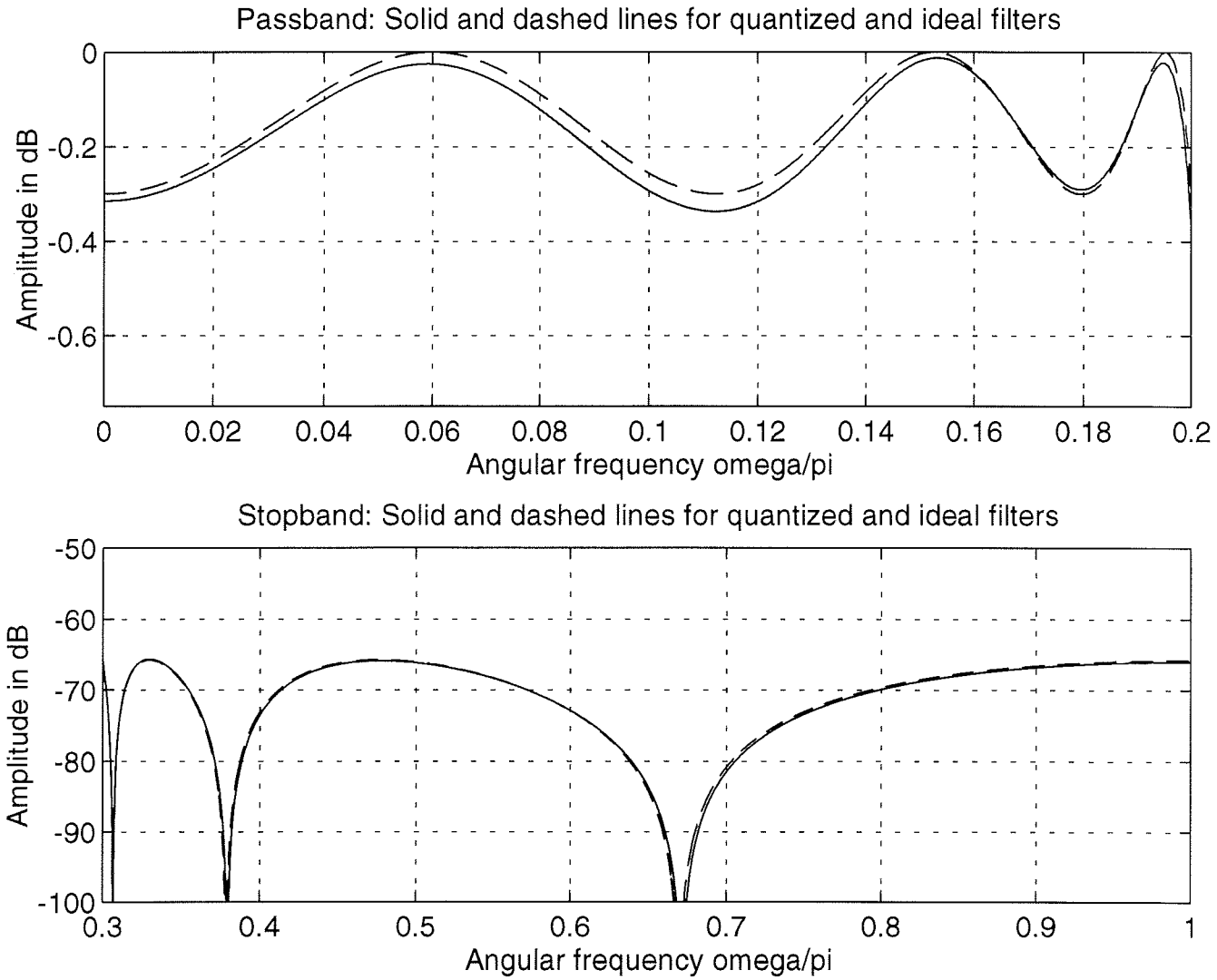
- The first two input parameters are the passband edge ω_p and the stopband edge ω_s, both given as fractions of π.
- Then the use drops out one of the following three parameters: the filter order N, the passband ripple A_p in dB, and the minimum stopband attenuation A_s in dB.
- The program then evaluates the remaining parameter for the corresponding elliptic filter.
- When N is dropped out, and $\omega_p = 0.2\pi$, $\omega_s = 0.3\pi$, $A_p = 0.5$, and $A_s = 60$, we get $N = 5.4588$. This means that $N = 6$ is the minimum order to meet the given criteria.
- In the following, we keep $\omega_p = 0.2\pi$ and $\omega_s = 0.3\pi$ fixed.
- When A_p is dropped out and $A_s = 60$, we get $A_p = 0.0832$.

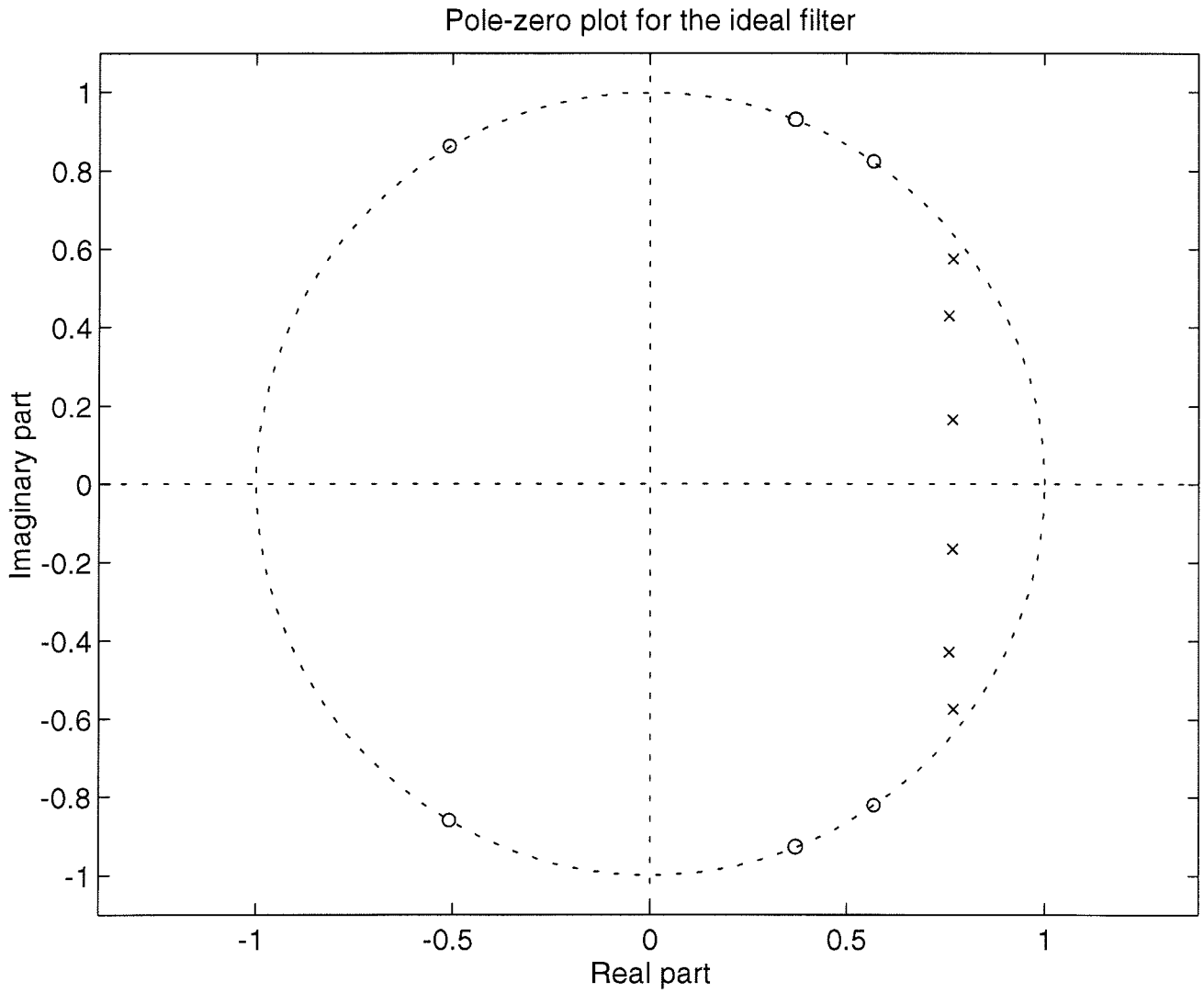
- For $A_p = 0.3,\ A_s = 65.72$. For $A_p = 0.2,\ A_s = 63.92$. For $A_p = 0.1,\ A_s = 60.85$.
- The input parameters of the Matlab-file iircoef.m consists of ω_p, A_p, and A_s.
- The basic advantage of the Matlab-file ordel.m is that if we give the above A_p and A_s pairs, then the stopband edge is automatically $\omega_s = 0.3\pi$.
- Now we are ready to test the above A_p and A_s pairs.

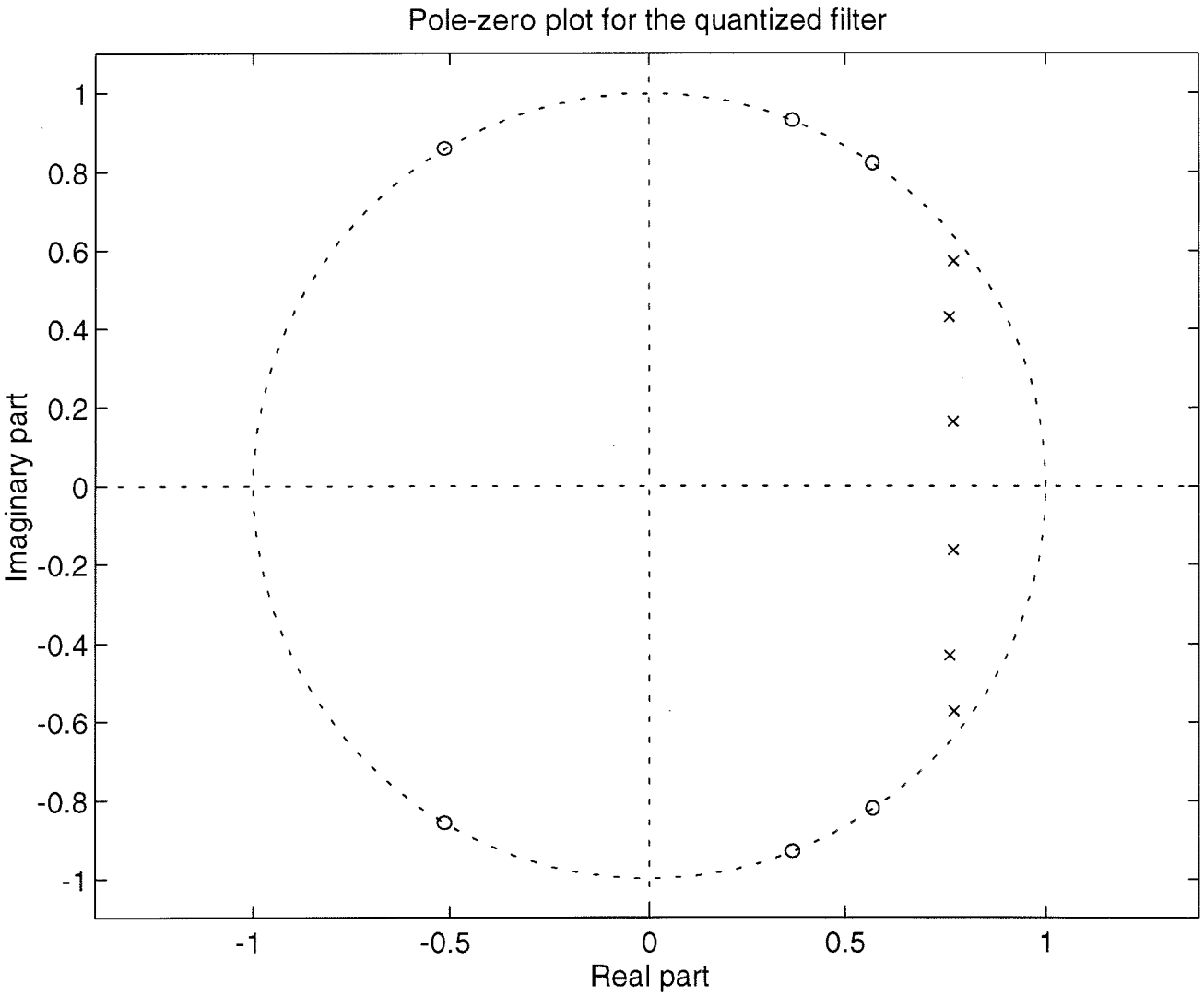
$A_p = 0.3,\ A_s = 65.72$

- In this case $b = 9$ fractional bits are needed for the passband ripple to be smaller than or equal to 0.5 dB and the minimum stopband attenuation to be at least 60 dB.
- The following four transparencies compare this design with the infinite-precision design.


Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

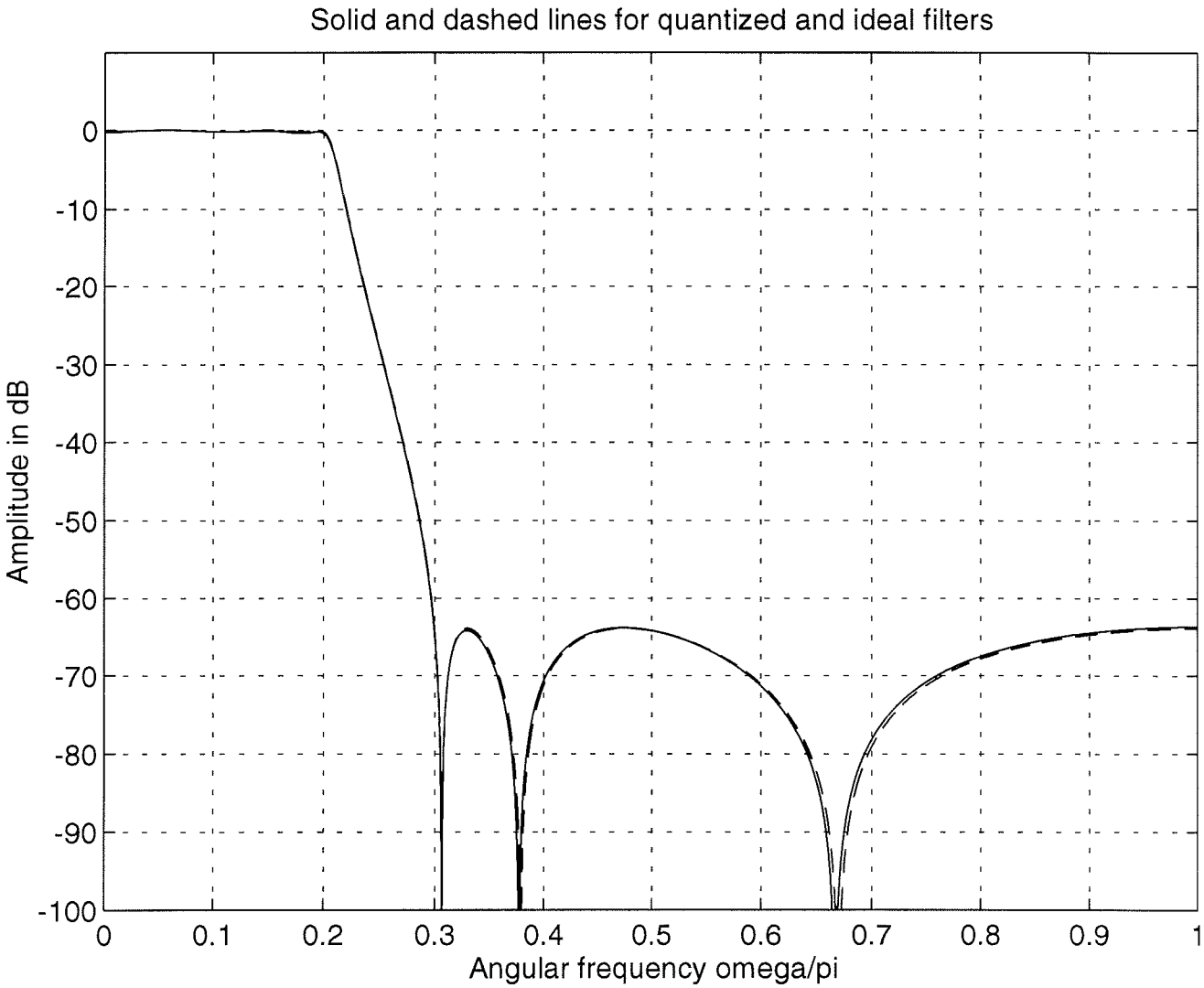

Passband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-0.2
-0.4
-0.6
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
Angular frequency omega/pi
Stopband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
-50
-60
-70
-80
-90
-100
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

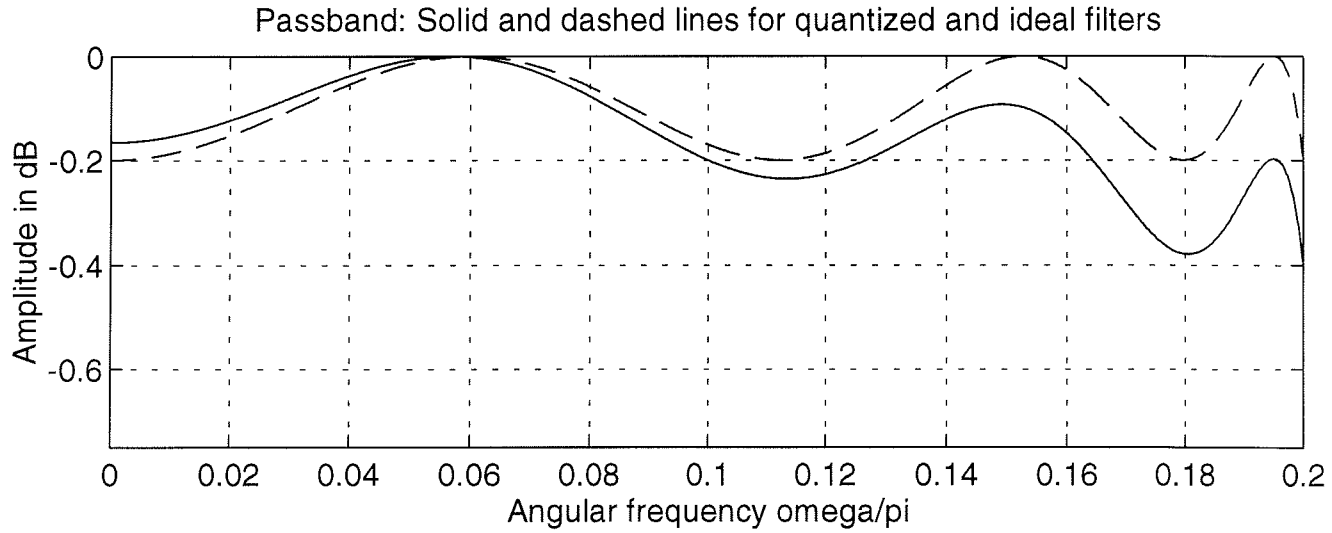

Pole-zero plot for the ideal filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

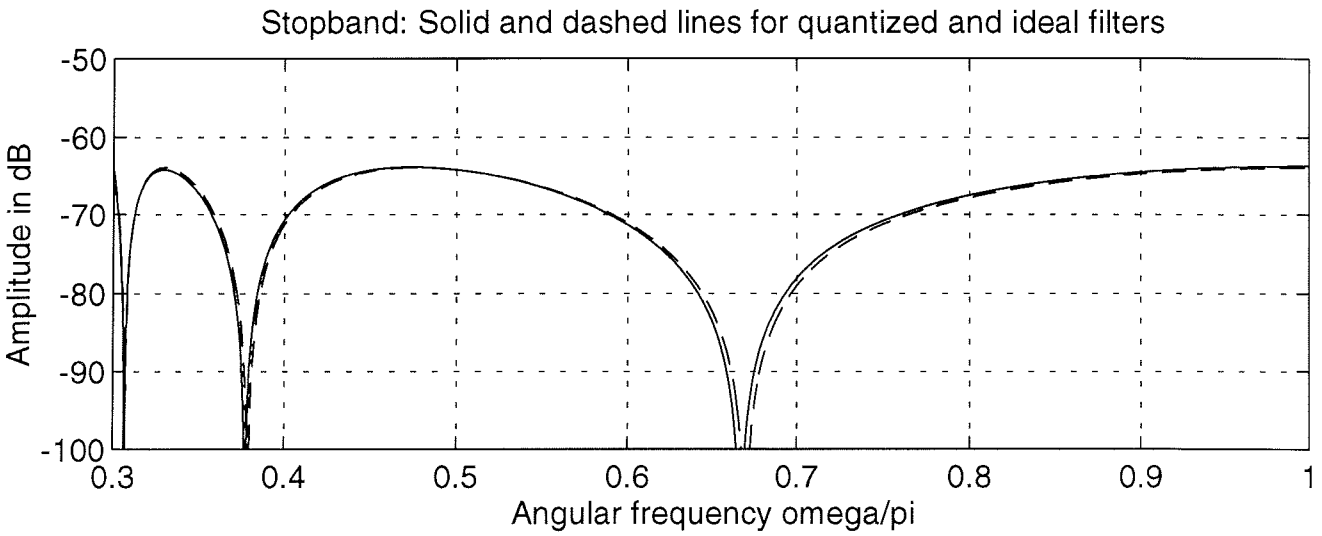

Pole-zero plot for the quantized filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

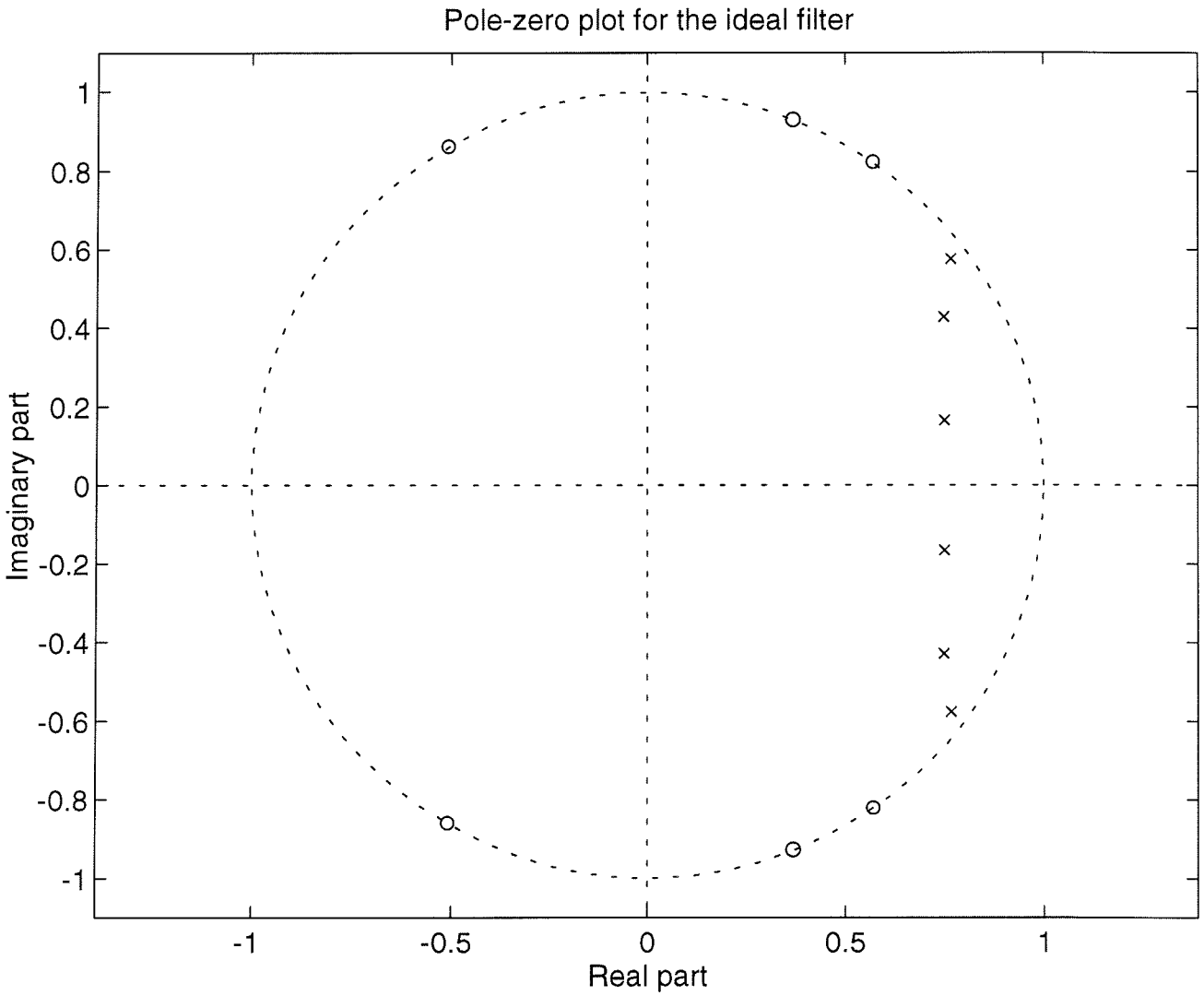
$A_p = 0.2,\ A_s = 63.92$

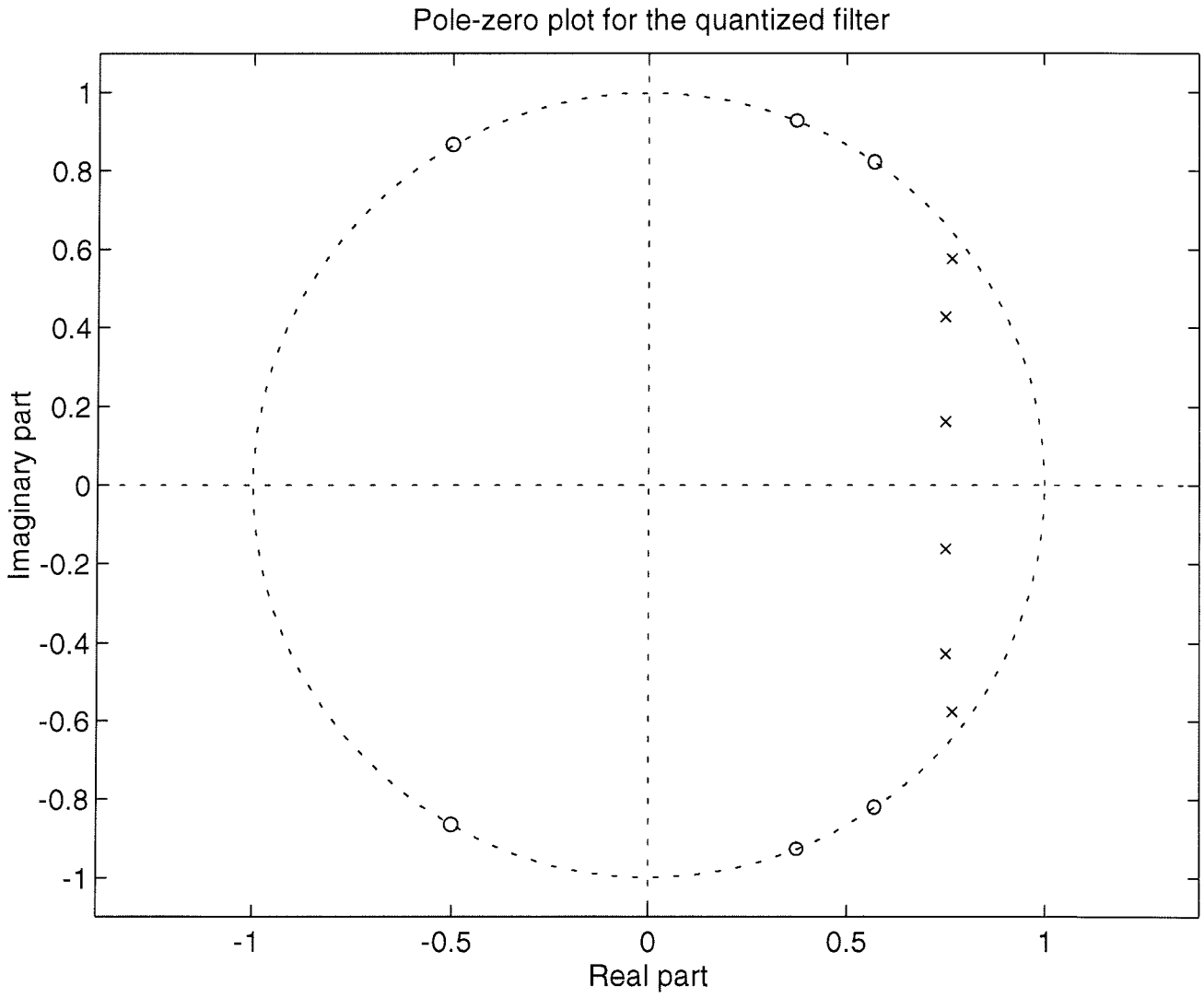
- In this case $b = 8$ fractional bits are needed for the passband ripple to be smaller than or equal to 0.5 dB and the minimum stopband attenuation to be at least 60 dB.
- The following four transparencies compare this design with the infinite-precision design.

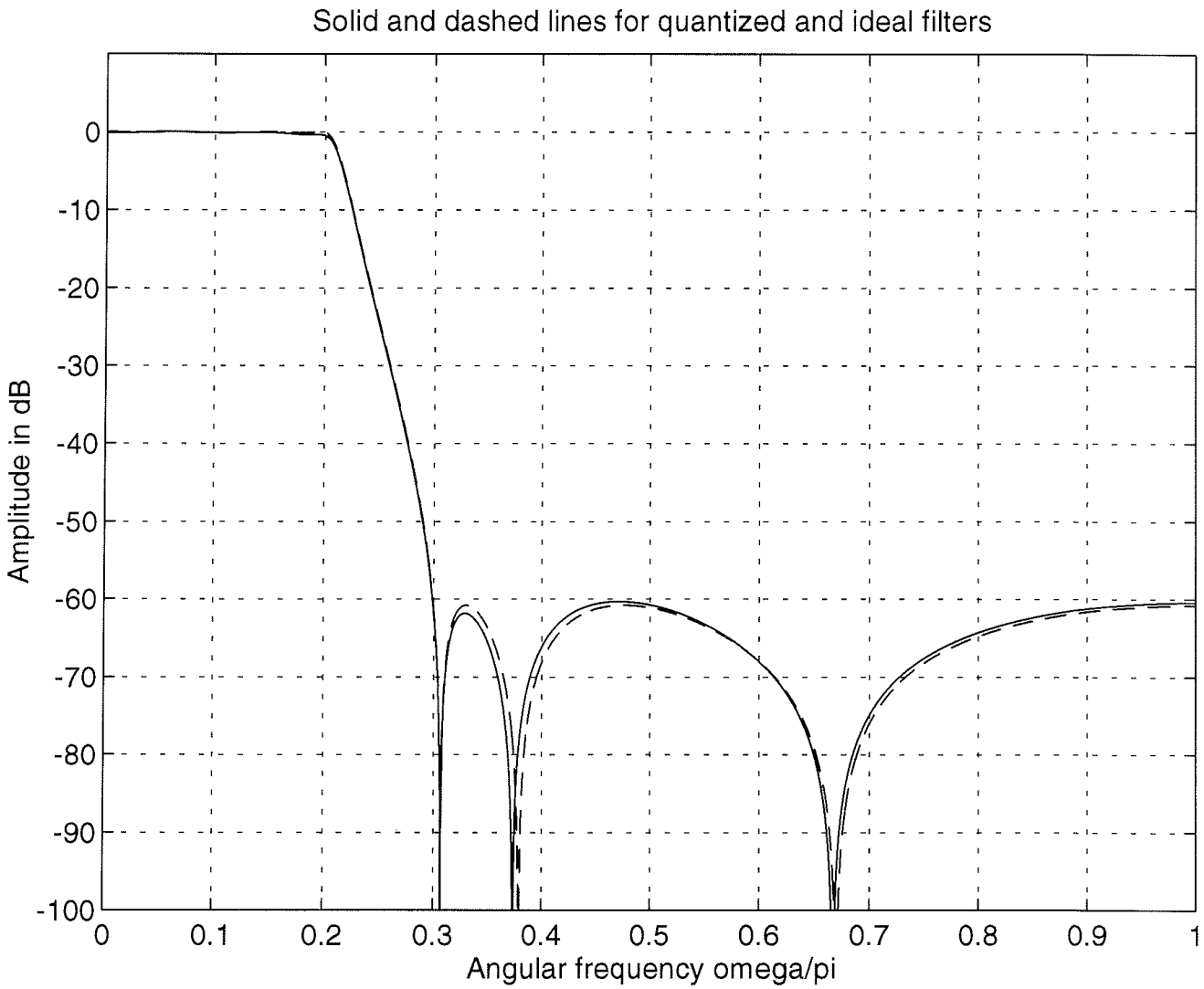

Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Passband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-0.2
-0.4
-0.6
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
Angular frequency omega/pi


Stopband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
-50
-60
-70
-80
-90
-100
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

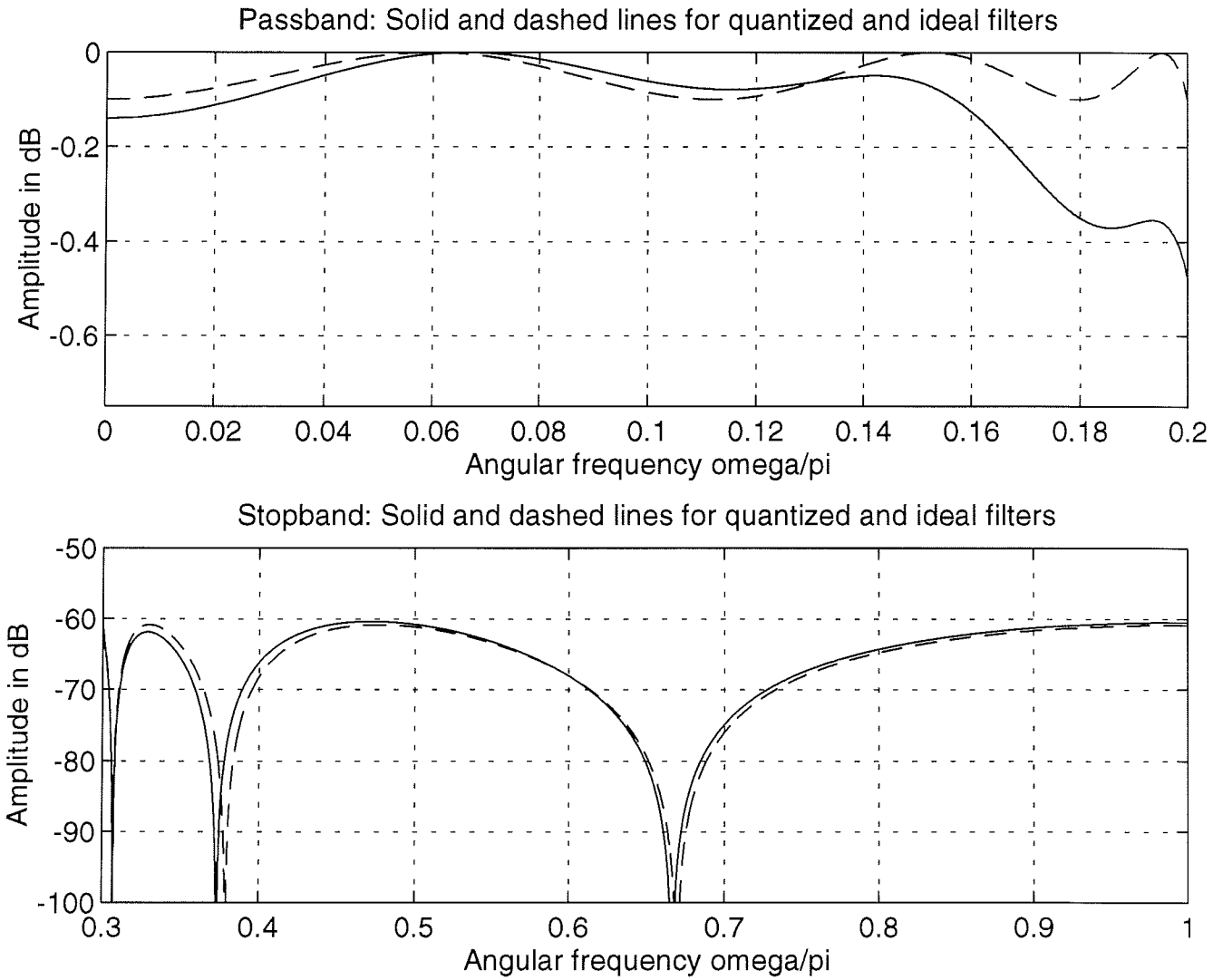

Pole-zero plot for the ideal filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

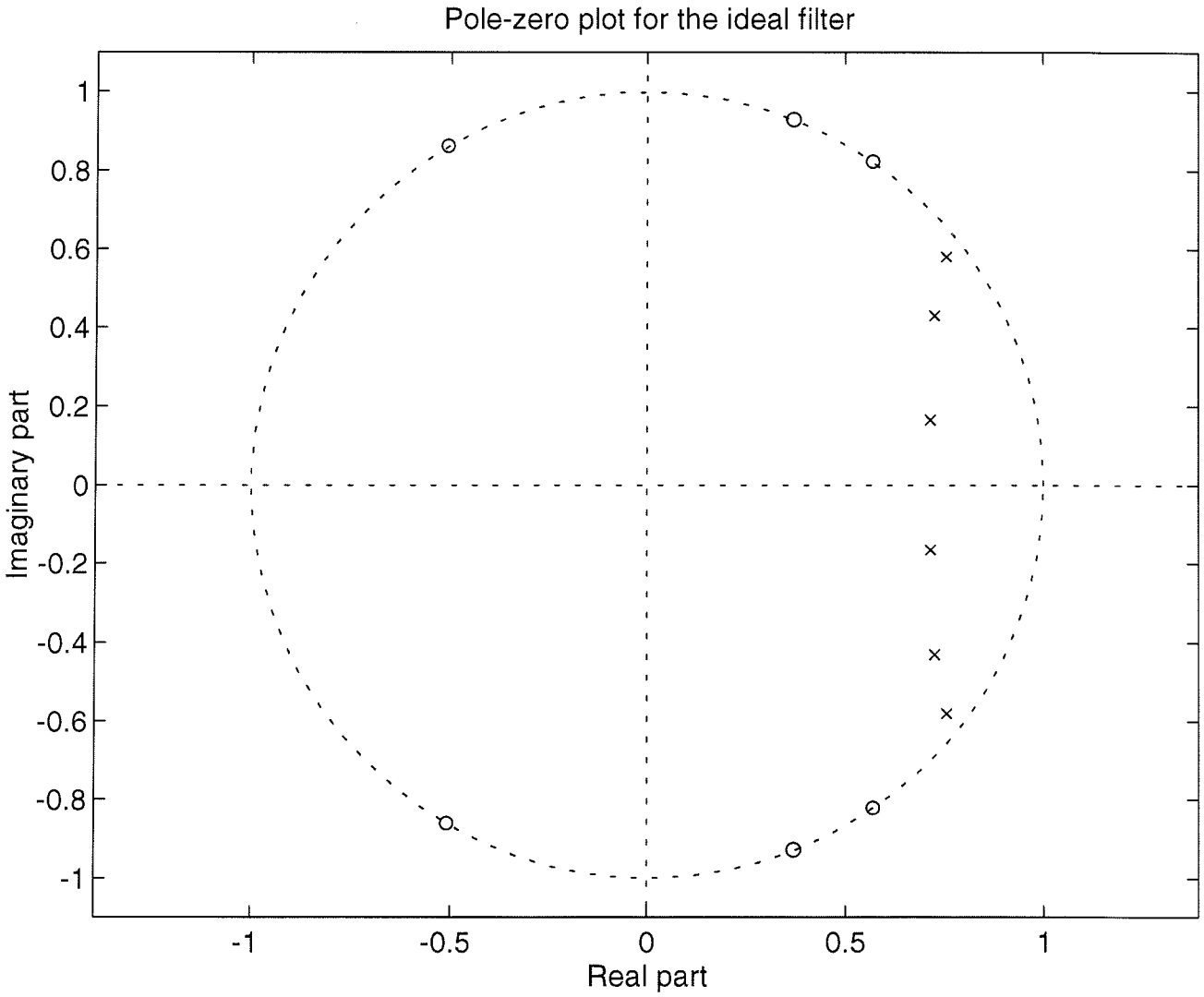

Pole-zero plot for the quantized filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

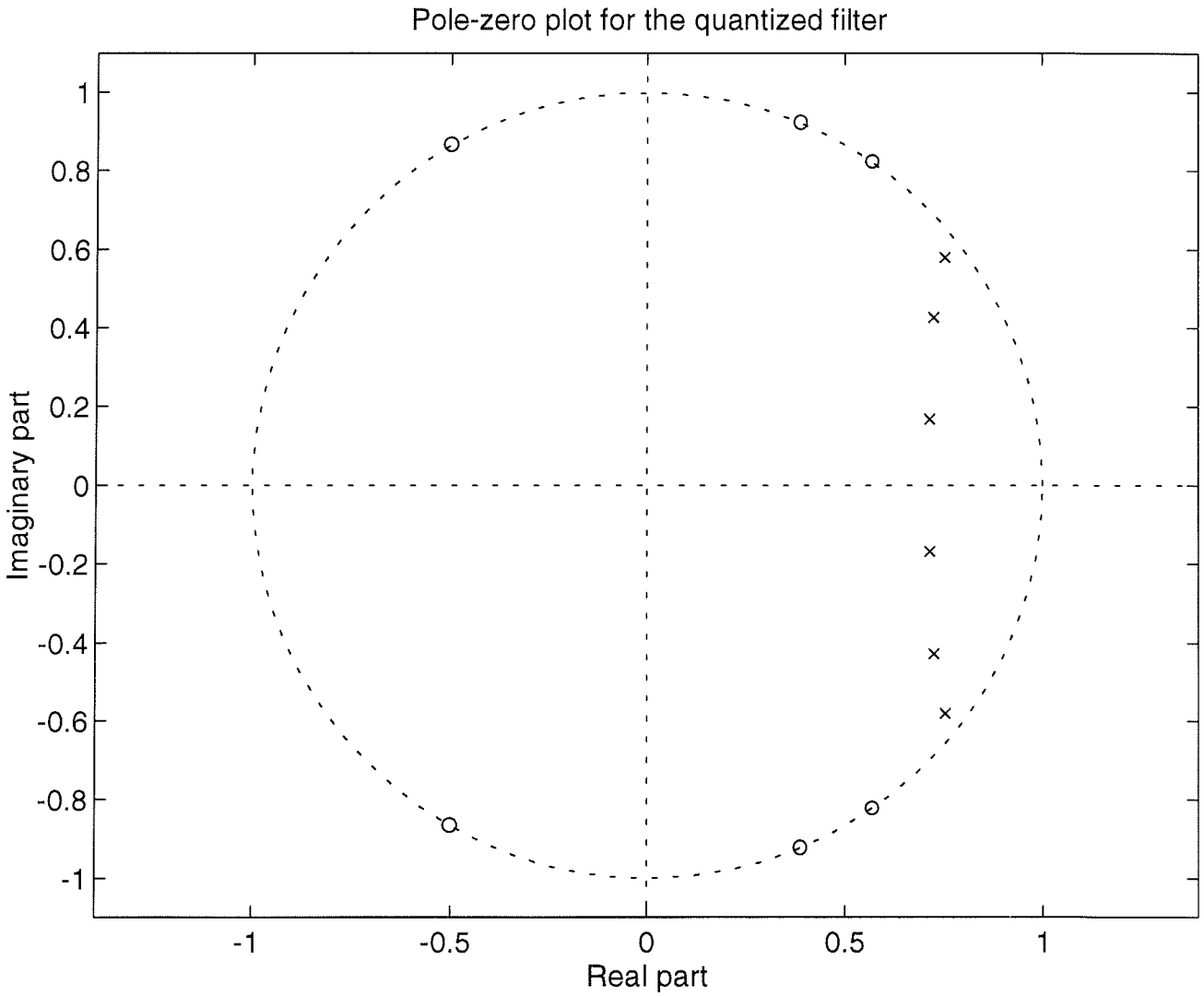

Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

$A_p = 0.1,\ A_s = 60.85$

- In this case $b = 7$ fractional bits are needed for the passband ripple to be smaller than or equal to 0.5 dB and the minimum stopband attenuation to be at least 60 dB.
- The following four transparencies compare this design with the infinite-precision design.

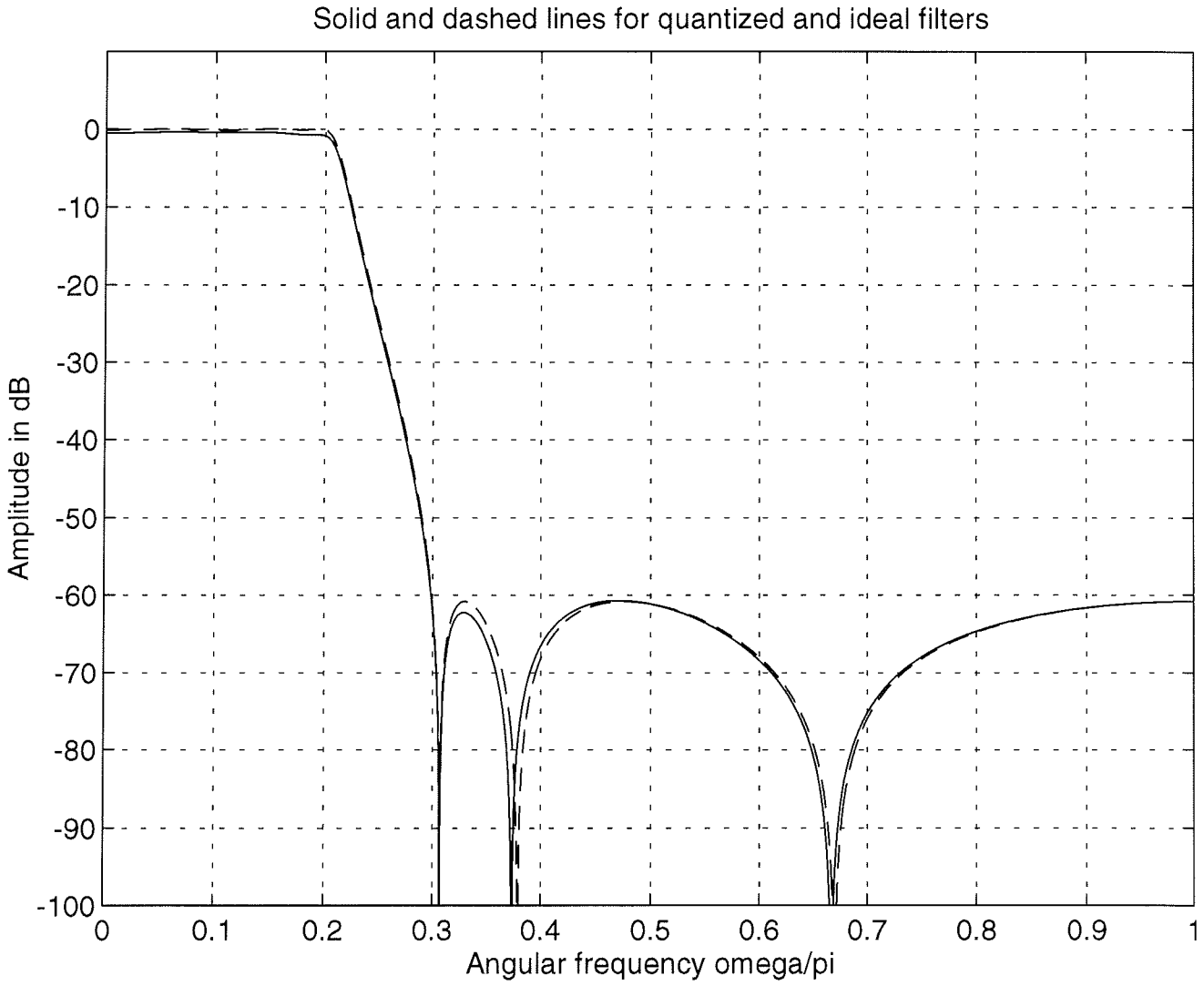

Passband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-0.2
-0.4
-0.6
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
Angular frequency omega/pi
Stopband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
-50
-60
-70
-80
-90
-100
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

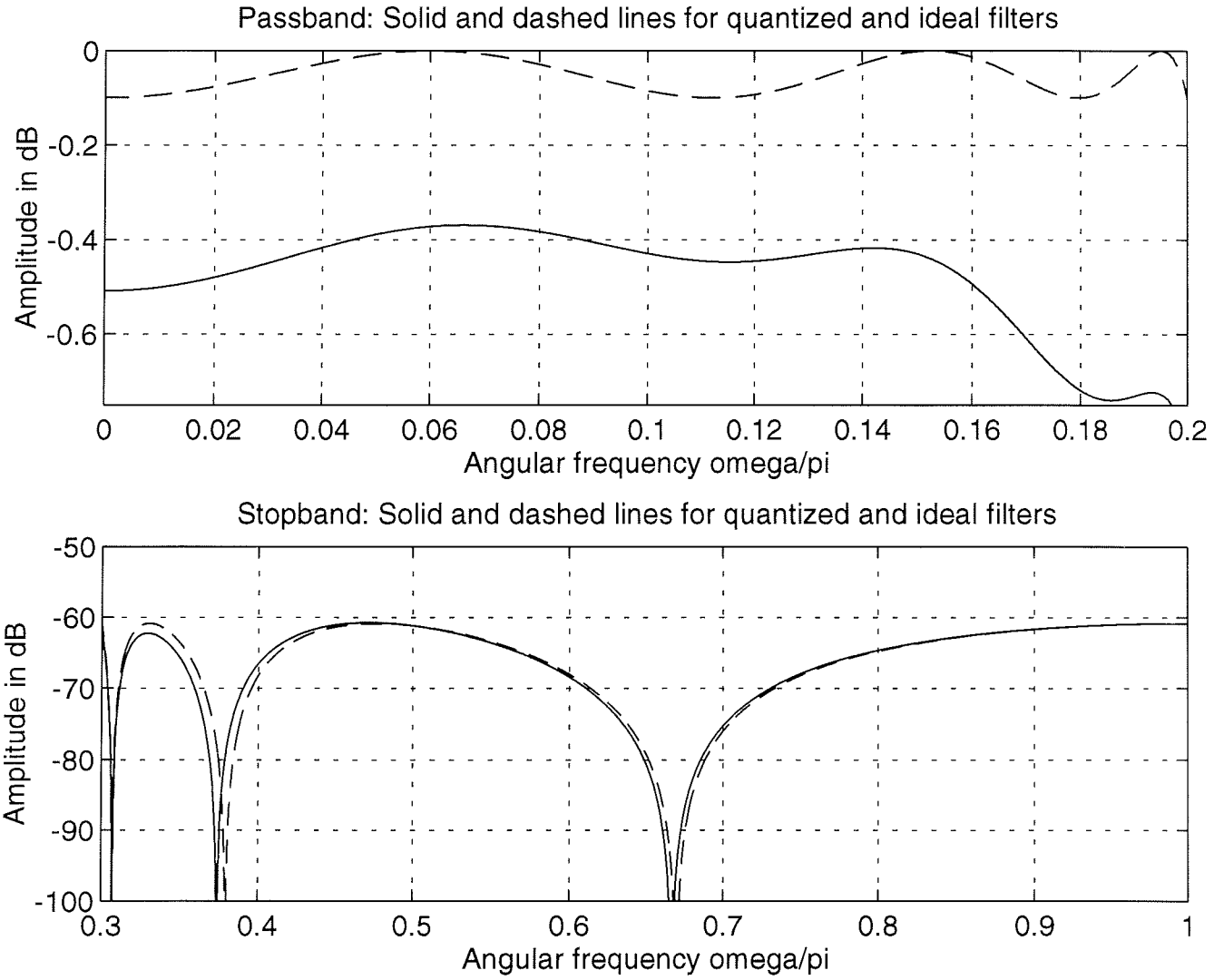

Pole-zero plot for the ideal filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1


Pole-zero plot for the quantized filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

$A_p = 0.1$, $A_s = 60.85$: The passband maximum is not scaled to be 0 dB

- When the passband maximum is not scaled to be 0 dB, the resulting responses for $b = 7$ fractional bits are as shown in the following two transparencies.
- In this case, maximum and minimum of the passband response are approximately -0.35 dB and -0.85 dB.
- In practice, this is tolerable.
- For this design, $S = 14 \cdot 2^{-7}$, $b_{11} = 183 \cdot 2^{-7}$, $b_{21} = -69 \cdot 2^{-7}$, $b_{12} = 186 \cdot 2^{-7}$, $b_{22} = -91 \cdot 2^{-7}$, $b_{13} = 193 \cdot 2^{-7}$, $b_{23} = -116 \cdot 2^{-7}$, $a_{01} = a_{21} = 11 \cdot 2^{-7}$, $a_{11} = 11 \cdot 2^{-7}$, $a_{02} = a_{22} = 22 \cdot 2^{-7}$, $a_{12} = -17 \cdot 2^{-7}$, $a_{03} = a_{23} = 266 \cdot 2^{-7}$, and $a_{13} = -303 \cdot 2^{-7}$.


Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Passband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-0.2
-0.4
-0.6
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
Angular frequency omega/pi
Stopband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
-50
-60
-70
-80
-90
-100
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

COMMENTS

- The above example shows clearly that in the case of cascaded-form implementation, the stopband response remains practically the same when the filter coefficients are quantized.
- The passband response, in turn, is much more sensitive to the quantization effects.
- Therefore, it is advisable to design the starting point elliptic filter in such a manner that its stopband attenuation is rather close to the desired one thereby allowing more quantization error in the passband region.
- In the above, we concentrated only on direct rounding of the filter coefficients. Better results can be obtained by using a more sophisticated optimization algorithms.
- If you are interested in these algorithms, please contact the lecturer.

EFFECTS OF COEFFICIENT ROUNDING FOR IIR FILTERS IMPLEMENTED AS A PARALLEL CONNECTION OF TWO ALLPASS FILTERS

- These filters have been considered in details in the course DIGITAL FILTERSING II, lecture notes DESIGN OF RECURSIVE FILTERS USING ALLPASS FILTERS AS BUILDING BLOCKS.
- In the end of this pile you can find a Matlab-file allcoe.m working together with allphaa.m.
- First, the user specifies an odd filter order N, the passband and stopband edges ω_p and ω_s, both as fractions of π, and the passband ripple A_p in dB.
- After that the routine finds out the minimum stopband attenuation A_s in dB such that the stopband edge of the corresponding elliptic filter is the desired one.
- The order N must be odd in order for the overall filter to be implementable in the form $H(z) = (1/2)[A(z) + B(z)]$ where $A(z)$ and $B(z)$ are allpass

filters of orders N_A and N_B such that $N_A + N_B = N$ and $N_A = N_B - 1$ or $N_A = N_B + 1$. The program automatically forms the desired $A(z)$ and $B(z)$ which are implementable as a cascade of first-order and second-order sections.

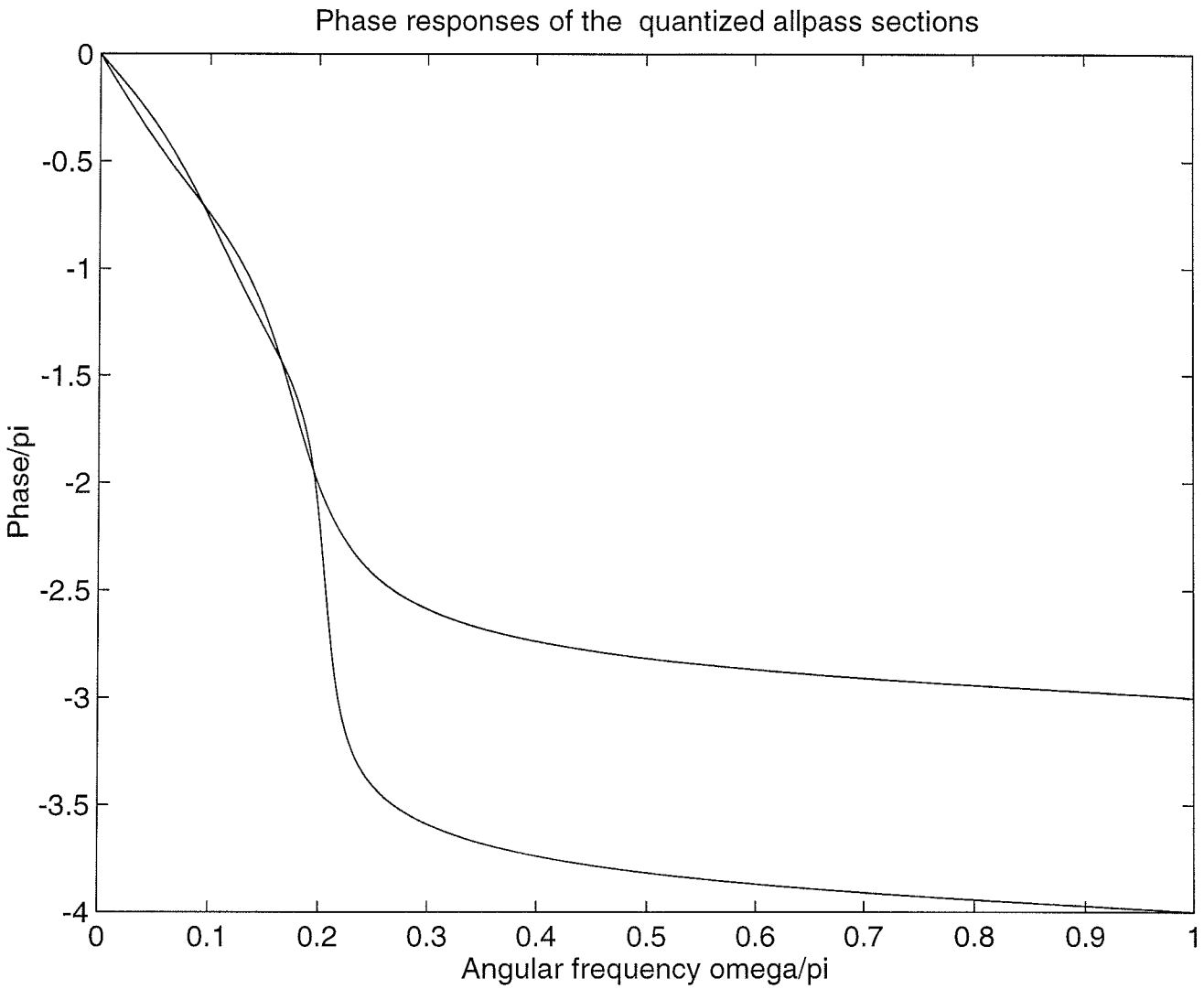
- Note that only the pole locations are needed in forming $A(z)$ and $B(z)$.
- These sections can be implemented either using direct-form implementations (see pages 21 and 22 in the above-mentioned lecture notes) or using wave digital filter structures (see pages 23 and 24). The use specifies which implementation is desired to be used.
- Finally, b, the number of fractional bits for the coefficient representation are given and the resulting amplitude response is compared with that of the infinite-precision filter. Also the corresponding pole-zero plots are given.

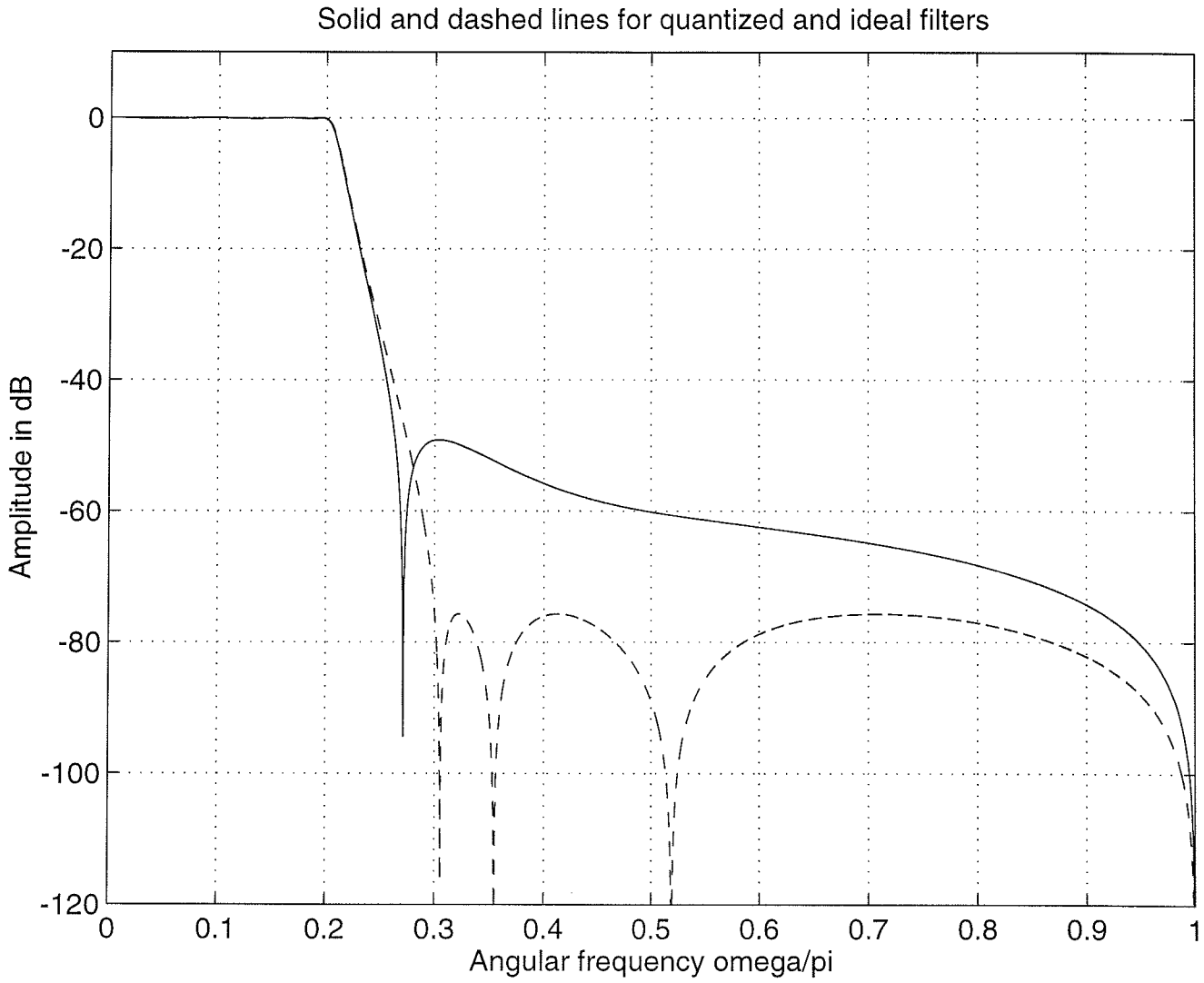
Example 1

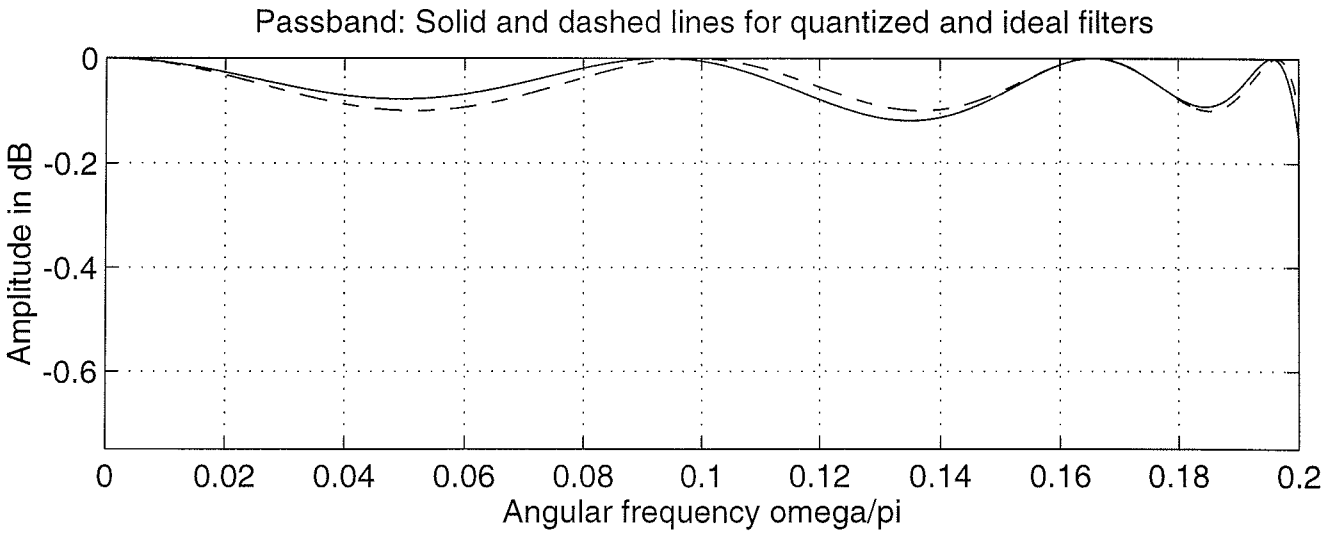
- $N = 7$, $\omega_p = 0.2\pi$, $\omega_s = 0.3\pi$, $A_p = 0.1$. The corresponding $A_s = 75.72$.
- The following five transparencies consider the coefficient quantization of the direct-form allpass sections to $b = 7$ fractional bits, whereas the last five transparencies consider the corresponding wave digital filter transfer functions.
- It is seen that in both cases the passband response does not change very much after coefficient quantization, whereas in the stopband the effect is dramatic.
- The explanation to this is that in order to keep the passband ripple less than or equal to 0.1 dB, the difference in the phases of the two allpass sections must be in the passband less than or equal to $\pm 0.136\pi$. In order to keep the minimum stopband attenuation less than or equal to 75.72 dB, the difference in the phases of the two allpass sections must be in the stopband within the limits

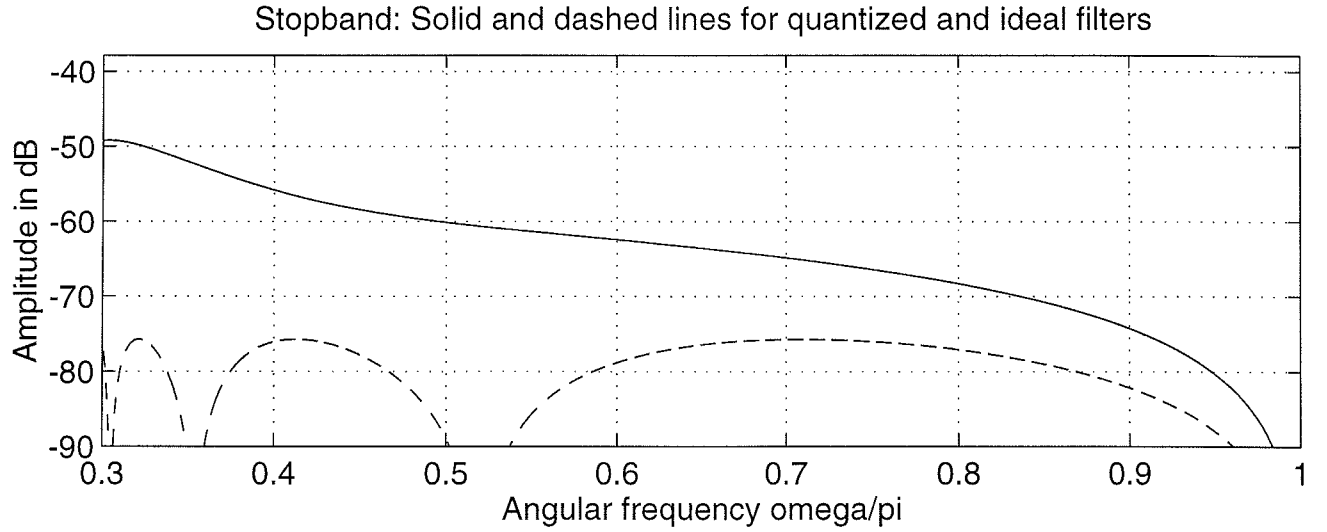
$(1 \pm 1.7 \cdot 10^{-8})\pi$.

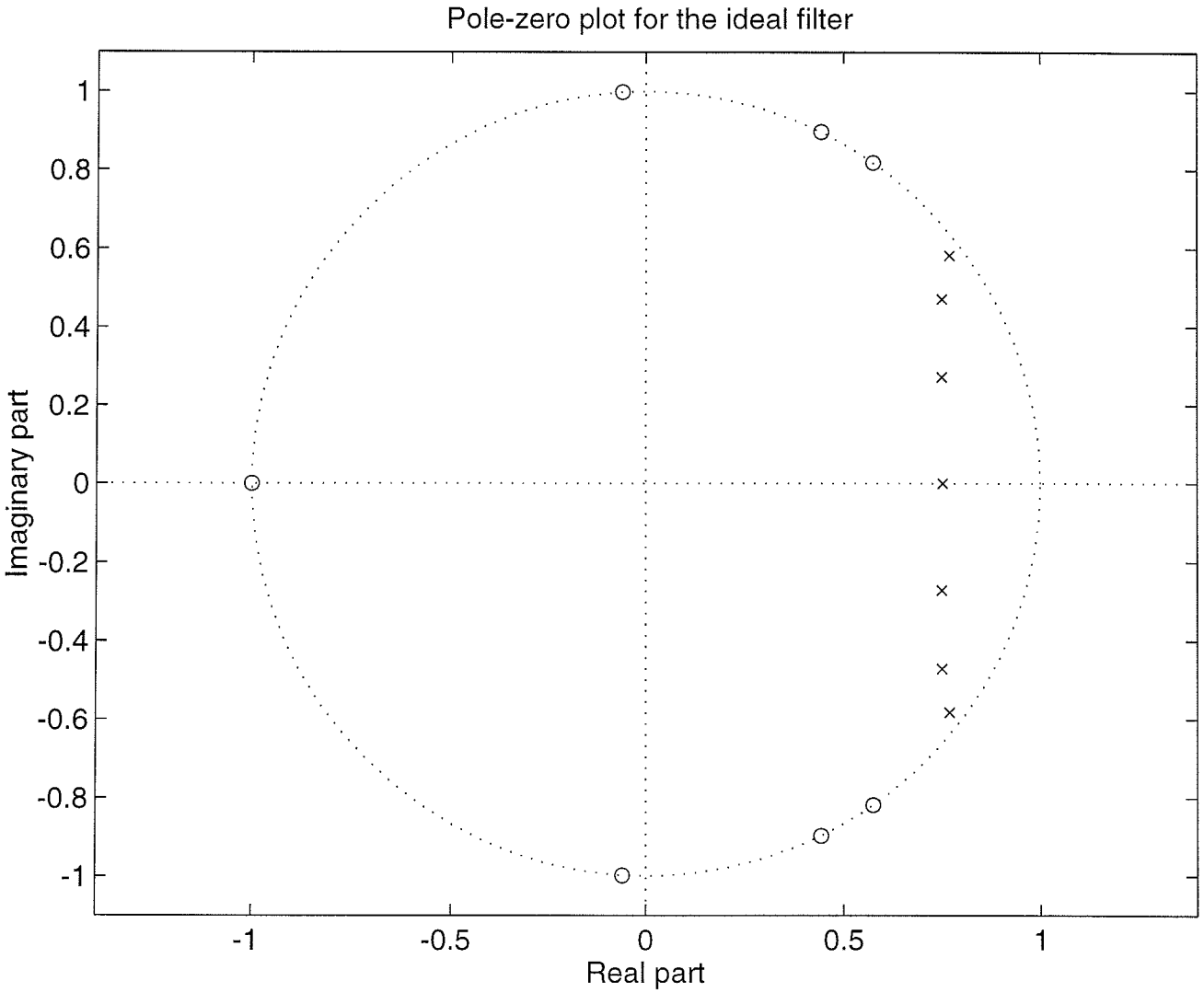
- This is why the coefficient quantization has more effect in the stopband region, where smaller changes in the phase responses are required.
- The above performance is almost opposite to that of the filters implemented in the cascade form.
- It is also interesting to observe in the following transparencies that after coefficient quantization the filters still have in the passband four points where the filter achieves exactly 0 dB (at these points the phases of the two allpass sections have the same values).

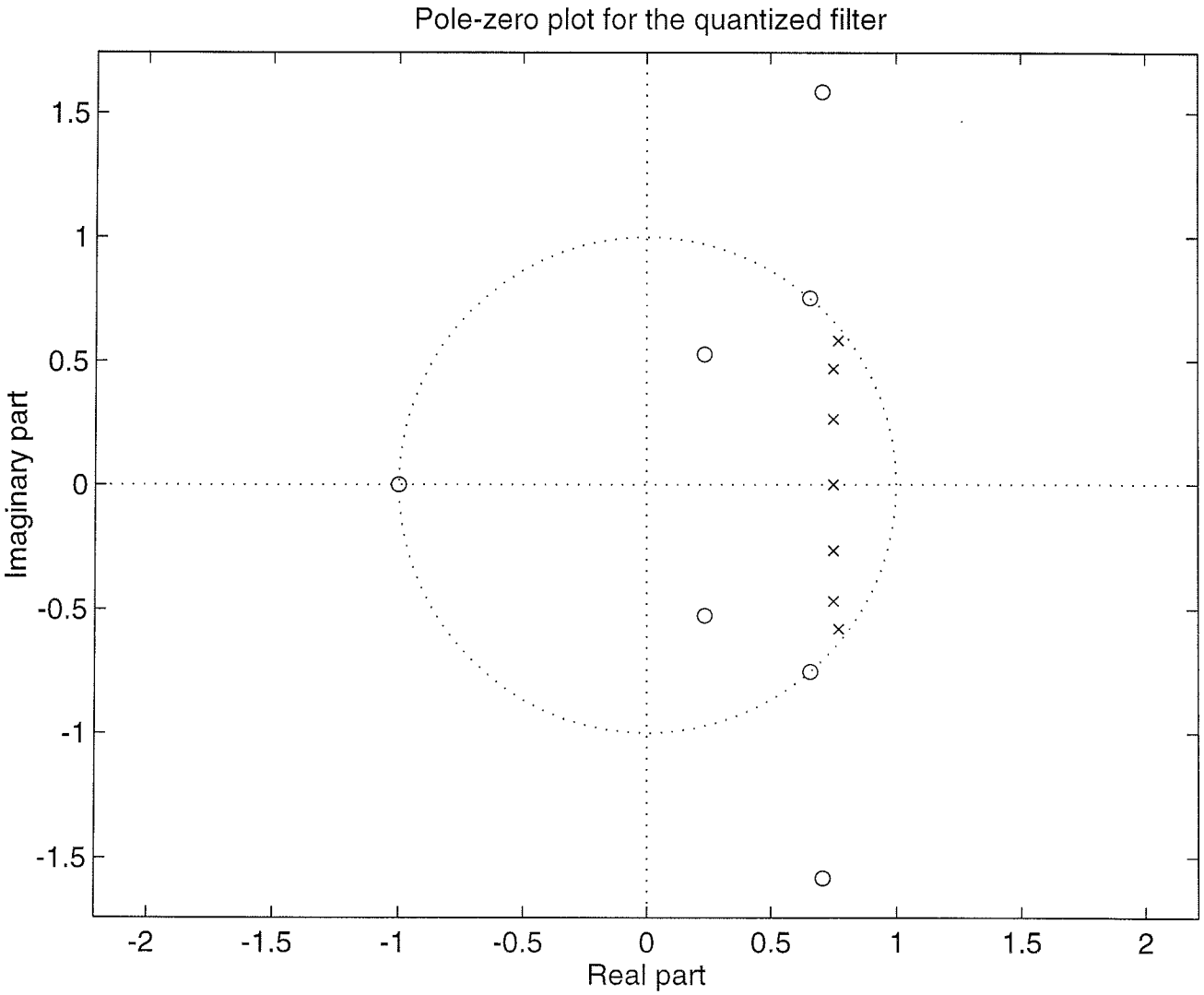

Phase responses of the quantized allpass sections
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
Phase/pi
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

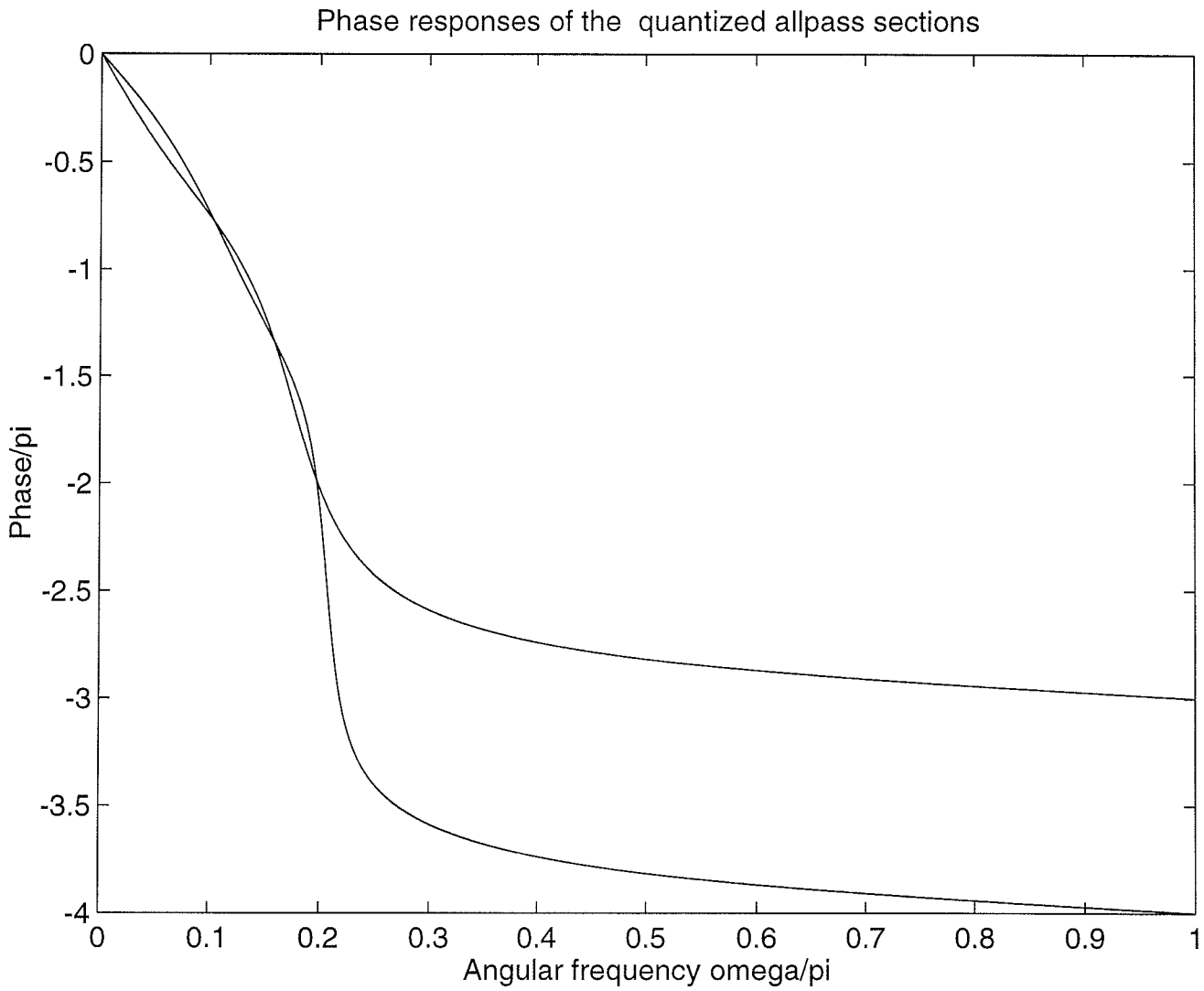

Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-20
-40
-60
-80
-100
-120
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

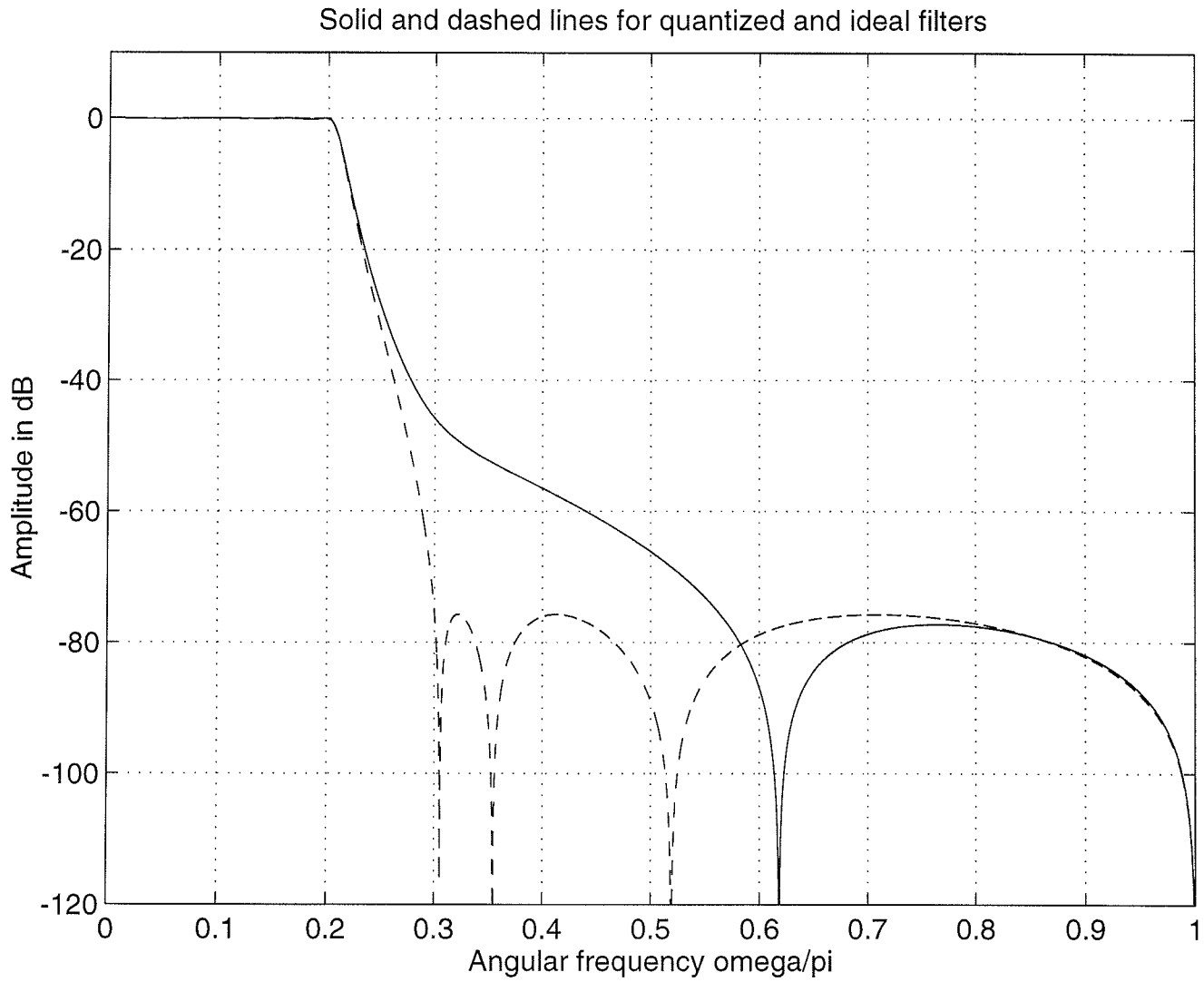

Passband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-0.2
-0.4
-0.6
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
Angular frequency omega/pi

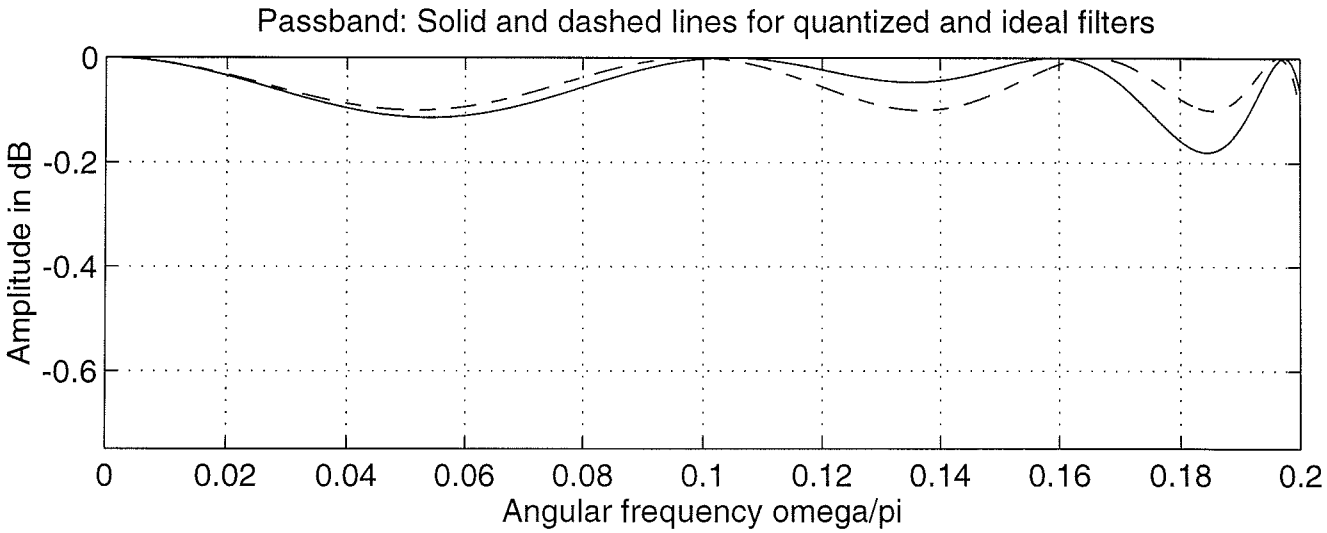

Stopband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
-40
-50
-60
-70
-80
-90
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

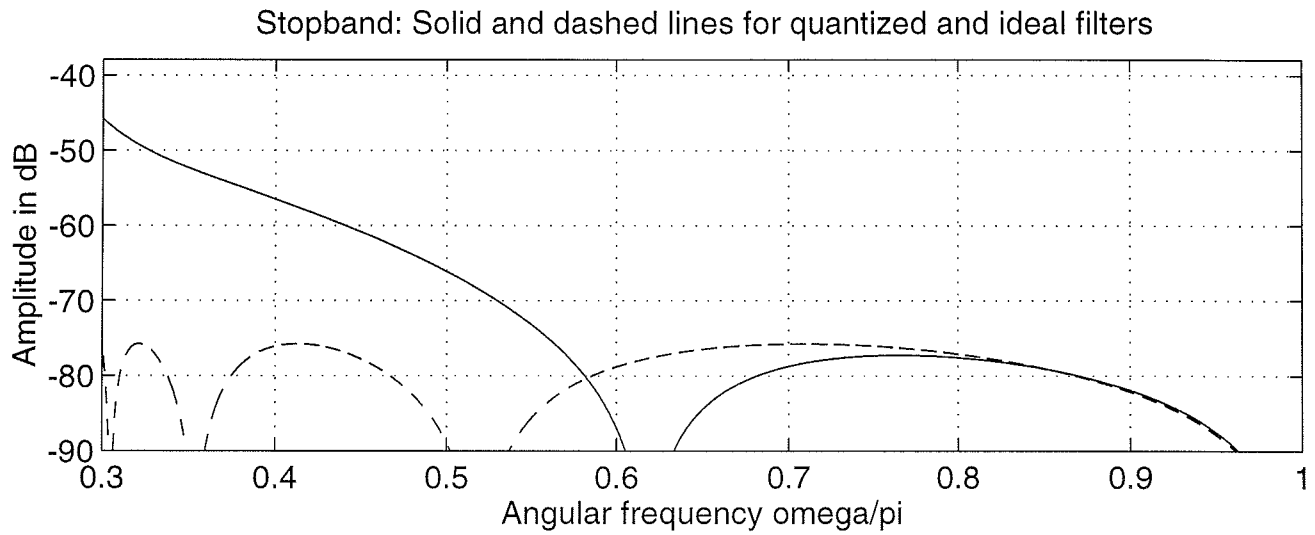

Pole-zero plot for the ideal filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

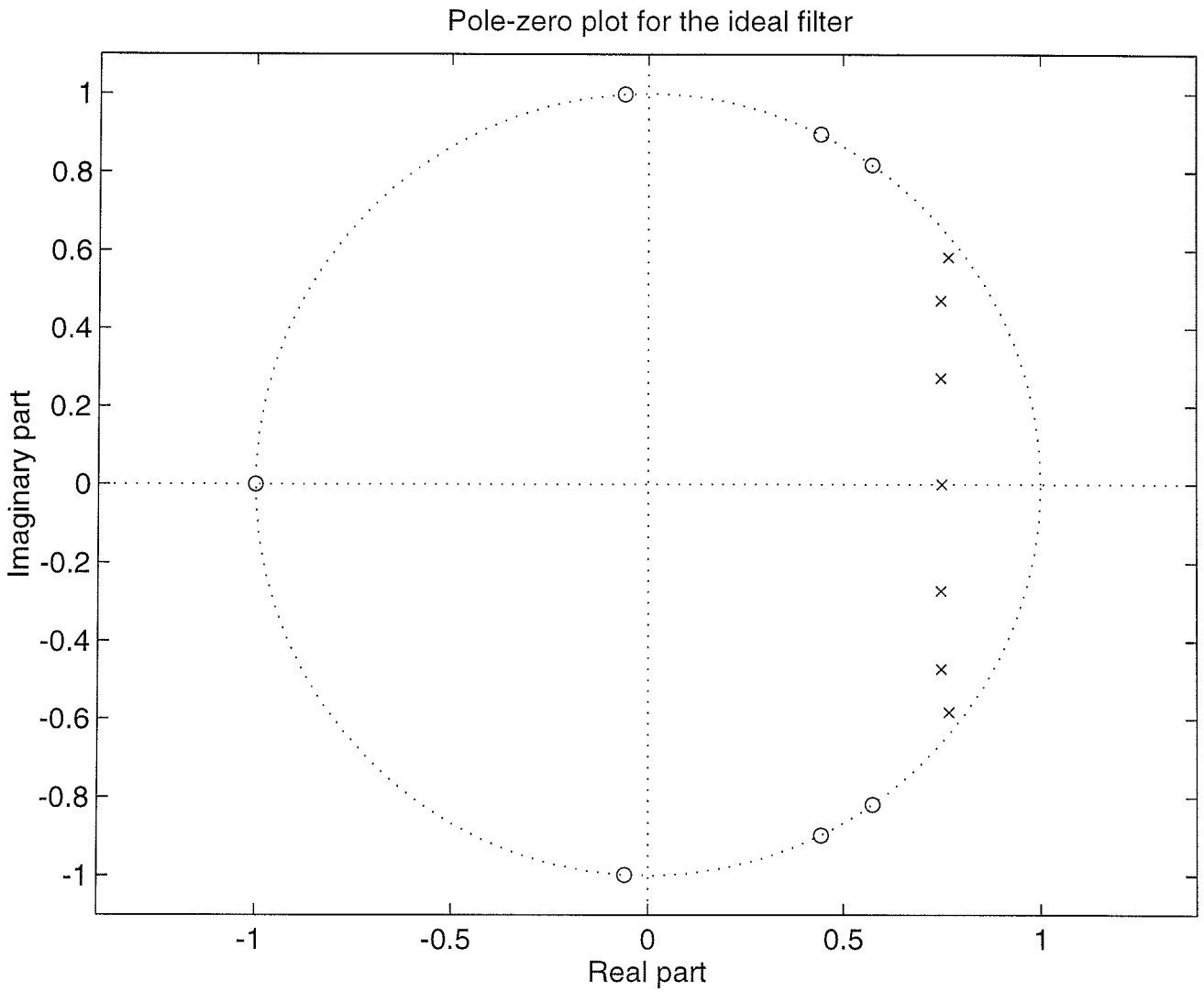

Pole-zero plot for the quantized filter
Imaginary part
Real part
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

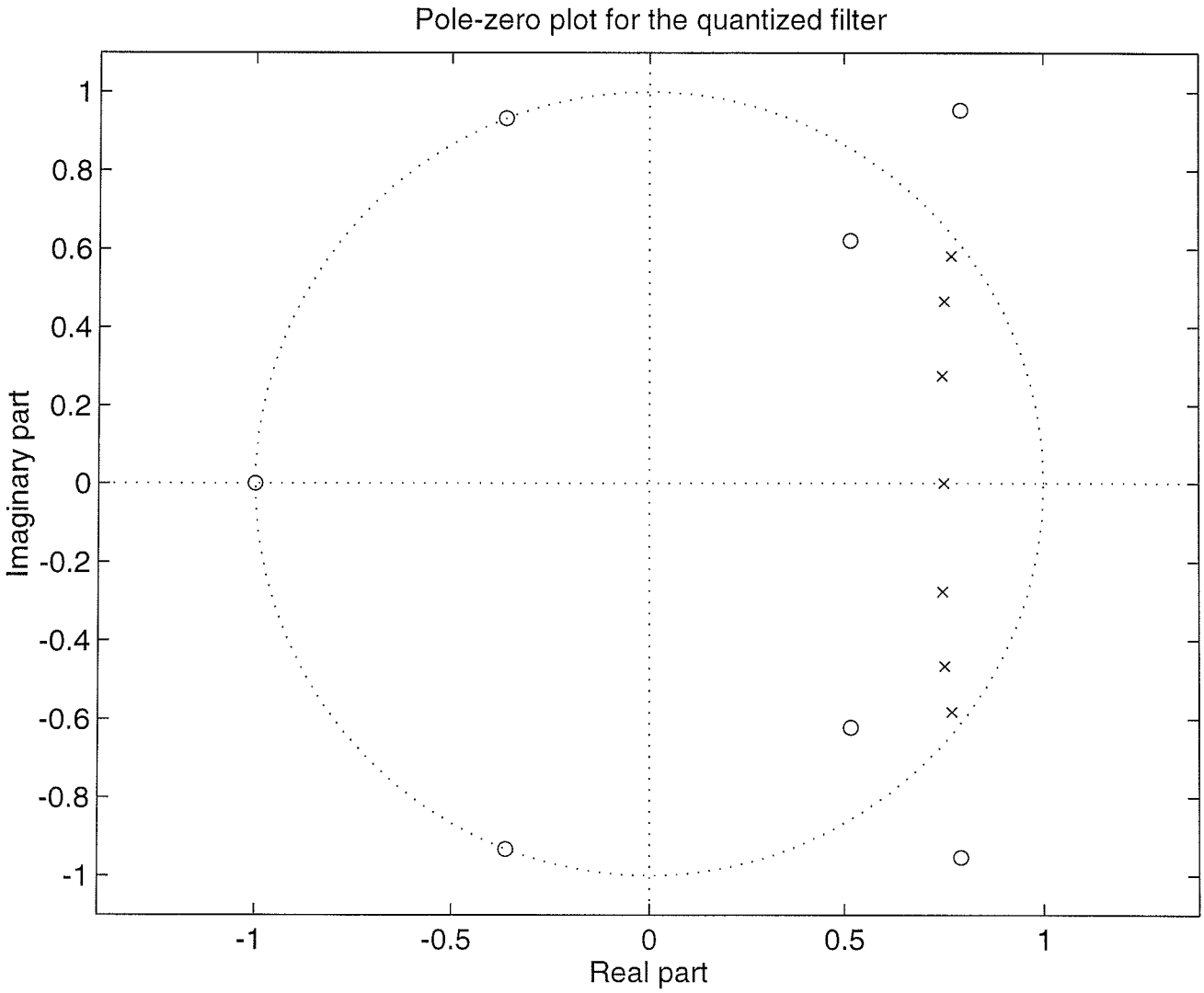

Phase responses of the quantized allpass sections
Phase/pi
Angular frequency omega/pi
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-20
-40
-60
-80
-100
-120
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Passband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-0.2
-0.4
-0.6
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
Angular frequency omega/pi


Stopband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
-40
-50
-60
-70
-80
-90
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Pole-zero plot for the ideal filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1


Pole-zero plot for the quantized filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

Example 2

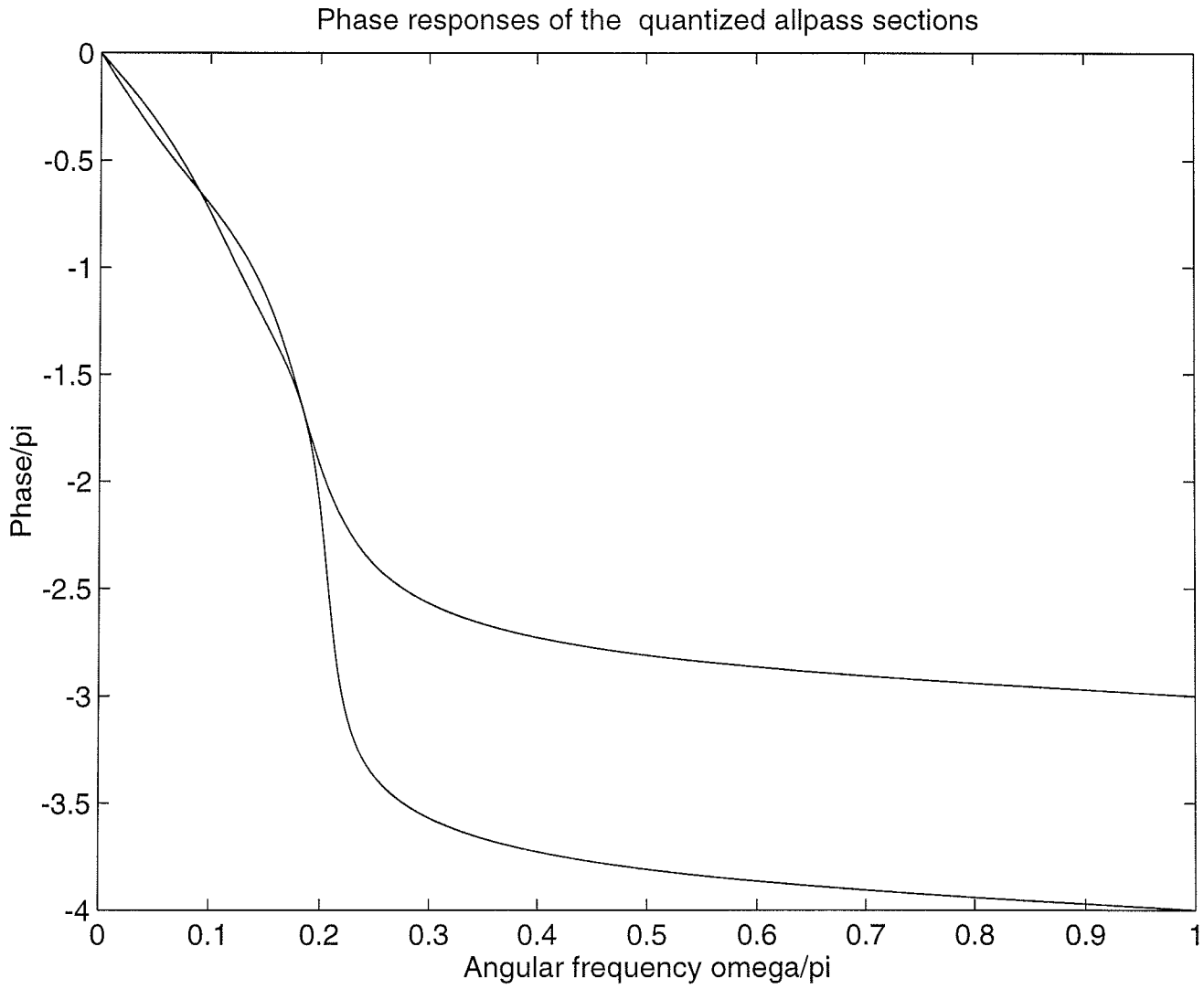
- We consider the same problem as for the cascaded-form IIR filters:
- What is the minimum number of fractional bits required for a seventh-order IIR filter to meet the criteria?:
  - In the passband $[0, 0.2\pi]$ the amplitude response stays between 0 dB and 0.5 dB.
  - In the stopband $[0.3\pi, \pi]$ the minimum attenuation is at least 60 dB.
- Since the overall order must be odd, we design a seventh-order filter, instead of a sixth-order filter.
- For cascaded-form IIR filters, we observed that a good starting point infinite-precision elliptic filter is the one whose stopband ripple is close to the desired ripple after coefficient quantization. This is because the effect of the coefficient quantization is very small in the stopband region.
- For filters being implementable as a parallel connec-

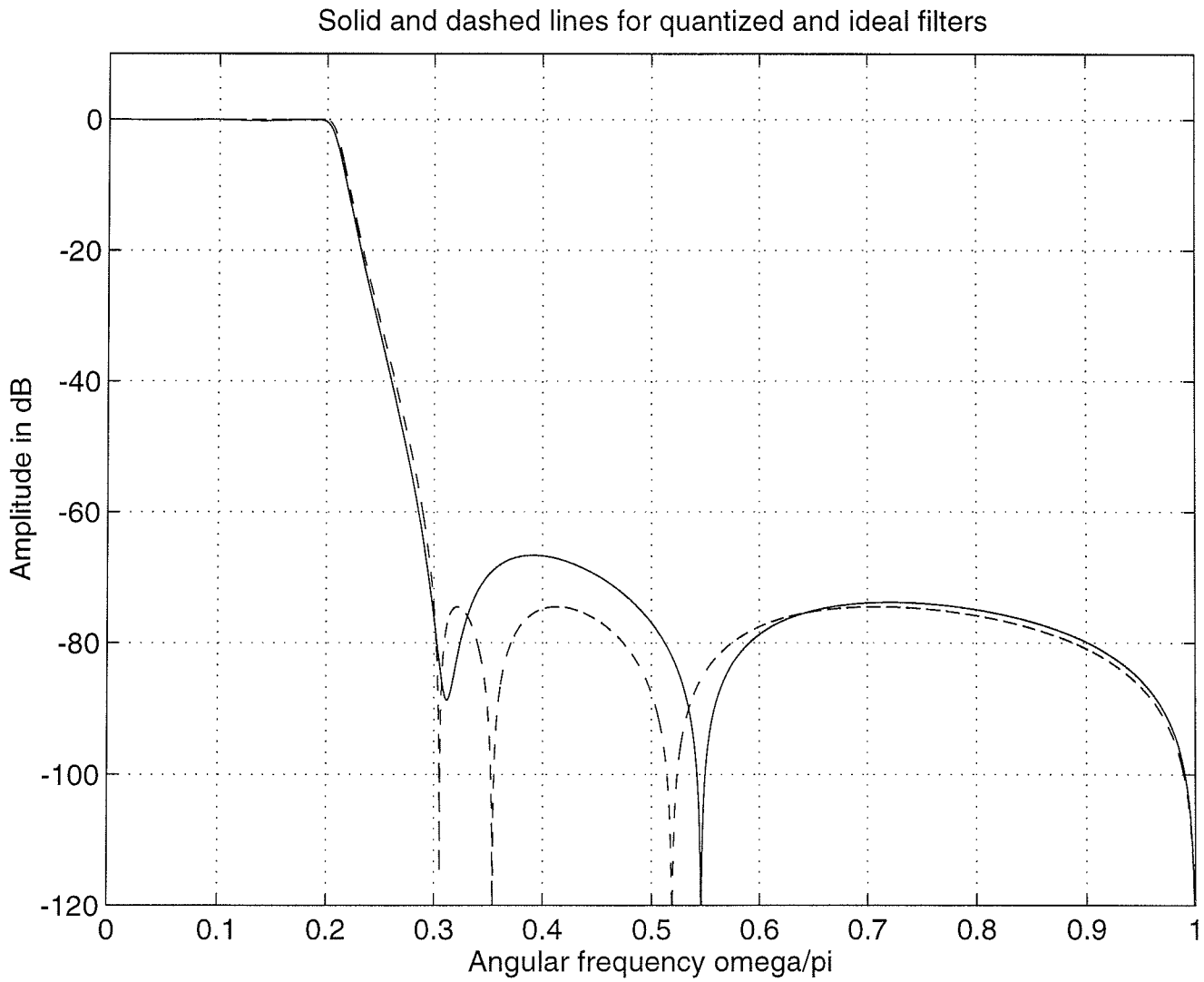
tion of two allpass filters, there are no real rules of thumb for selecting the starting point elliptic filter.

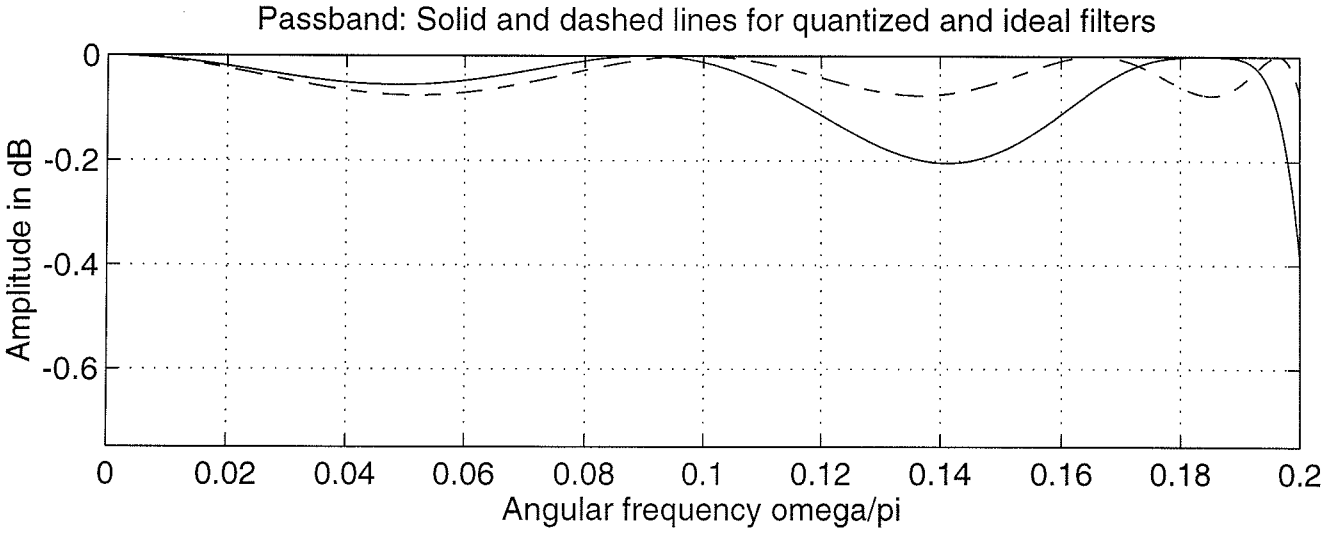
- Therefore, it is advisable to test several admissible values of A_p and to select among them the one giving the best solution.
- For this purpose there is a Matlab-file alltest.m. In this routine, we are able to use a for-loop for testing several values of A_p. For our problem, the minimum and maximum values of A_p are 0.0027 ($A_s = 60$) and 0.5 ($A_s = 82.19$)

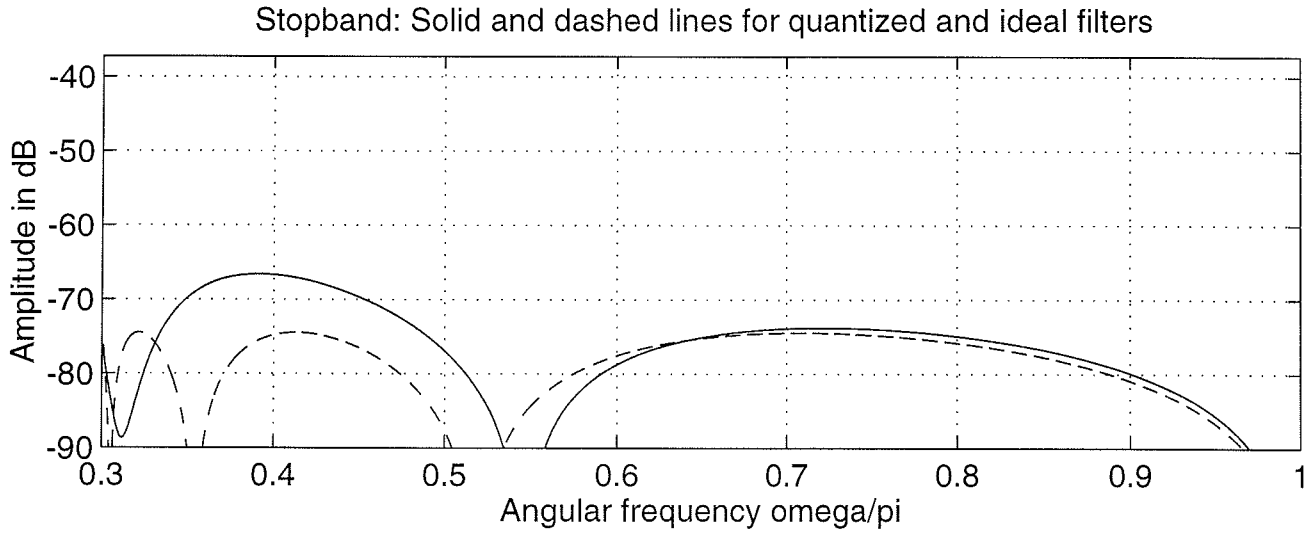
Results

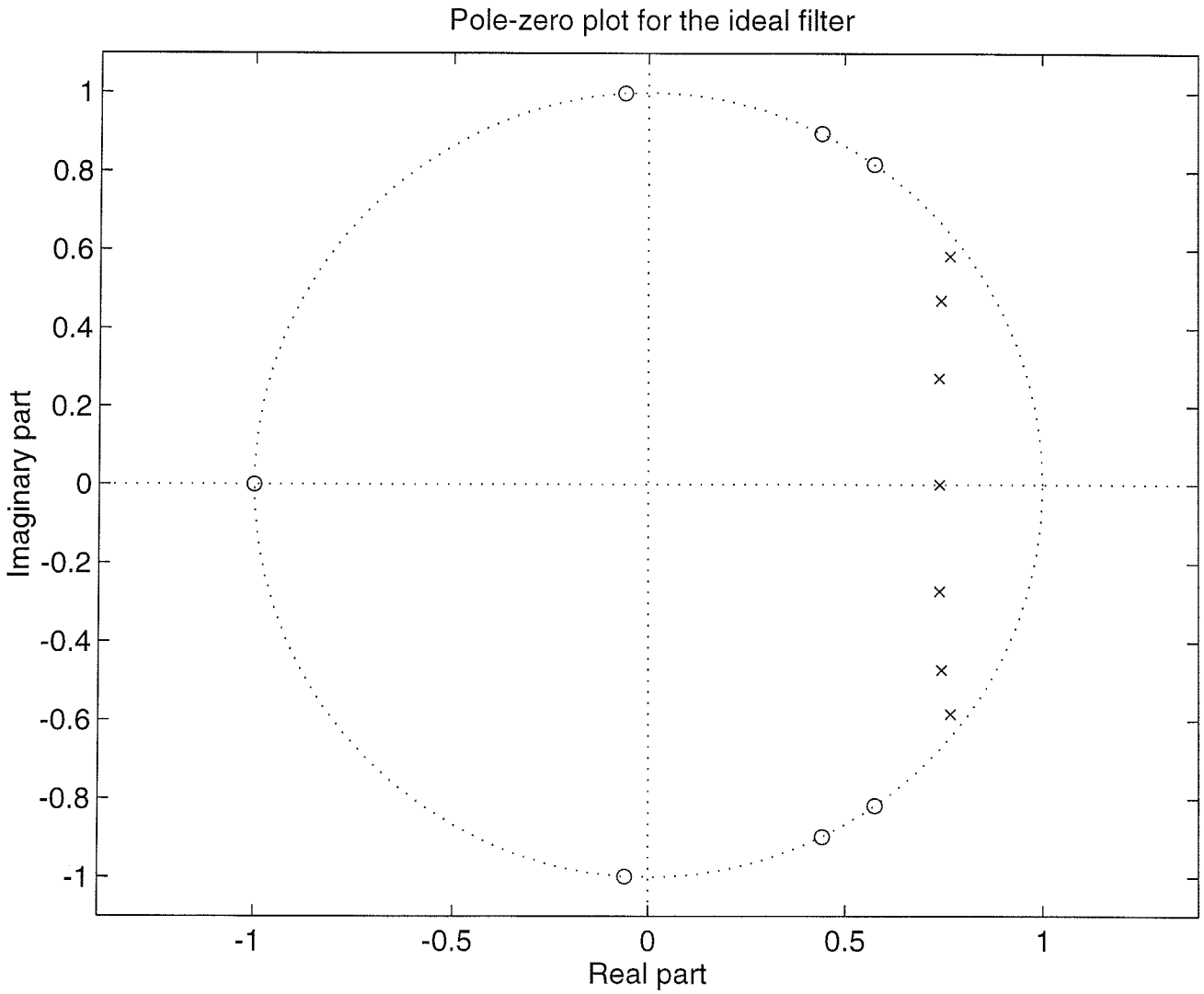
- For the direct allpass implementations, $b = 6$ fractional bits are needed. The best result is obtained by using $A_p = 0.0750$
- After coefficient quantization, the passband ripple and the minimum stopband attenuation are 0.3898 dB and 66.60 dB, respectively. The following five transparencies compare this design with the infinite-precision design.
- The first allpass section consists of a first-order allpass section with the denominator $1 - 47 \cdot 2^{-6} z^{-1}$ and a second-order allpass section with the denominator $1 - 95 \cdot 2^{-6} z^{-1} + 50 \cdot 2^{-6} z^{-2}$.
- The second allpass section consists of two second-order allpass sections with the denominators $1 - 95 \cdot 2^{-6} z^{-1} + 40 \cdot 2^{-6} z^{-2}$ and $1 - 98 \cdot 2^{-6} z^{-1} + 59 \cdot 2^{-6} z^{-2}$.

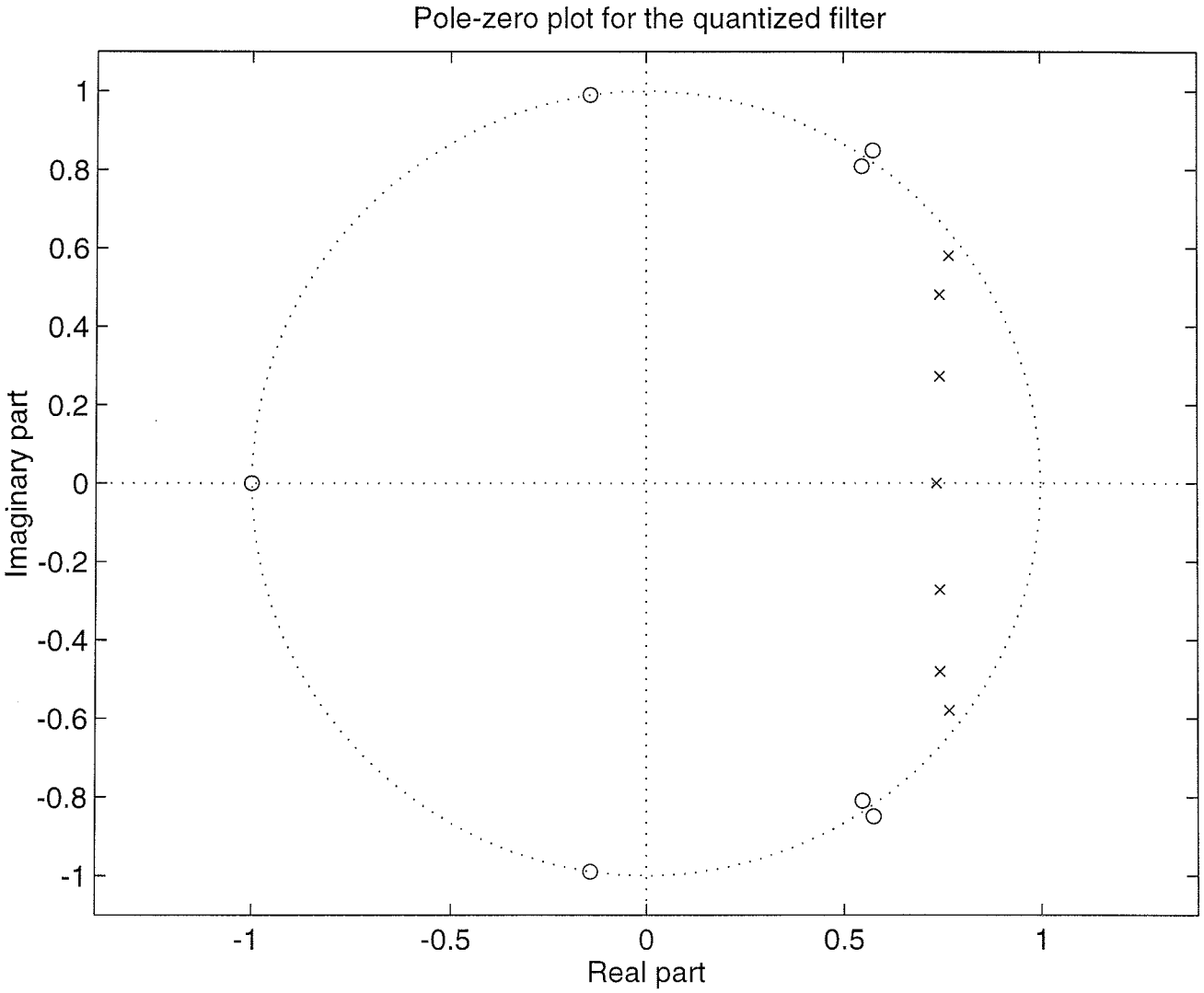

Phase responses of the quantized allpass sections
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
Phase/pi
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-20
-40
-60
-80
-100
-120
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Passband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-0.2
-0.4
-0.6
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
Angular frequency omega/pi

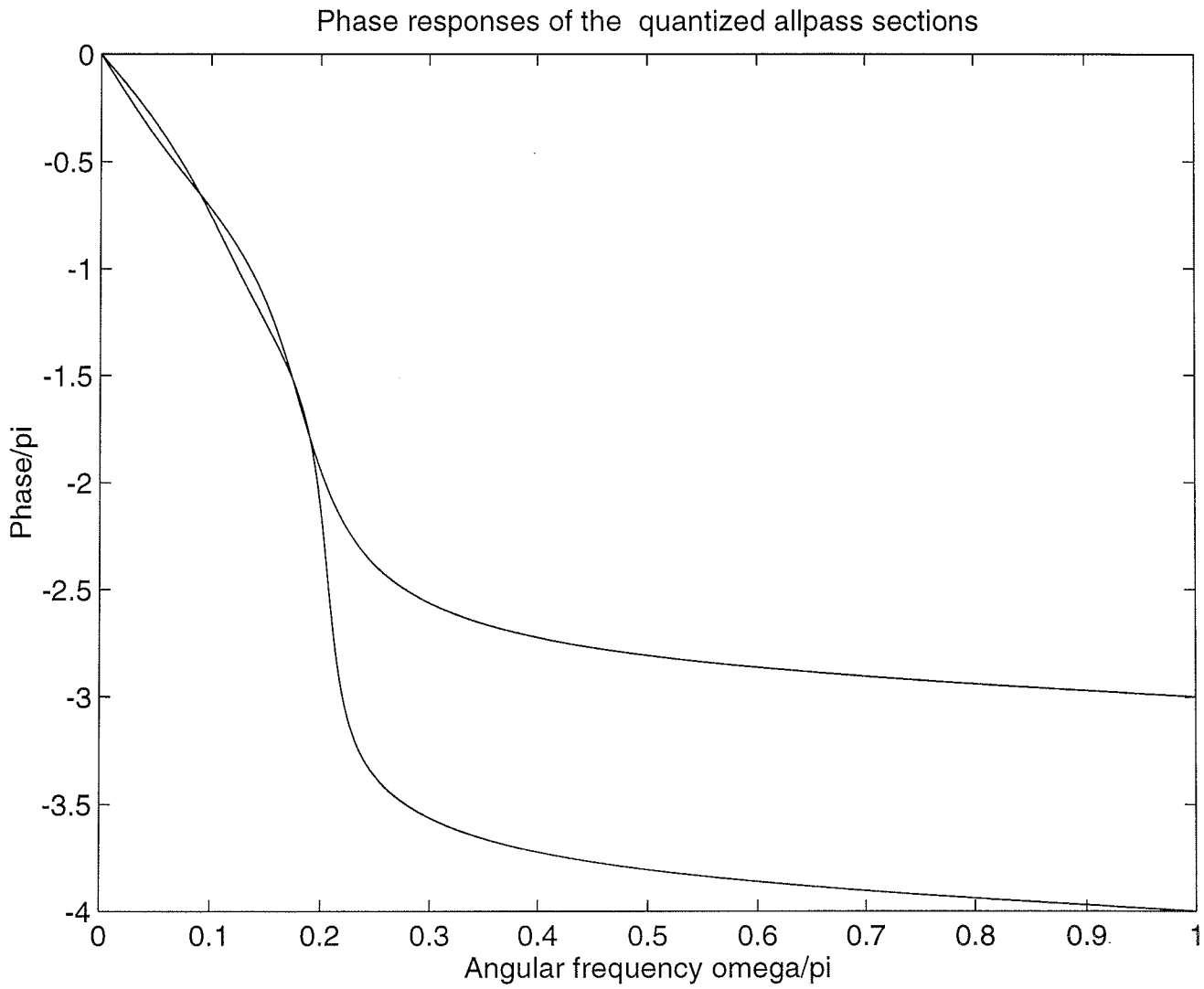

Stopband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
-40
-50
-60
-70
-80
-90
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

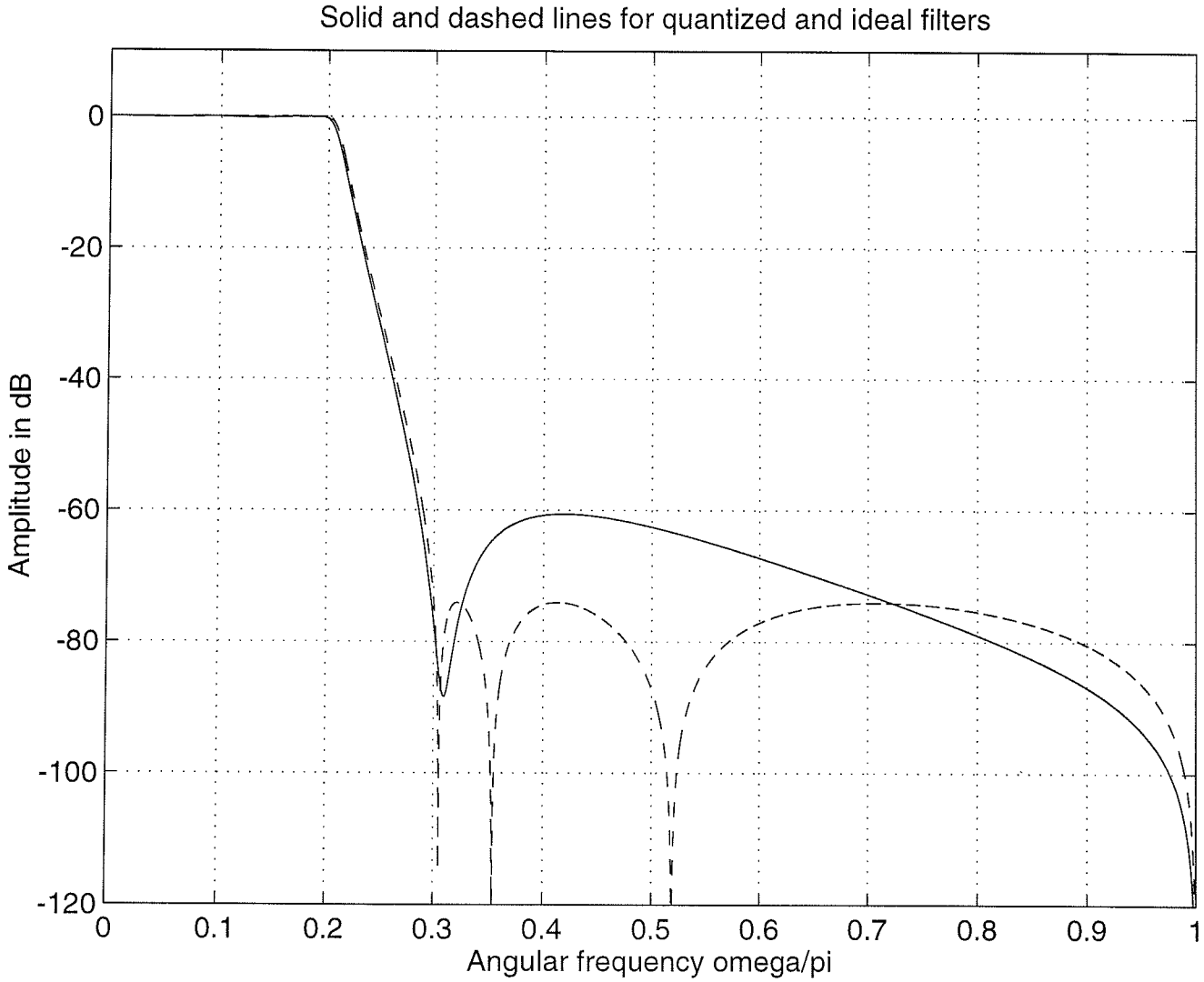

Pole-zero plot for the ideal filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

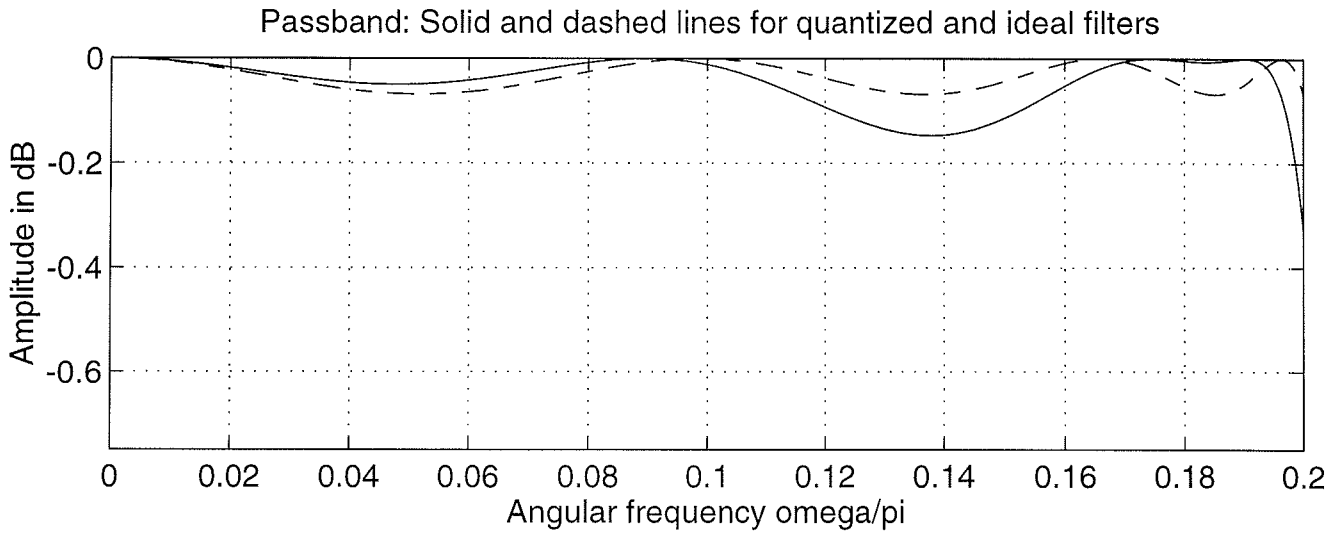

Pole-zero plot for the quantized filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

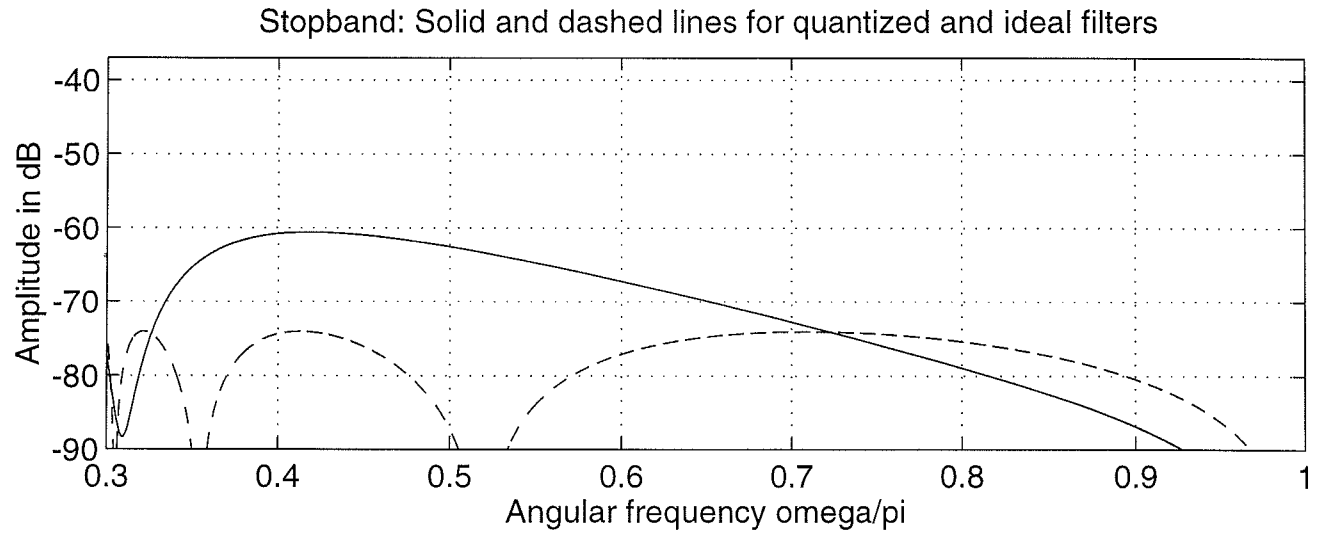
Results

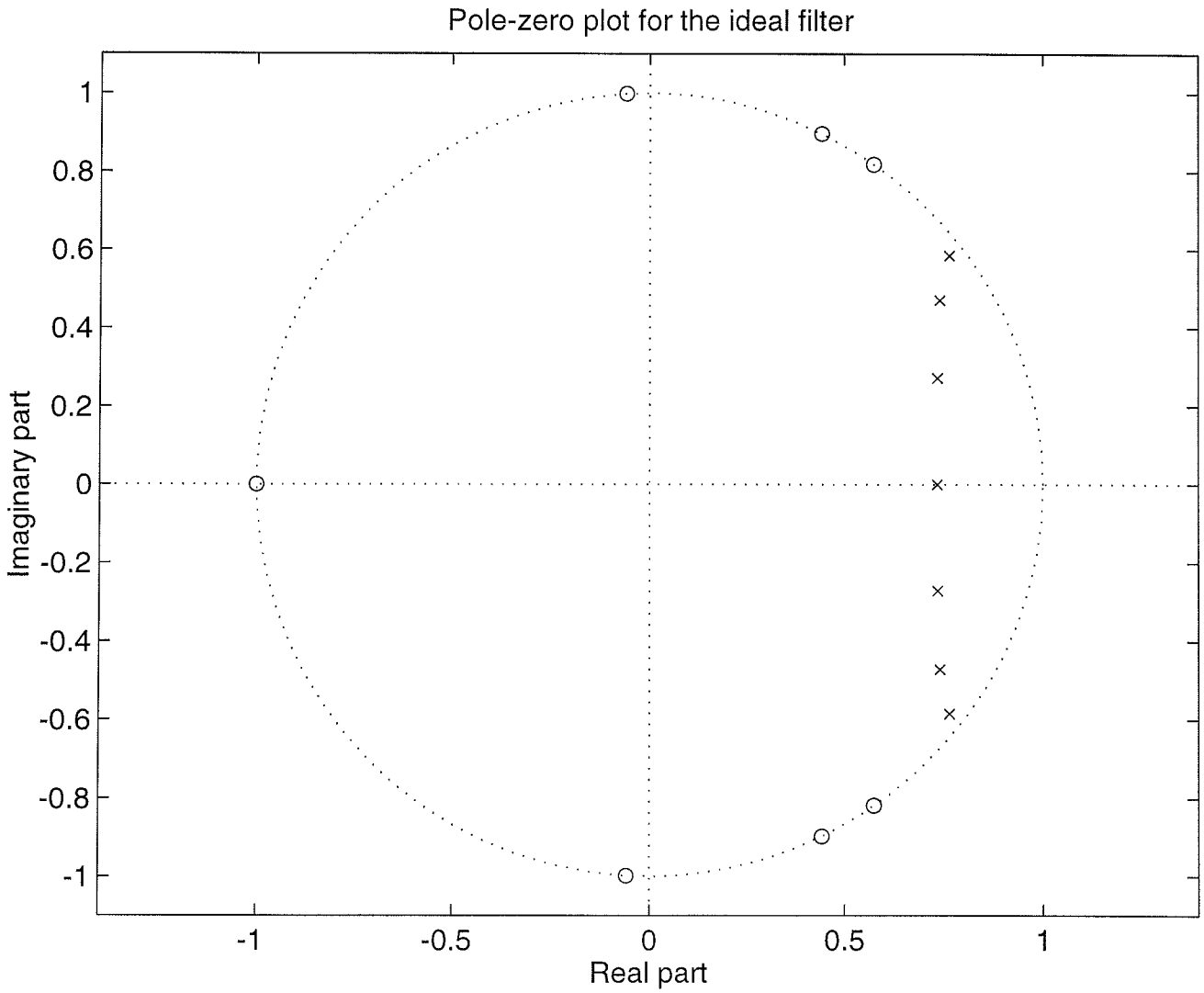
- For the wave digital allpass implementations, $b = 7$ fractional bits are needed. The best result is obtained by using $A_p = 0.0680$
- After coefficient quantization, the passband ripple and the minimum stopband attenuation are 0.3272 dB and 60.58 dB, respectively. The following five transparencies compare this design with the infinite-precision design.
- The first allpass section consists of a first-order allpass section with $\gamma = 94 \cdot 2^{-7}$ and a second-order allpass section with $\gamma_1 = -99 \cdot 2^{-7}$ and $\gamma_2 = 107 \cdot 2^{-7}$.
- The second allpass section consists of two second-order allpass sections with $\gamma_1 = -79 \cdot 2^{-7}$ and $\gamma_2 = 117 \cdot 2^{-7}$ and $\gamma_1 = -118 \cdot 2^{-7}$ and $\gamma_2 = 102 \cdot 2^{-7}$.

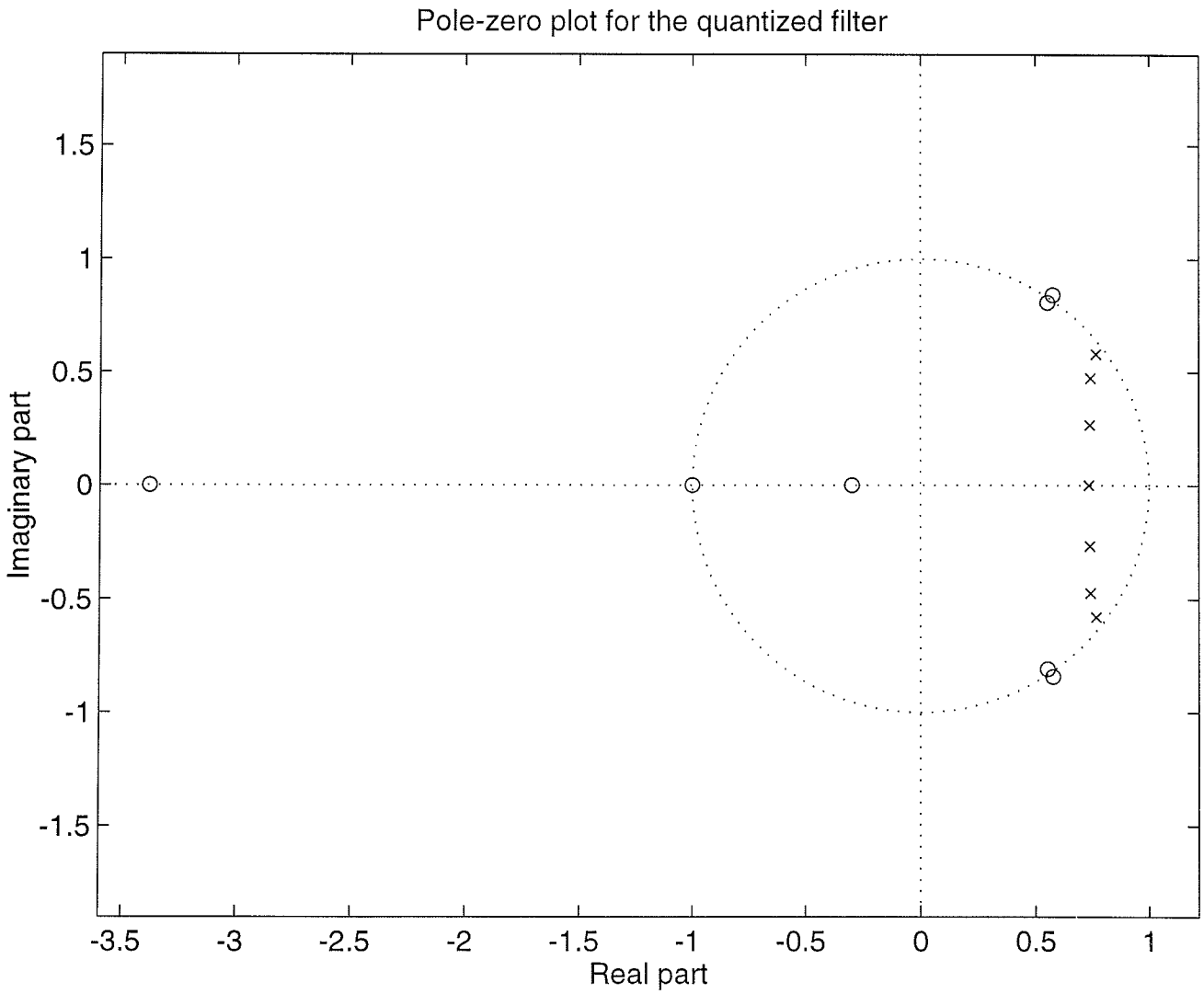

Phase responses of the quantized allpass sections
Phase/pi
Angular frequency omega/pi
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-20
-40
-60
-80
-100
-120
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Passband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
0
-0.2
-0.4
-0.6
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
Angular frequency omega/pi


Stopband: Solid and dashed lines for quantized and ideal filters
Amplitude in dB
-40
-50
-60
-70
-80
-90
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi


Pole-zero plot for the ideal filter
Imaginary part
Real part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1


Pole-zero plot for the quantized filter
Imaginary part
Real part
1.5
1
0.5
0
-0.5
-1
-1.5
-3.5
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1

VALIDITY OF THE NOISE MODEL

- Here, we simulate whether the results given on pages 45–608of the lectures notes FINITE WORD-LENGTH EFFECTS IN DIGITAL FILTERS in the course DIGITAL FILTERING II are also valid in practice. For the filters considered there we obtained the following output noise variances in terms of $\sigma_e^2 = 2^{-2b}/12$:

- The output noise variances are

$$\sigma_f^2 = \begin{cases} 18390\sigma_e^2 & \text{for unscaled} \\ 153.16\sigma_e^2 & \text{for worst-case} \\ 70.604\sigma_e^2 & \text{for } L_\infty \\ 19.071\sigma_e^2 & \text{for } L_2. \end{cases}$$

- In terms of the quantity

$$\text{NOISE GAIN} = 10\log_{10}(\sigma_f^2/\sigma_e^2),$$

the results are

$$\text{NOISE GAIN} = \begin{cases} 42.6 \text{ dB} & \text{for unscaled} \\ 21.9 \text{ dB} & \text{for worst-case} \\ 18.5 \text{ dB} & \text{for } L_\infty \\ 12.8 \text{ dB} & \text{for } L_2. \end{cases}$$

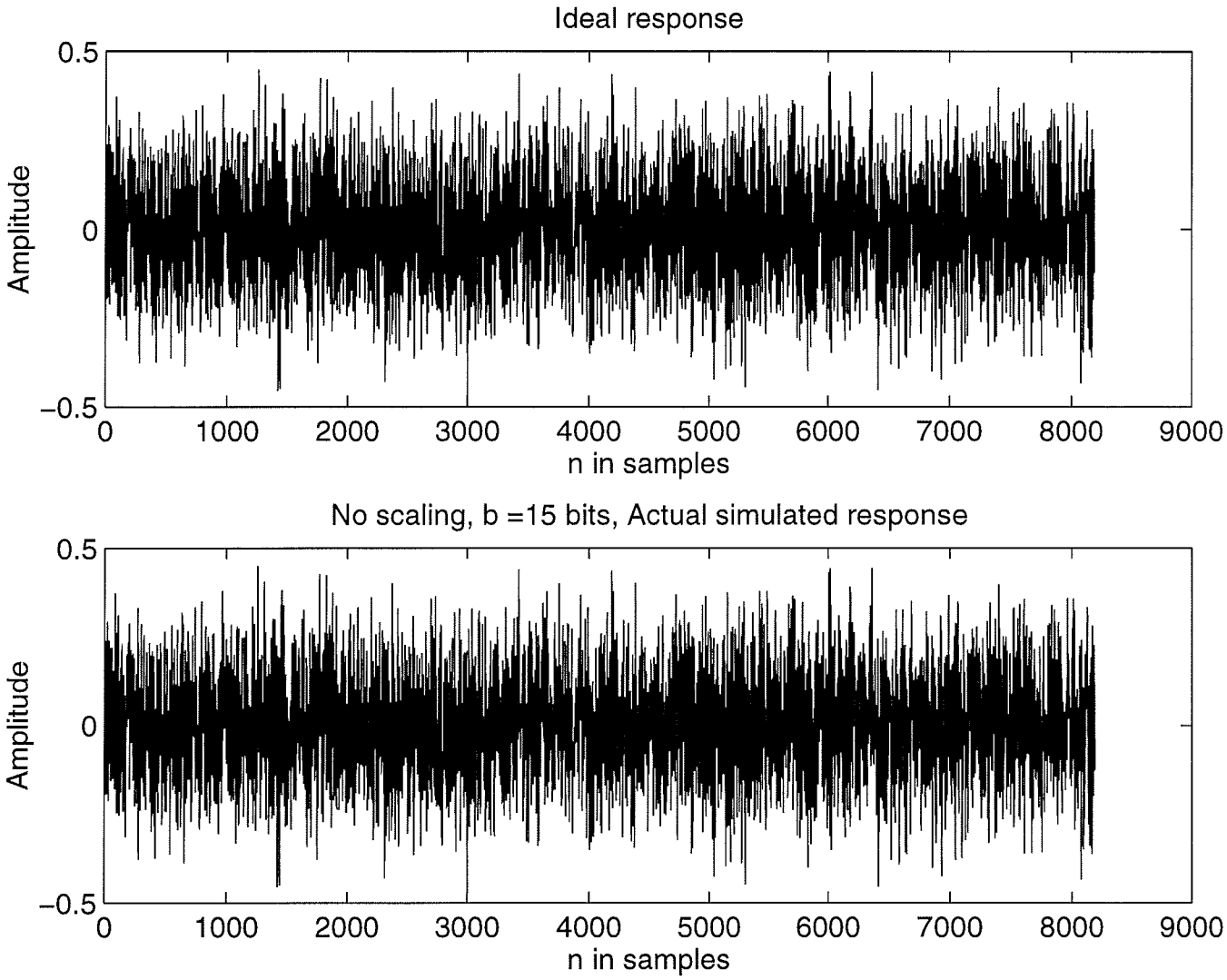
- For testing purposes, two Matlab-files, iirnois.m and dirgen.m, have been generated.
- Given ω_p, $N = 6$, A_p, and A_s, iirnois.m finds first an elliptic filter such that the given ripple values are just met. For the fiter under consideration $\omega_p = 0.4\pi$, $A_p = 0.5$, and $A_s = 80.24$. The resulting $\omega_s = 0.6\pi$.
- After that, the second-order blocks are formed in the manner discussed in the connection with the cascaded-form IIR filters previously.
- The next step is to scale the filter. The four alternatives are: no scaling, worst-case scaling, L_∞-norm scaling, and L_2-norm scaling.
- Then, dirgen.m is used to implement the second order sections in the manner which mimics two's complement arithmetic with the given b, the number of fractional bits for the data.
- Finally, the simulated response and the ideal response, obtained when all the multiplications are performed using the infinite-precision arithmetic, are

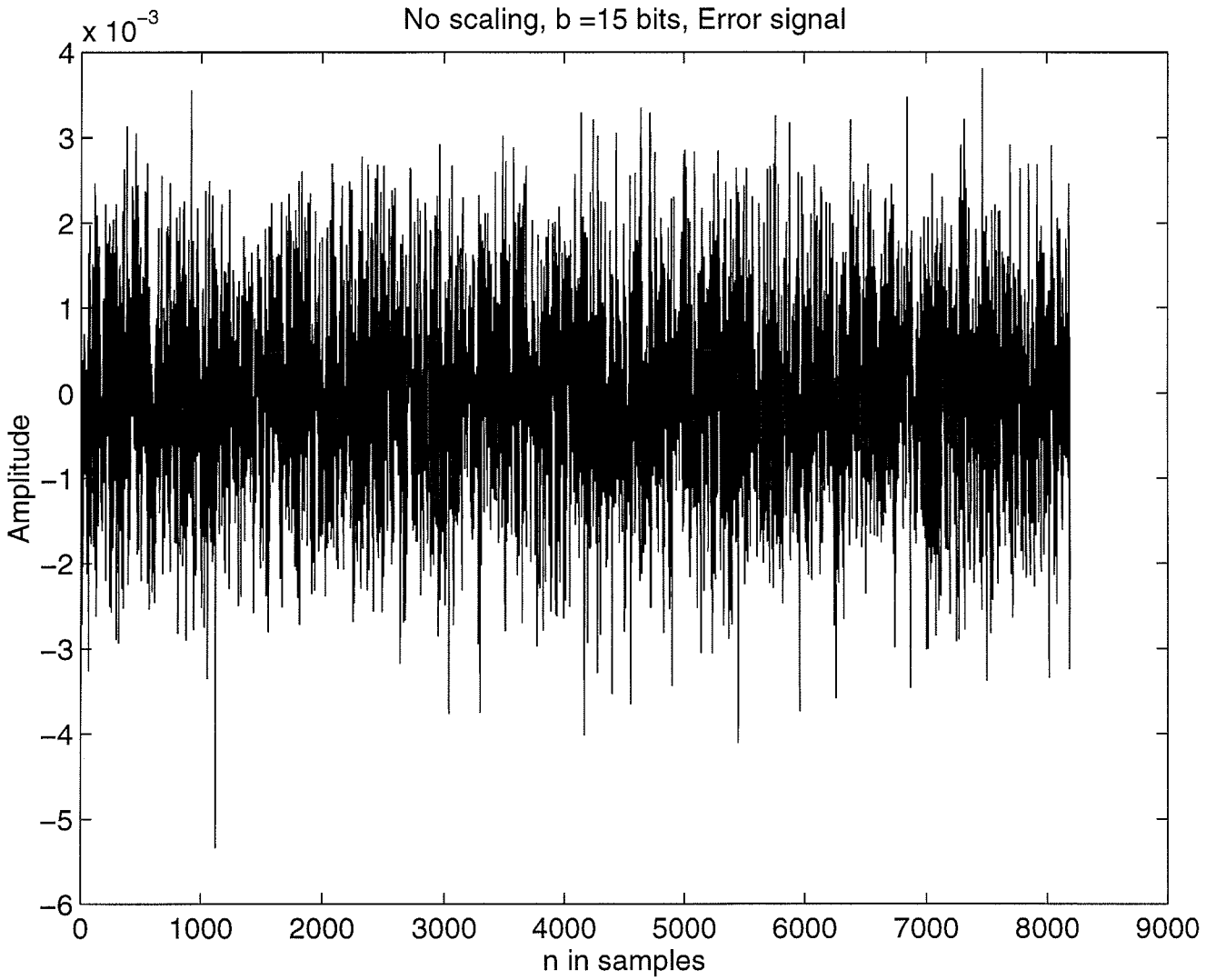
evaluated and compared with each other in order to determine the simulated error.

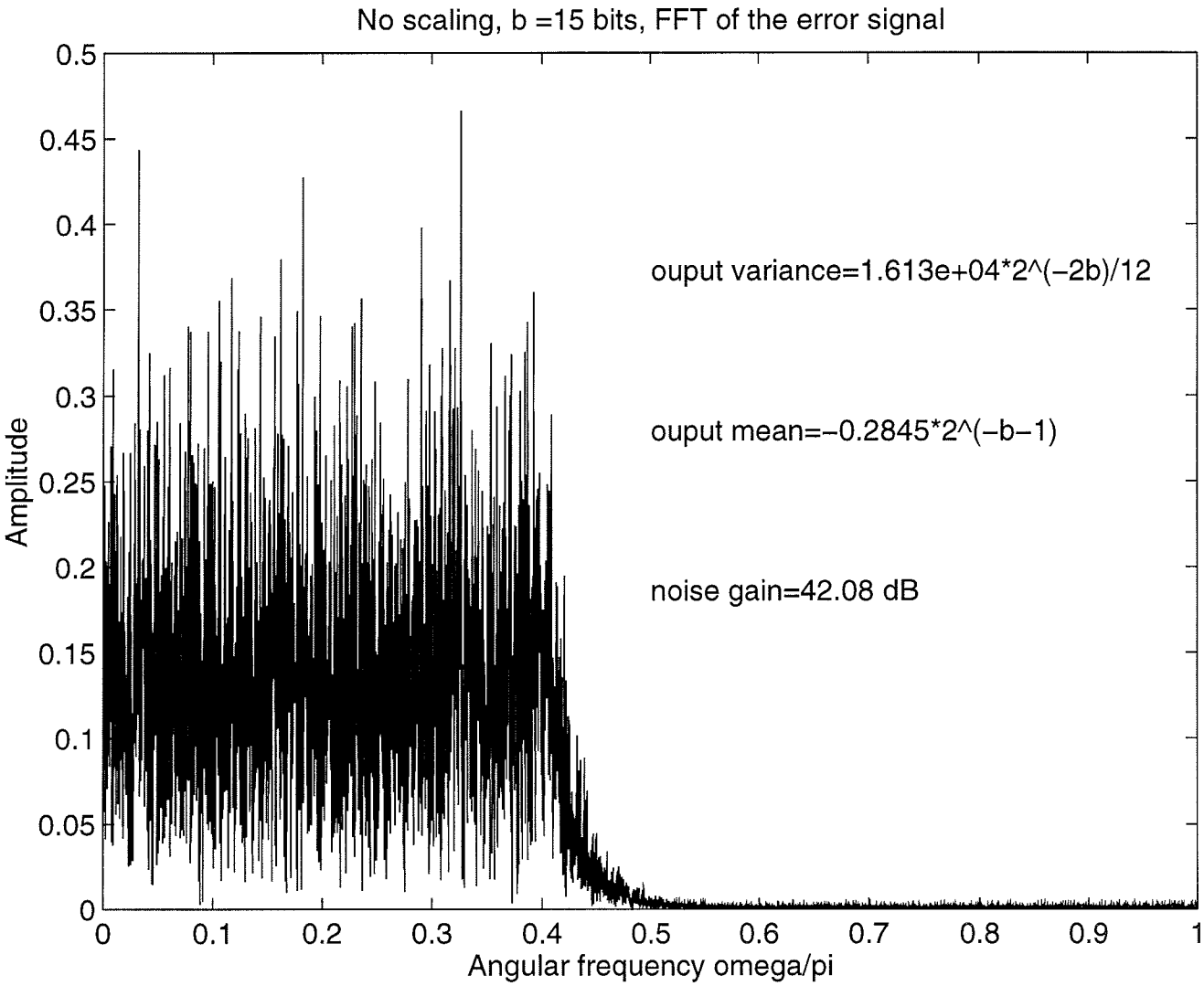
- The actual simulated response is then the ideal response plus the simulated error, where the error = the actual response − the ideal response.

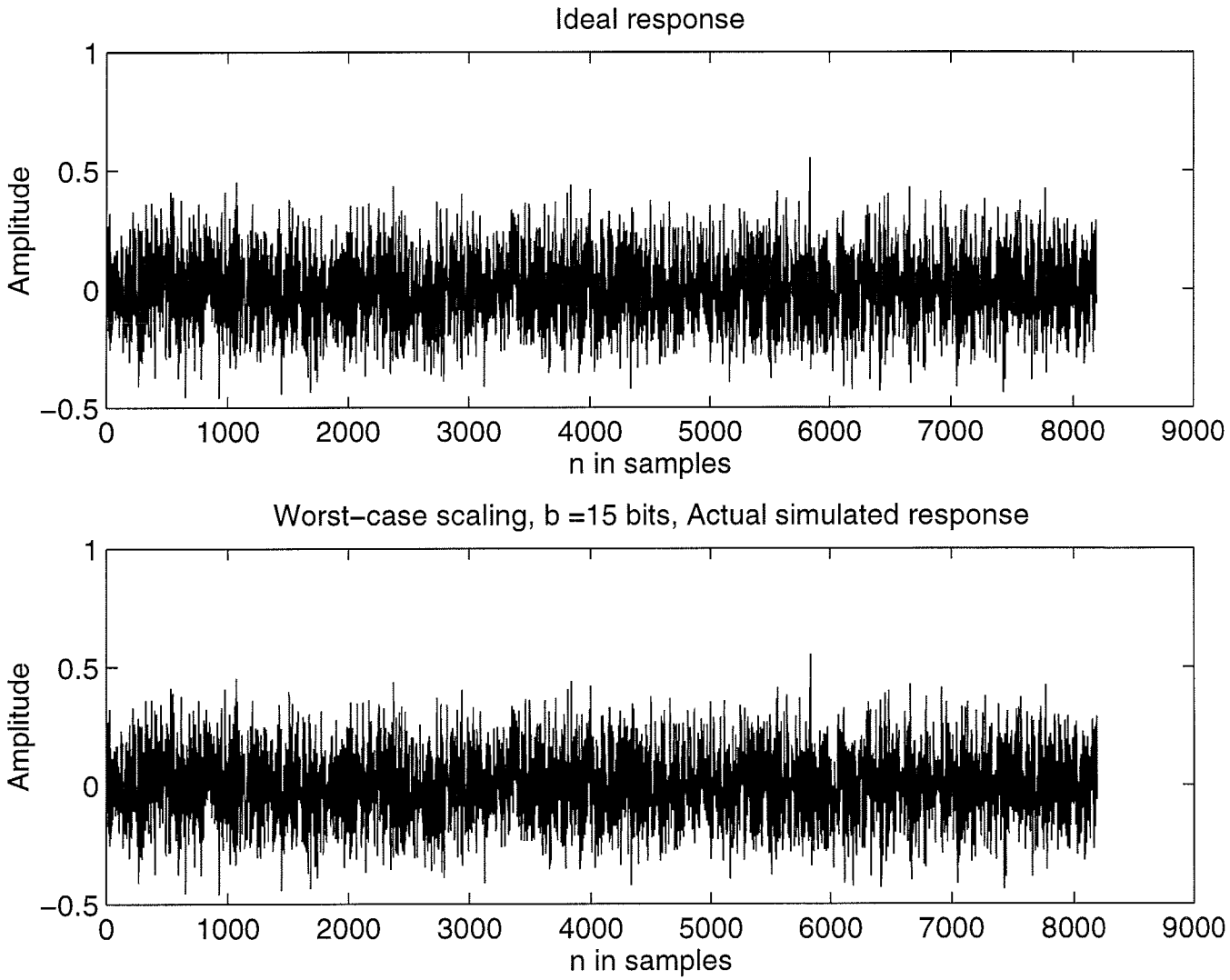
Results with uniformly distributed white noise and with rounding

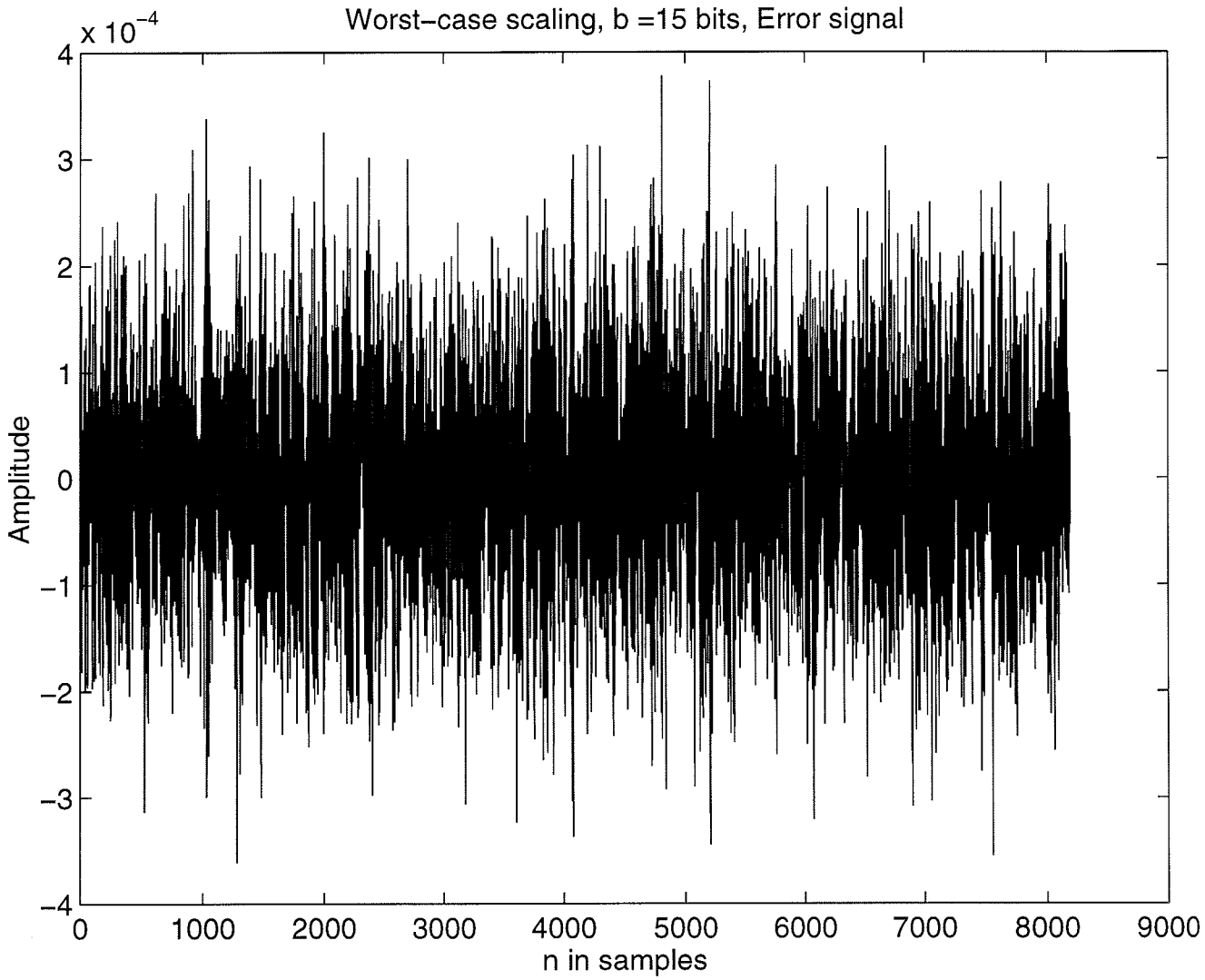
- As a test signal we use white noise of 2^{13} samples. This noise is uniformly distributed between -0.4 and 0.4.
- The following 12 transparencies give the simulation results for each of the four scaling norms for rounding after each multiplier.
- For each case, the ideal and actual response are given when the number of fractional bits for the data is $b = 15$.
- Also the difference between the simulated actual response and the ideal response is given both in the time and frequency domains.
- When comparing the numbers given in the FFT-figures with those of page 20, it is seen that the simulation results are very close to those based on the noise model.

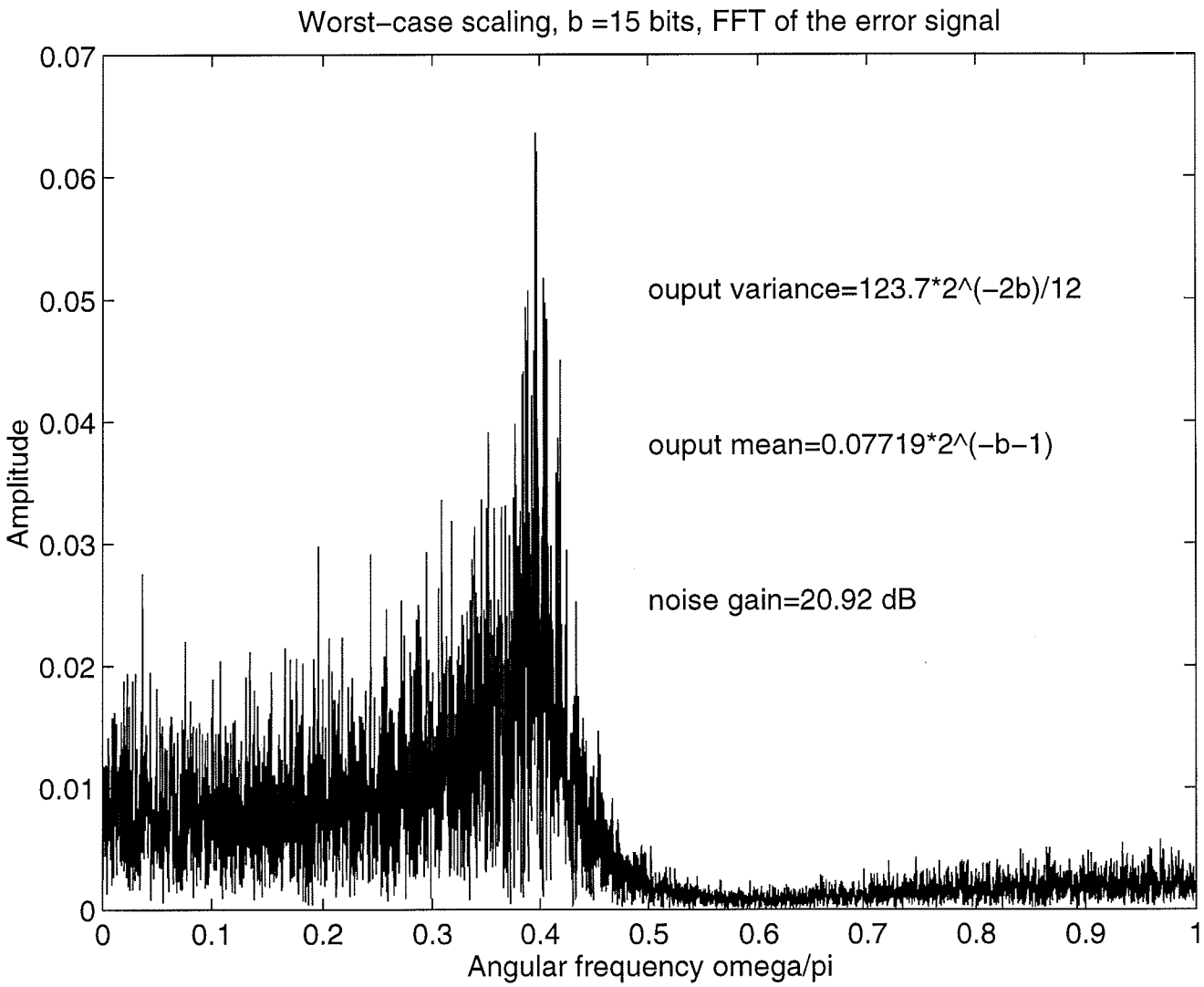

Ideal response
0.5
0
−0.5
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples
No scaling, b =15 bits, Actual simulated response
0.5
0
−0.5
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples

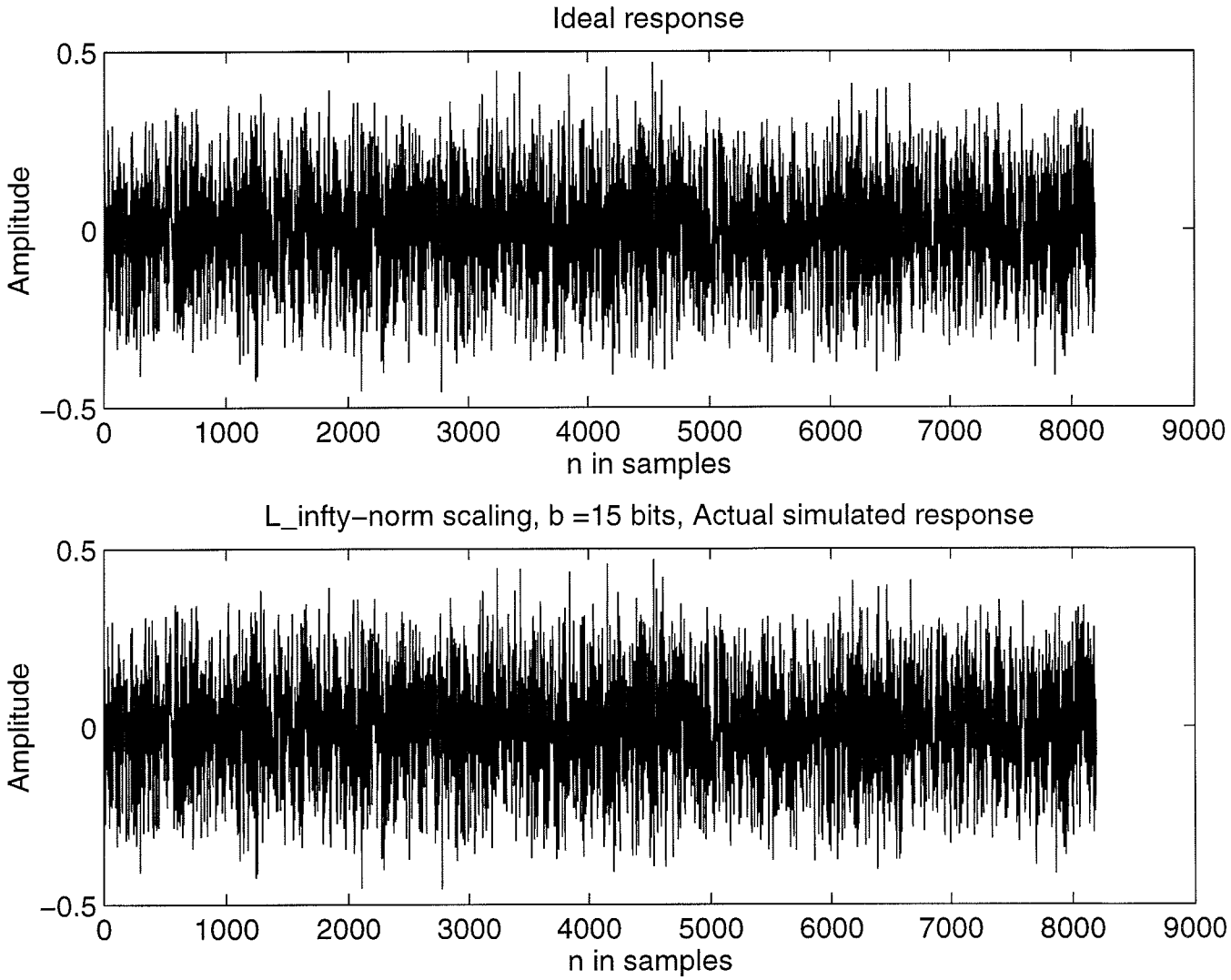

No scaling, b =15 bits, Error signal
x 10^-3
4
3
2
1
0
−1
−2
−3
−4
−5
−6
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples

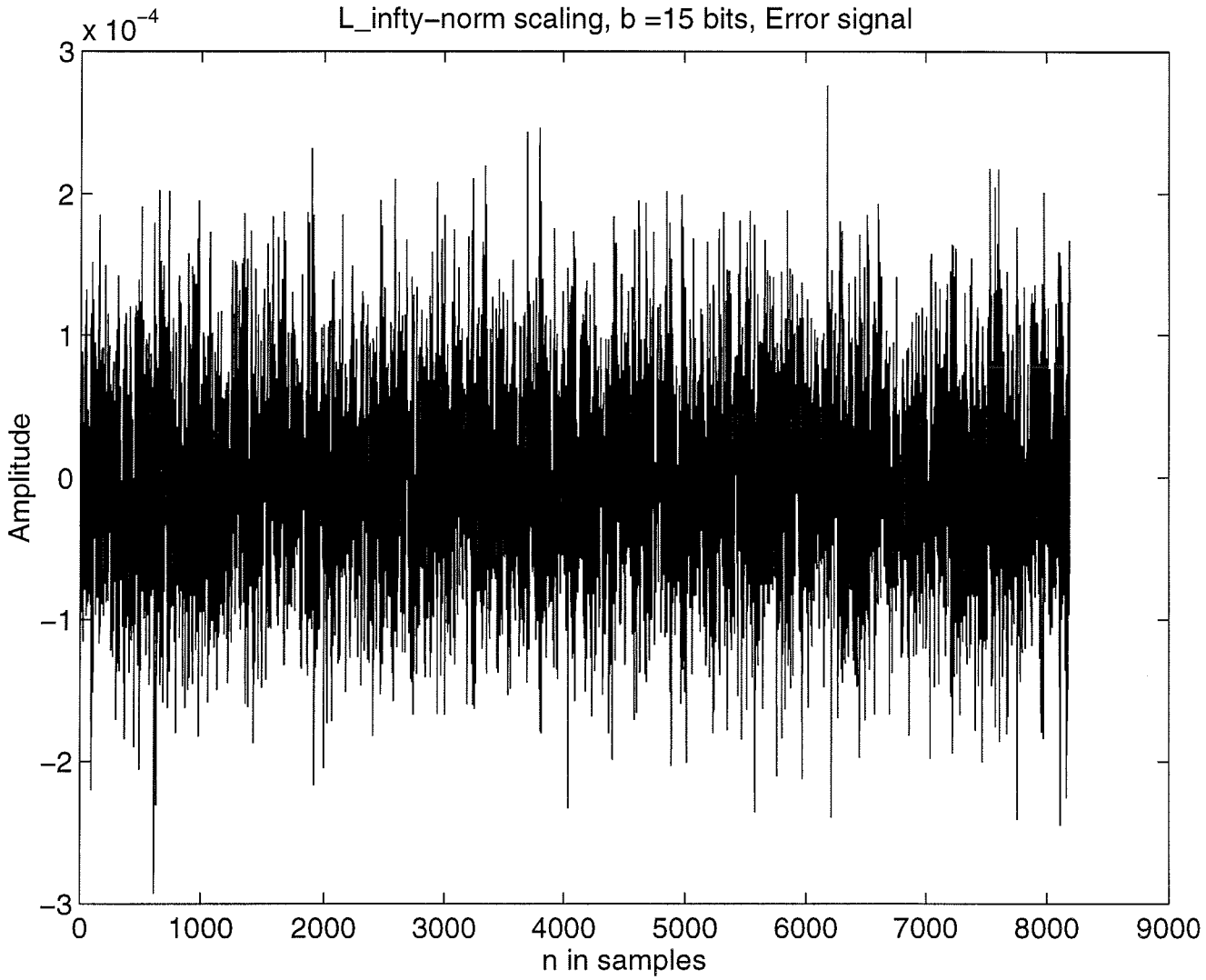

No scaling, b =15 bits, FFT of the error signal
ouput variance=1.613e+04*2^(−2b)/12
ouput mean=−0.2845*2^(−b−1)
noise gain=42.08 dB
Amplitude
Angular frequency omega/pi
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

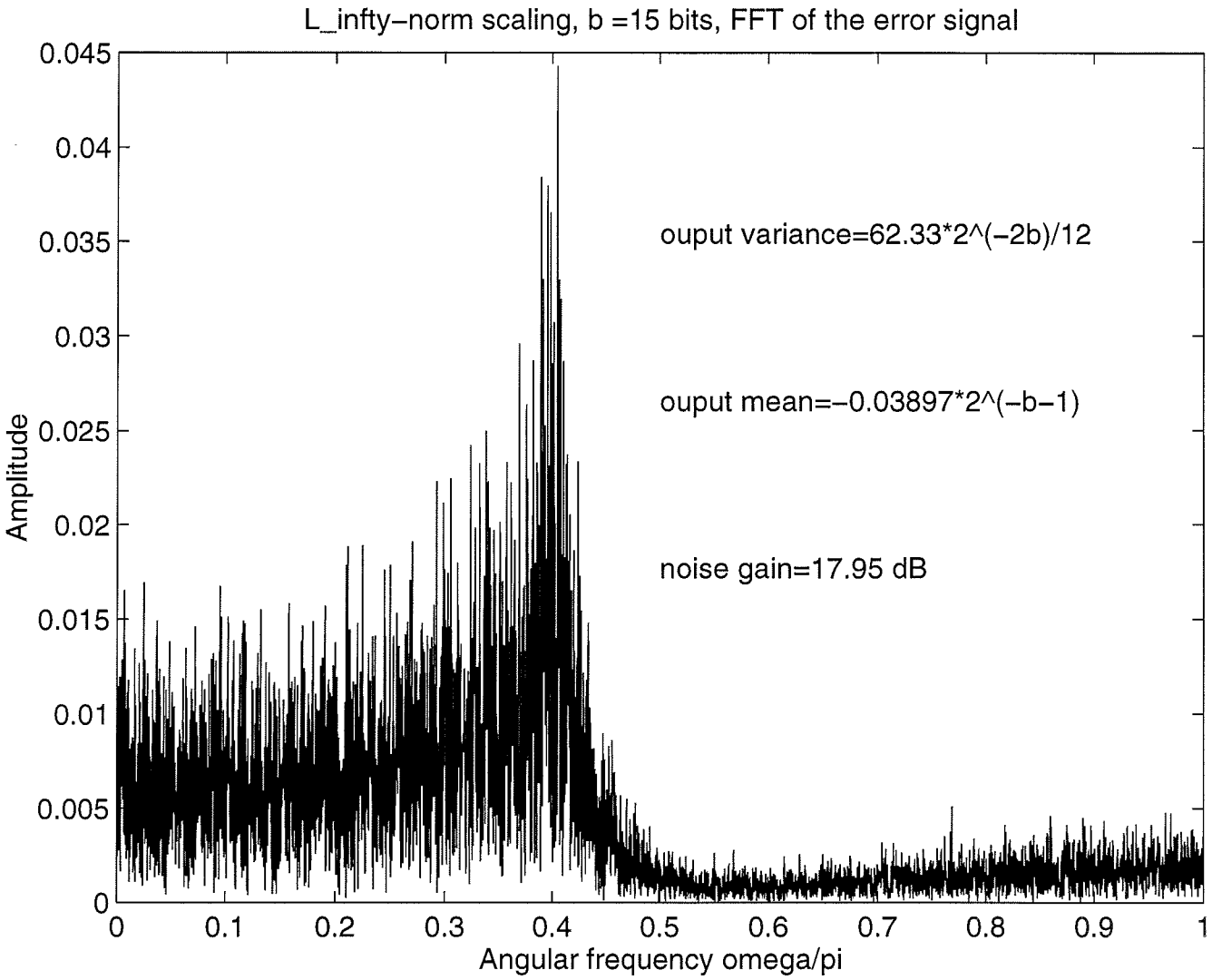

Ideal response
Amplitude
1
0.5
0
−0.5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples
Worst−case scaling, b =15 bits, Actual simulated response
Amplitude
1
0.5
0
−0.5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples

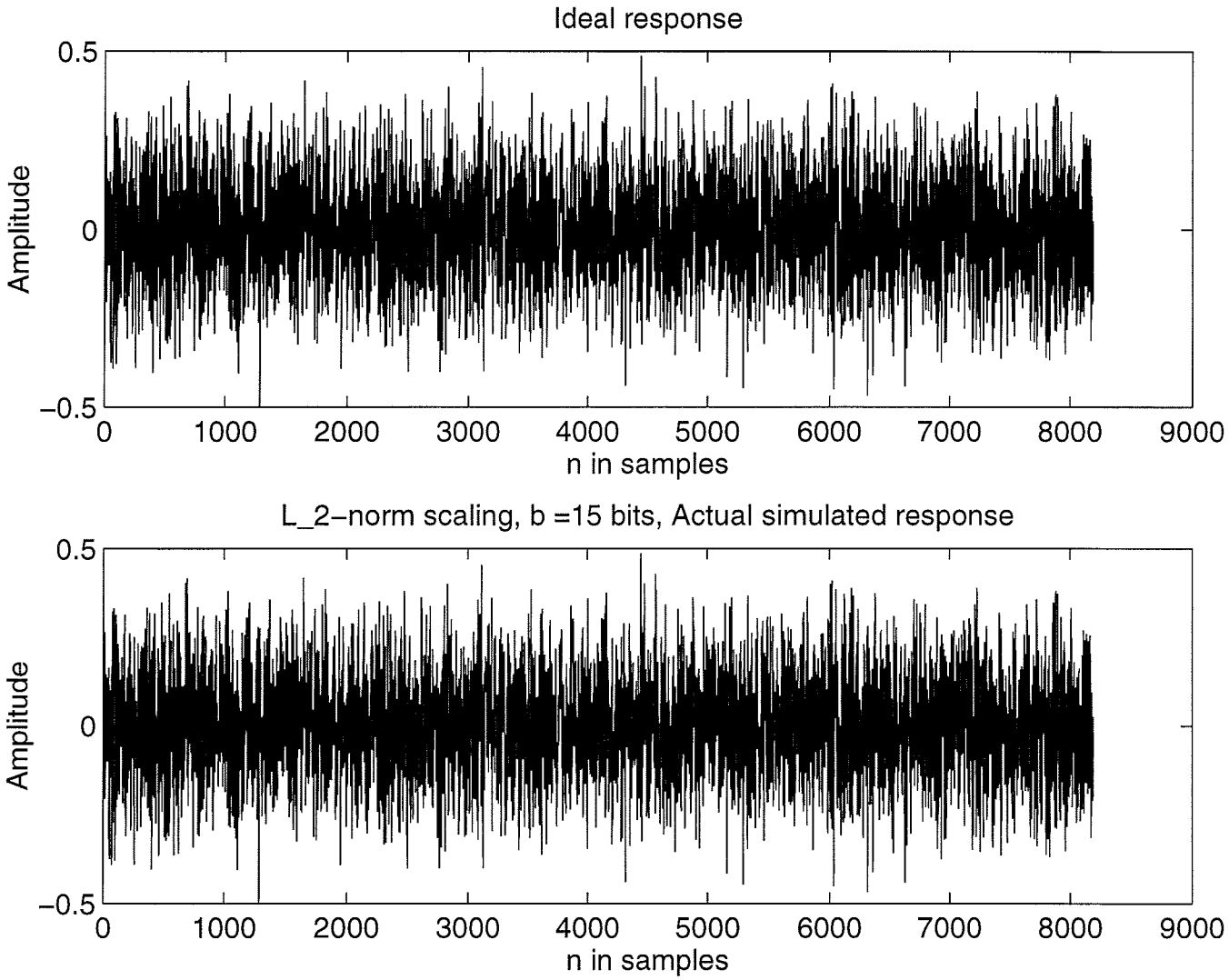

Worst−case scaling, b =15 bits, Error signal
x 10^−4
4
3
2
1
0
−1
−2
−3
−4
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples

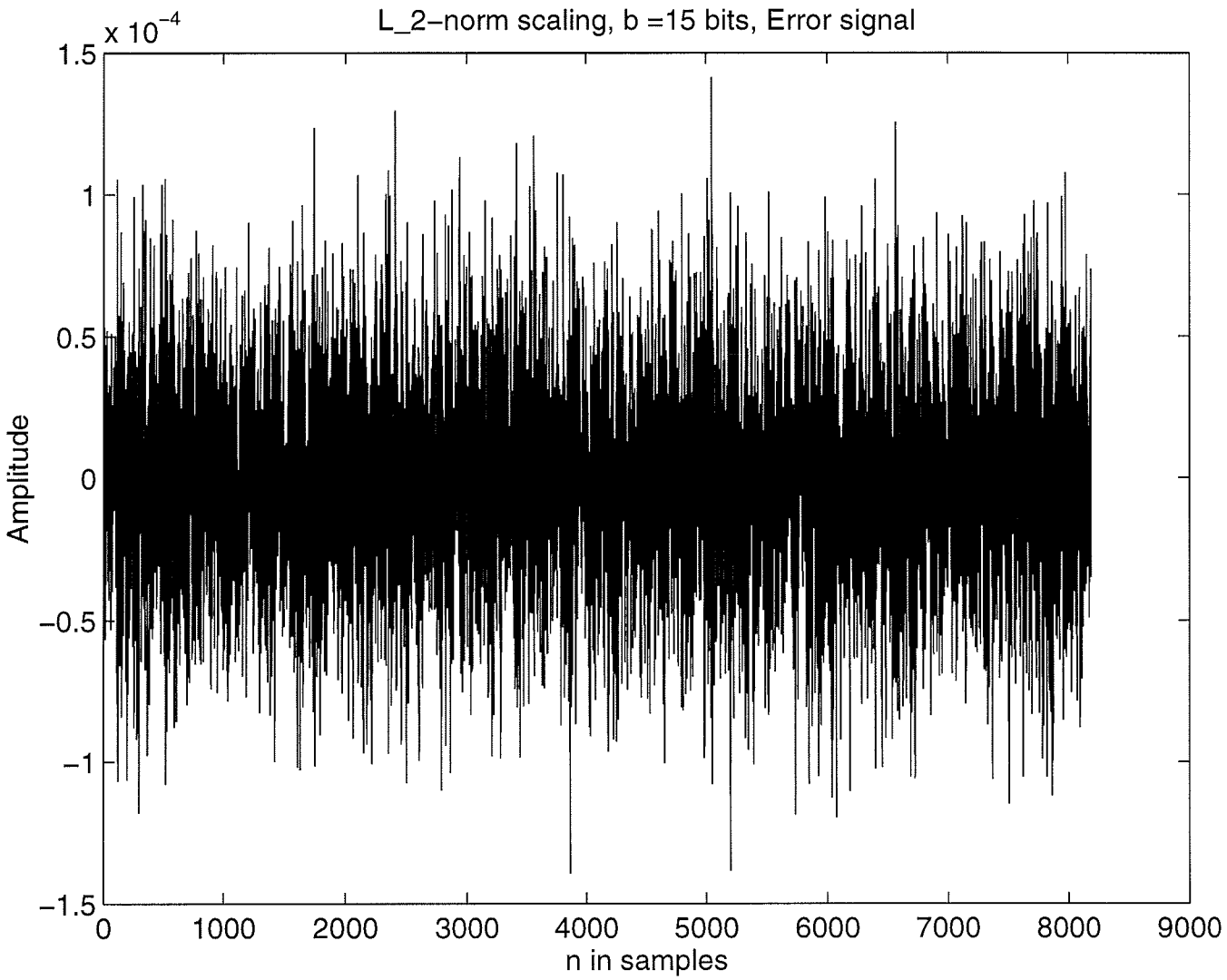

Worst–case scaling, b =15 bits, FFT of the error signal
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
Amplitude
ouput variance=123.7*2^(–2b)/12
ouput mean=0.07719*2^(–b–1)
noise gain=20.92 dB
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

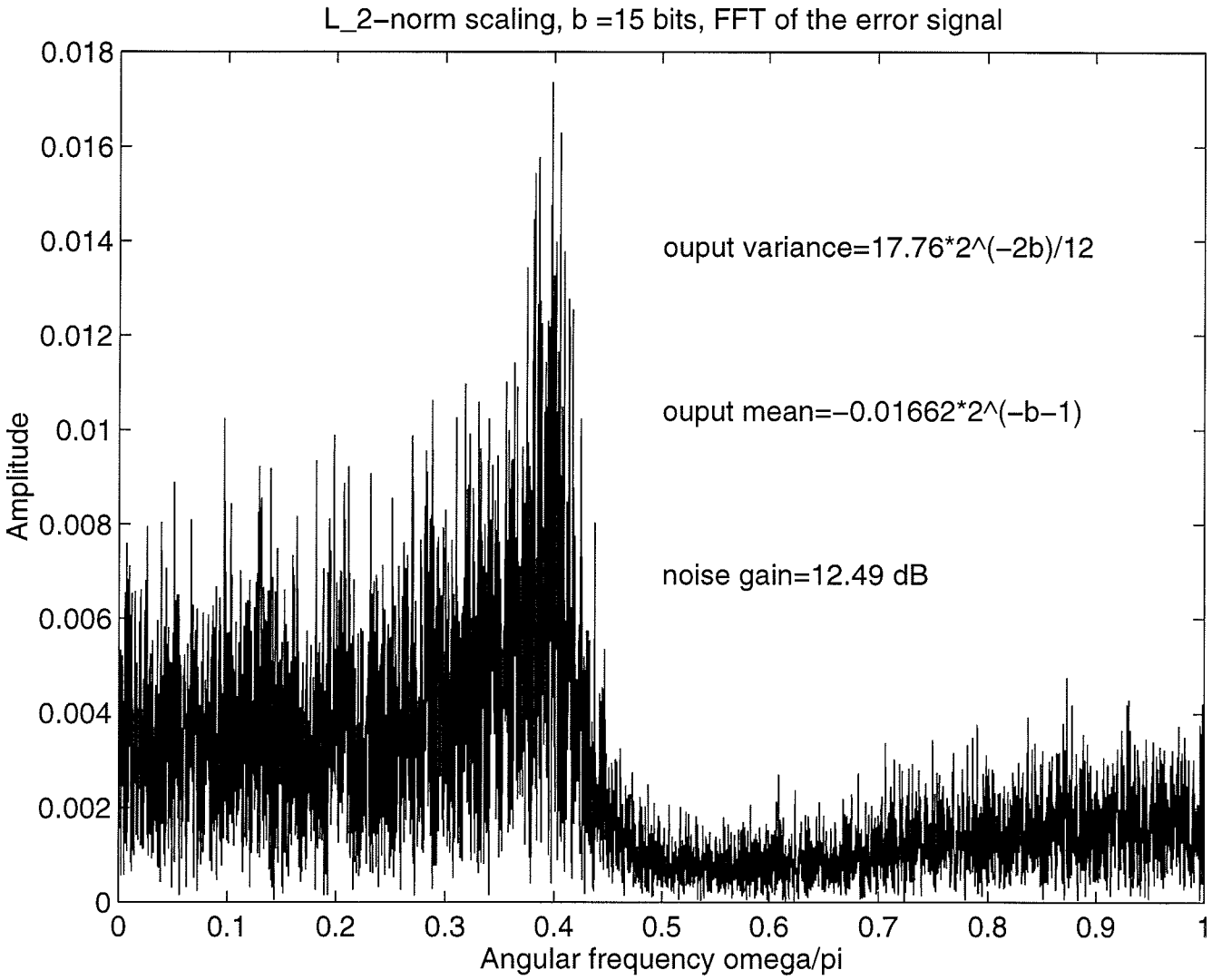

Ideal response
0.5
0
−0.5
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples
L_infty−norm scaling, b =15 bits, Actual simulated response
0.5
0
−0.5
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples


L_infty−norm scaling, b =15 bits, Error signal
x 10^−4
3
2
1
0
−1
−2
−3
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples


L_infty−norm scaling, b =15 bits, FFT of the error signal
ouput variance=62.33*2^(−2b)/12
ouput mean=−0.03897*2^(−b−1)
noise gain=17.95 dB
Amplitude
Angular frequency omega/pi
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Ideal response
0.5
0
−0.5
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples
L_2−norm scaling, b =15 bits, Actual simulated response
0.5
0
−0.5
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples

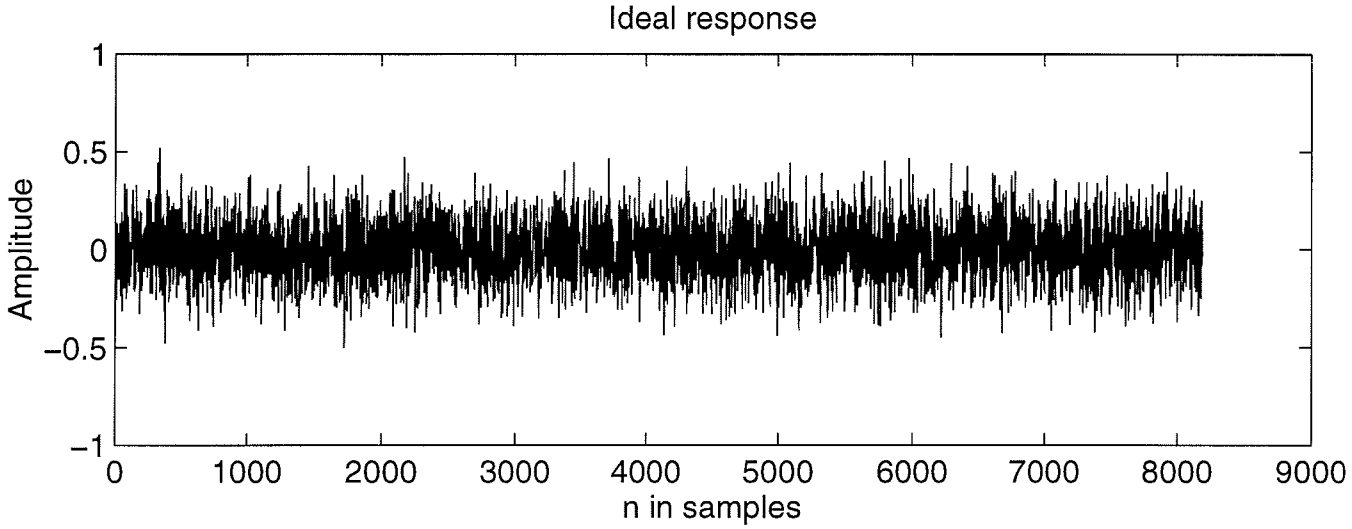

L_2−norm scaling, b =15 bits, Error signal
x 10^−4
1.5
1
0.5
0
−0.5
−1
−1.5
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples

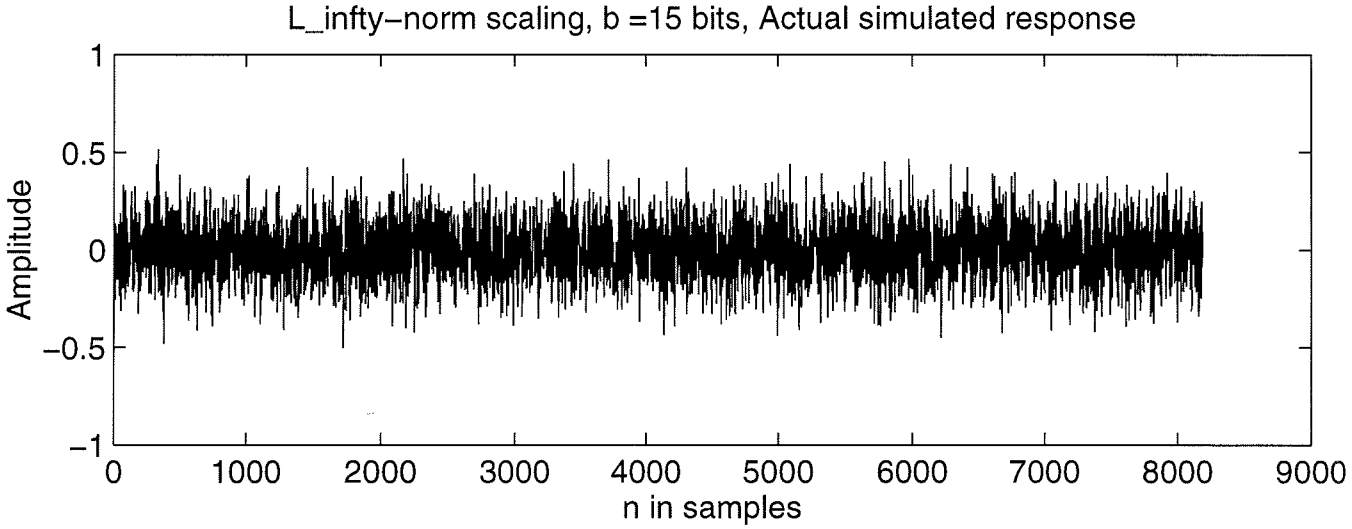

L_2−norm scaling, b =15 bits, FFT of the error signal
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
Amplitude
ouput variance=17.76*2^(−2b)/12
ouput mean=−0.01662*2^(−b−1)
noise gain=12.49 dB
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Angular frequency omega/pi

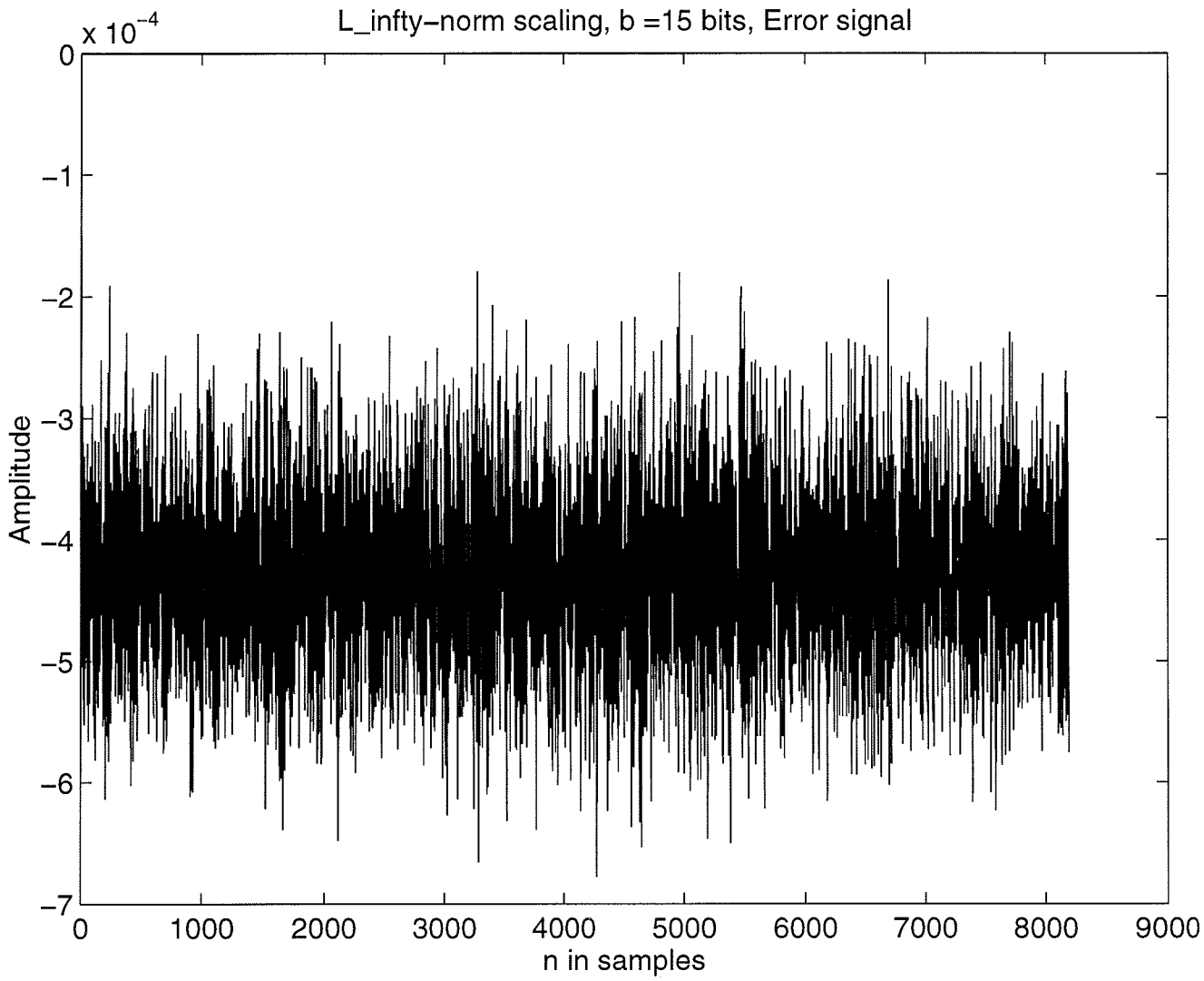
Results with uniformly distributed white noise and with truncation

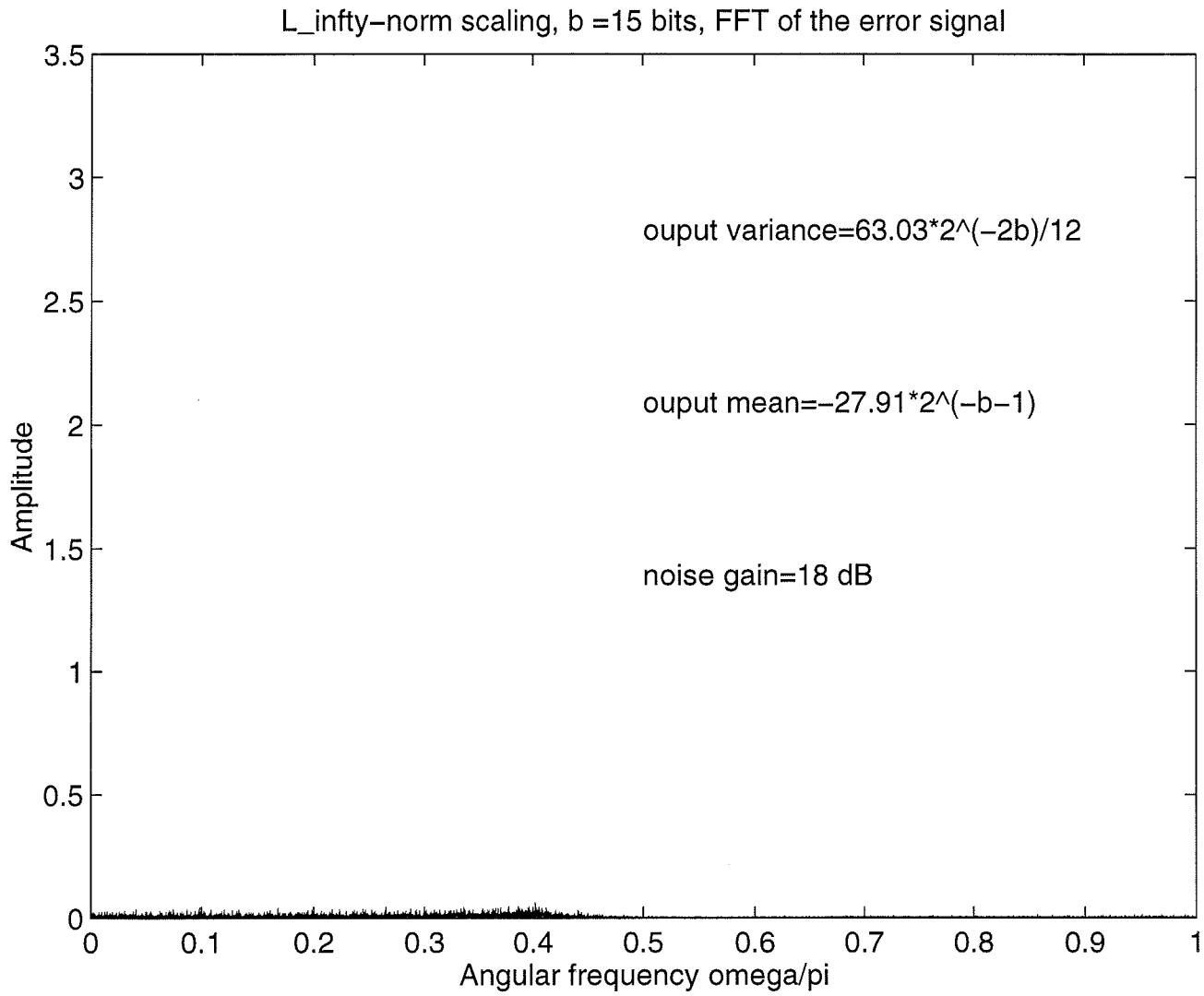
- Here we consider only the L_∞-norm scaling with $b = 15$ fractional bits for the data.
- As can be seen in the following three transparencies, in the case of truncation the output error has a rather high mean value, which is in fact dominating.
- In these simulations, the mean value is $-29.71 \cdot 2^{-b-1}$.
- For rounding, the mean value is practically equal to zero.
- This shows that it is preferrable to use rounding whenever possible.
- It is worth pointing out that in practical implemenations of digital filters it is often useful to increase the number of bits for internal calculations. If the number of extra bits is large enough, then when rounding the data in the filter output to the original number of bits, this corresponds to the case

where there is just a single noise sourse at the filter output.


Ideal response
Amplitude
1
0.5
0
−0.5
−1
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples

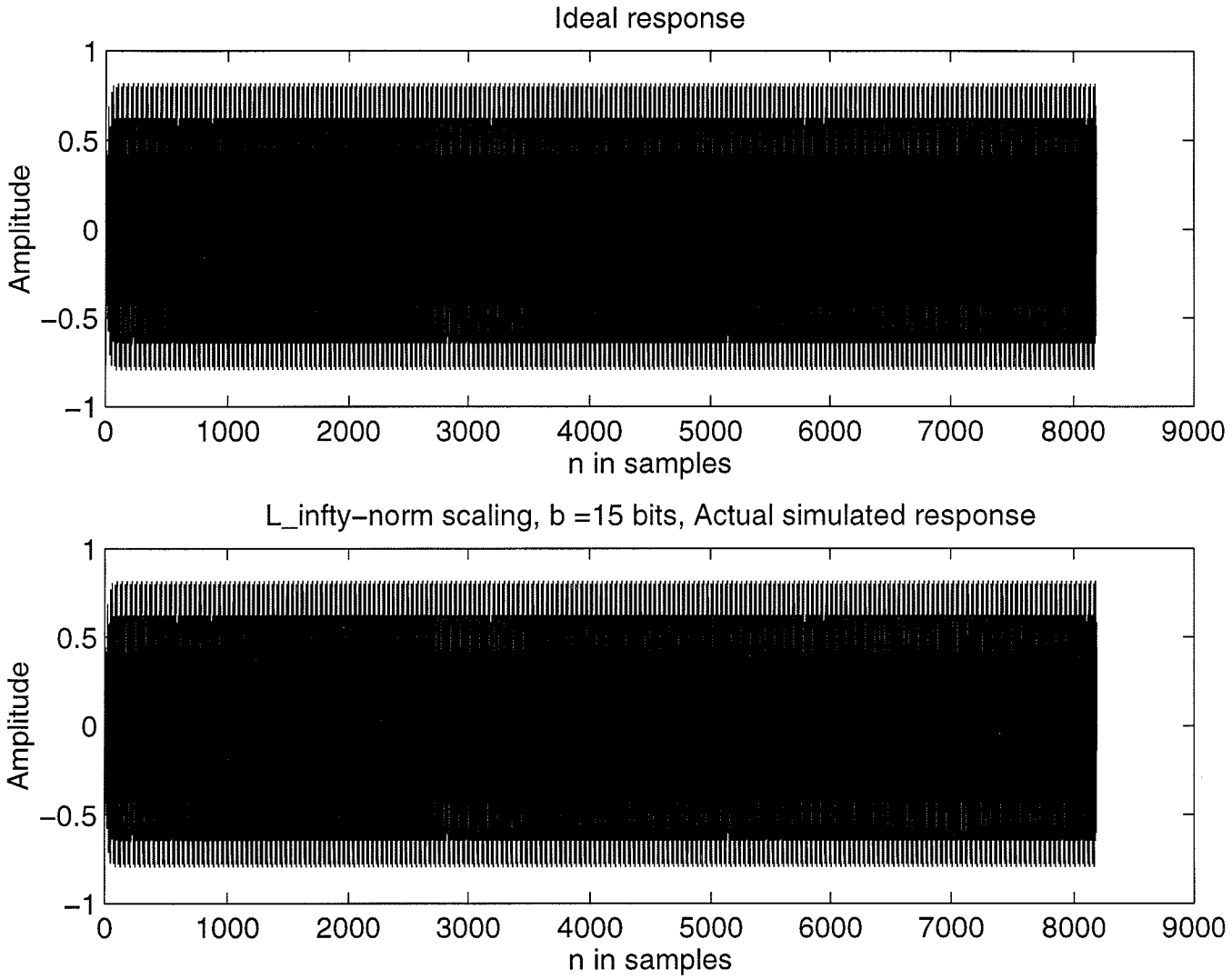

L_infty−norm scaling, b =15 bits, Actual simulated response
Amplitude
1
0.5
0
−0.5
−1
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples

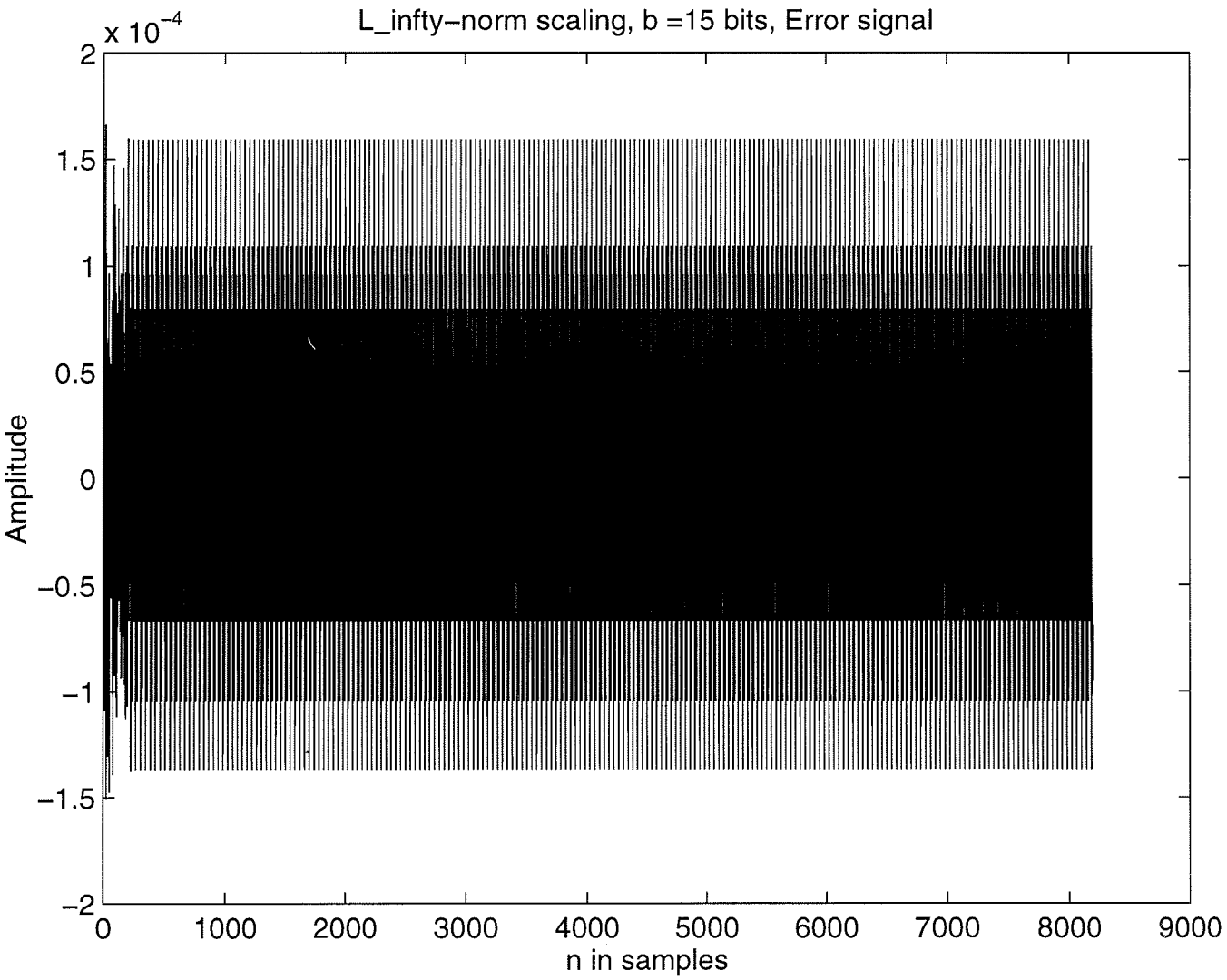

L_infty−norm scaling, b =15 bits, Error signal
x 10^−4
0
−1
−2
−3
−4
−5
−6
−7
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples

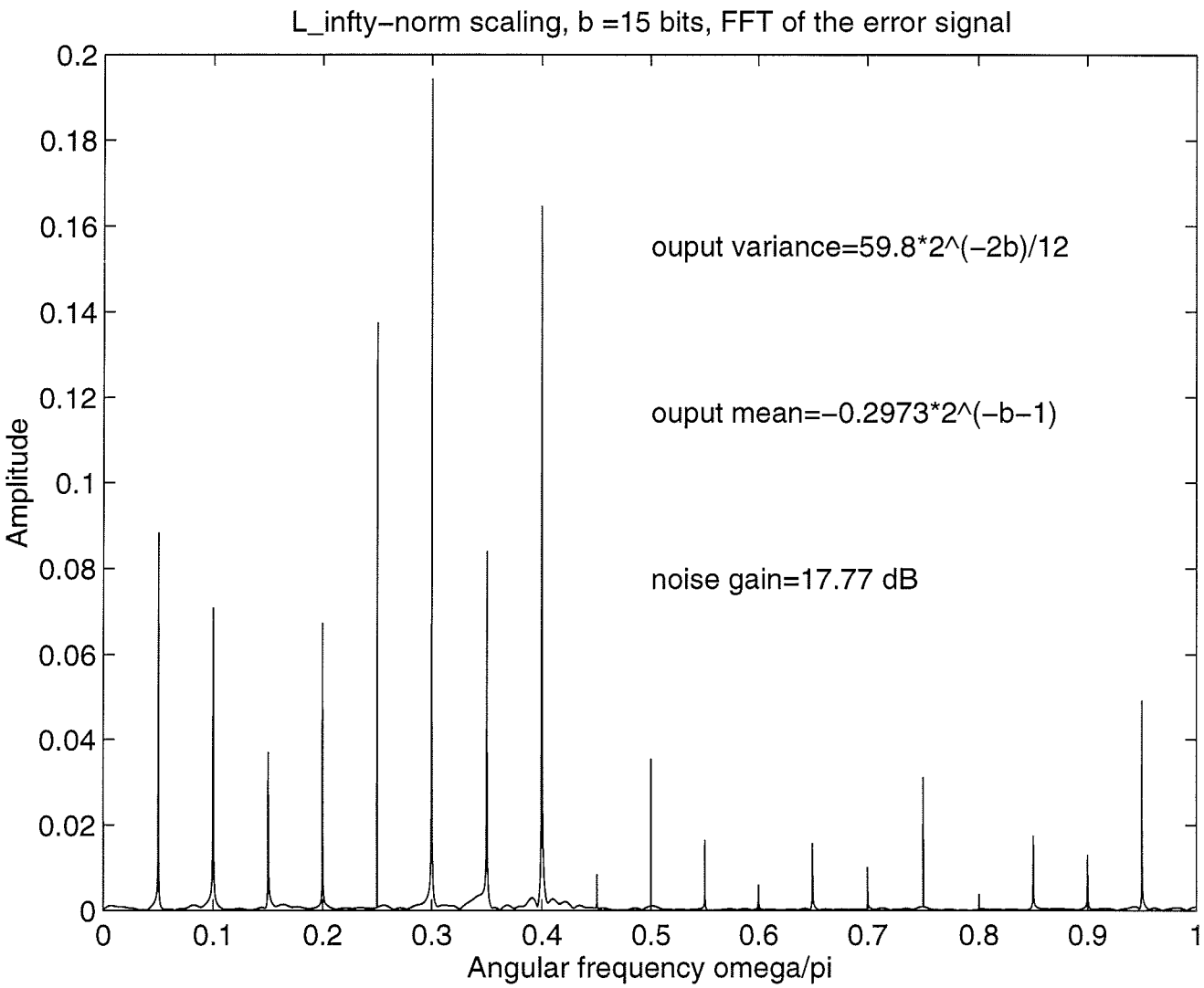

L_infty−norm scaling, b =15 bits, FFT of the error signal
ouput variance=63.03*2^(−2b)/12
ouput mean=−27.91*2^(−b−1)
noise gain=18 dB
Amplitude
Angular frequency omega/pi
0
0.5
1
1.5
2
2.5
3
3.5
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Results with the sinusoidal input

- The input signal is $0.5\sin(0.4\pi n) + 0.4\sin(0.25\pi n)$ and L_∞-norm scaling is used.
- Furthermore, the filter is implemented using two's complement arithmetic with rounding and $b = 15$ fractional bits for the data.
- As can be seen from the following three transprarencies, the error signal is not colored noise. It consists of several sinusoidals.
- This shows that the noise model is valid as long as the input signal is random enough.


Ideal response
Amplitude
1
0.5
0
−0.5
−1
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples
L_infty−norm scaling, b =15 bits, Actual simulated response
Amplitude
1
0.5
0
−0.5
−1
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples


L_infty–norm scaling, b =15 bits, Error signal
x 10^-4
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
Amplitude
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
n in samples


L_infty–norm scaling, b =15 bits, FFT of the error signal
ouput variance=59.8*2^(–2b)/12
ouput mean=–0.2973*2^(–b–1)
noise gain=17.77 dB
Amplitude
Angular frequency omega/pi
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

WARNING

- When preparing the above results, the lecturer noticed that even for the worst-case scaling there may occur an overflow at the filter output.
- This is because the sum of the absolute values of the impulse response from the input to the output is 2.316 for all the scaling norms (see pages 45–68 the lectures notes FINITE WORDLENGTH EFFECTS IN DIGITAL FILTERS)
- One alternative to get around this problem is to divide a_{03}, a_{13}, and a_{03} by 2.316. In this case the passband maximum is 1/2.316, instead of unity.
- Another alternative is to use two integer bits when multipying by a_{03}, a_{13}, and a_{03} and adding the results.
- **This problem has not been considered at all in the literature.**

```
% Matlab-file ordel.m for analysing elliptic filters
% can be found in ~ts/matlab/sldsp/ordel.m
% Tapio Saramaki 14.4.1997
%
 wp=input('Passband edge: ')
 ws=input('Stopband edge: ')
 disp('You can select two out the parameters:')
 disp('order, passband and stopband ripples')
 disp('and drop out one: 1 for order, 2 for')
 itype=input('passband ripple, 3 for stopband ripple: ')
 if itype==1
  Ap=input('Passband ripple in dB: ');
  As=input('Stopband ripple in dB: ');
 end
 if itype==2
  order=input('Order: ');
  As=input('Stopband ripple in dB: ');
 end
 if itype==3
  order=input('Order: ');
  Ap=input('Passband ripple in dB: ');
 end
 wwp=tan(wp*pi/2);
 wws=tan(ws*pi/2);
 k4=sqrt(wws/wwp);
 for i=1:4
  k4=k4*k4;
  k4=k4+sqrt(k4-1)*sqrt(k4+1);
 end
 if itype==1
  epss=sqrt(10^(As/10)-1);
  epsp=sqrt(10^(Ap/10)-1);
  order=8*log(4*epss/epsp)/log(2*k4)
 end
 if itype==3
  atte=exp(order*(log(2*k4)/8))/4;
  epss=sqrt(10^(Ap/10)-1)*atte;
  epss=epss*epss;
  As=10*log10(epss+1)
 end
 if itype==2
  atte=exp(order*(log(2*k4)/8))/4;
```

```
  epsp=sqrt(10^(As/10)-1)/atte;
  epsp=epsp*epsp;
  Ap=10*log10(epsp+1)
end
```

```
%Matlab-file iircoef.m for quantizing the coefficients
%of a sixth-order elliptic filter implemented as a cascade
%of three direct-form II blocks
%can be found in ~ts/matlab/sldsp
%
Ap=input('Passband ripple in dB: ');
As=input('Stopband ripple in dB: ');
wp=input('Passband edge/pi ');
[B,A] = ellip(6,Ap,As,wp);

%Poles and zeros
zer=roots(B);
pol=roots(A);
pol=sort(pol);
[Y,I]=sort(real(zer));
zer=zer(I);


it=input('1 for outermost pole pair last, 2 for first')
%Three numerator terms
%Note that zer(1) and zer(2) are complex conjugates
%The same is true for zer(3) and zer(4); and for
%zer(5) and zer(6)
if it==1 a1(1)=1;a1(3)=1;a1(2)=-2*real(zer(1));end
if it==2 a1(1)=1;a1(3)=1;a1(2)=-2*real(zer(5));end
a2(1)=1;a2(3)=1;a2(2)=-2*real(zer(3));
if it==1 a3(1)=1;a3(3)=1;a3(2)=-2*real(zer(5));end
if it==2 a3(1)=1;a3(3)=1;a3(2)=-2*real(zer(1));end

%Three denominator terms
if it==1
  b1(1)=1;b1(2)=-2*real(pol(1));
  b1(3)=real(pol(1))^2+imag(pol(1))^2;
end
if it==2
  b1(1)=1;b1(2)=-2*real(pol(5));
  b1(3)=real(pol(5))^2+imag(pol(5))^2;
end
b2(1)=1;b2(2)=-2*real(pol(3));
b2(3)=real(pol(3))^2+imag(pol(3))^2;
if it==1
  b3(1)=1;b3(2)=-2*real(pol(5));
```

```
  b3(3)=real(pol(5))^2+imag(pol(5))^2;
end
if it==2
  b3(1)=1;b3(2)=-2*real(pol(1));
  b3(3)=real(pol(1))^2+imag(pol(1))^2;
end
S=B(1);
mm=input('number of fractional bits: ')
disp('0 for no scaling, 1 for L_infty scaling, 2 for worst-case,')
it=input('3 for L_2 scaling')
%
%Scaling transfer functions
%
FN1(1)=S;
FD1=b1;
FN2=conv(FN1,a1);
FD2=conv(FD1,b2);
FN3=conv(FN2,a2);
FD3=conv(FD2,b3);
if it==1
  [H1,w]=freqz(FN1,FD1,8*1024);
  [H2,w]=freqz(FN2,FD2,8*1024);
  [H3,w]=freqz(FN3,FD3,8*1024);
  d1=max(abs(H1(1:8192)));
  d2=max(abs(H2(1:8192)));
  d3=max(abs(H3(1:8192)));
end
if it==2 | it==3
  [hh1,t]=impz(FN1,FD1,501);
  [hh2,t]=impz(FN2,FD2,501);
  [hh3,t]=impz(FN3,FD3,501);
end
if it==2
  d1=sum(abs(hh1));
  d2=sum(abs(hh2));
  d3=sum(abs(hh3));
end
if it==3
  d1=sqrt(sum(hh1.*hh1))
  d2=sqrt(sum(hh2.*hh2))
  d3=sqrt(sum(hh3.*hh3))
end
```

```
if it==1 | it==2 | it==3
  C1=1/d1;C2=1/(d2*C1);C3=1/(C1*C2*d3);
  S=C1*S;a1=C2*a1;a2=C3*a2;a3=a3/(C1*C2*C3);
end
%
%Rounding
%
S=round(2^mm*S)/(2^mm);end
for k=1:3
  a1(k)=round(2^mm*a1(k))/(2^mm);
  a2(k)=round(2^mm*a2(k))/(2^mm);
  a3(k)=round(2^mm*a3(k))/(2^mm);
  b1(k)=round(2^mm*b1(k))/(2^mm);
  b2(k)=round(2^mm*b2(k))/(2^mm);
  b3(k)=round(2^mm*b3(k))/(2^mm);
end
BB=S*a1;BB=conv(BB,a2);BB=conv(BB,a3);
AA=conv(b1,b2);AA=conv(AA,b3);
Ap=0.5
As=60.
[HH,w]=freqz(BB,AA,8*1024);
[H,w]=freqz(B,A,8*1024);
it=input('1 for scaling the maximum to be 0 dB, 0 for not')
cc=1;
if it==1 cc=max(abs(HH));end
figure(1)
plot(w/pi,20*log10(abs(HH)/cc),w/pi,20*log10(abs(H)),'- -');
axis([0 1 -100 10]);grid
title('Solid and dashed lines for quantized and ideal filters')
ylabel('Amplitude in dB');xlabel('Angular frequency omega/pi');
figure(2)
subplot(211);
plot(w/pi,20*log10(abs(HH)/cc),w/pi,20*log10(abs(H)),'- -');
axis([0 .2 -1.5*Ap 0]);grid
title('Passband: Solid and dashed lines for quantized and ideal
filters')
ylabel('Amplitude in dB');xlabel('Angular frequency omega/pi');
subplot(212);
plot(w/pi,20*log10(abs(HH)/cc),w/pi,20*log10(abs(H)),'- -');
axis([.3 1. -100 -50.]);grid
title('Stopband: Solid and dashed lines for quantized and ideal
filters')
```

```
ylabel('Amplitude in dB');xlabel('Angular frequency omega/pi');
figure(3)
zplane(B,A);title('Pole-zero plot for the ideal filter')
figure(4)
zplane(BB,AA);title('Pole-zero plot for the quantized filter')
```

```
% Matlab-file allcoe.m for quantizing the coefficients
% of a parallel connection of two allpass filters
% can be found in ~ts/matlab/sldsp
% Subroutine allphaa.m is required.
% The filter is of lowpass type and the order must
% be odd
% Tapio Saramaki 14.4.1997
 N=input('Filter order (odd)')
 wp=input('Passband edge/pi ');
 ws=input('Stopband edge/pi ');
 Ap=input('Passband ripple in dB: ');
 wwp=tan(wp*pi/2);
 wws=tan(ws*pi/2);
 k4=sqrt(wws/wwp);
 for i=1:4
  k4=k4*k4;
  k4=k4+sqrt(k4-1)*sqrt(k4+1);
 end
 atte=exp(N*(log(2*k4)/8))/4;
 epss=sqrt(10^(Ap/10)-1)*atte;
 epss=epss*epss;
 As=10*log10(epss+1)
 [B,A] = ellip(N,Ap,As,wp);
%
% Poles
%
 pol=roots(A);
 pol=sort(pol);
%
% First allpass section containing the real pole,
% the second innermost pole pair, the fourth innermost
% pole pair and so on. The denominator for the first
% order section is of the form 1+br(1)z^(-1). For
% the second-order sections of the form 1+b11(k)z^(-1)
% +b12(k)z^(-2) for k=1,2,...,N1. Here, N1=floor(N/4)
%
 N1=floor(N/4);
 br(1)=-pol(1);
 if N1 > 0
   for k=1:N1
     b=poly([pol(4*k) pol(4*k+1)]);
     b11(k)=b(2);b12(k)=b(3);
```

```
  end
 end
%
% Second allpass section containing the  innermost pole
% pair, the third innermost pole pair and so on. For
% the second-order sections of the form 1+b21(k)z^(-1)
% +b22(k)z^(-2) for k=1,2,...,N2. Here, N2=floor((N+2)/4)
%
 N2=floor((N+2)/4);
 if N2 > 0
  for k=1:N2
   b=poly([pol(4*k-2) pol(4*k-2+1)]);
   b21(k)=b(2);b22(k)=b(3);
  end
 end
 itype=input('1 for wave digital, 2 for direct-form');
 if itype==1
%
% for wave digital filters, the denominator for the first-order
% section is of the form 1-gamma*z^(-1). For the second-order
% section, it is of the form 1+gamma_2(gamma_1-1)z^(-1)
% -gamma_1*z^(-2).
%
  gr(1)=-br(1);
  if N1 > 0
   for k=1:N1
    g11(k)=-b12(k);
    g12(k)=-b11(k)/(1+b12(k));
   end
  end
  if N2 > 0
   for k=1:N2
    g21(k)=-b22(k);
    g22(k)=-b21(k)/(1+b22(k));
   end
  end
 end
%
 mm=input('number of fractional bits: ')
%
 if itype==2
  br(1)=round(2^mm*br(1))/(2^mm);
```

```
  if N1 > 0
    for k=1:N1
      b11(k)=round(2^mm*b11(k))/(2^mm);
      b12(k)=round(2^mm*b12(k))/(2^mm);
    end
  end
  if N2 > 0
    for k=1:N2
      b21(k)=round(2^mm*b21(k))/(2^mm);
      b22(k)=round(2^mm*b22(k))/(2^mm);
    end
  end
 end
%
 if itype==1
  gr(1)=round(2^mm*gr(1))/(2^mm);
  br(1)=-gr(1)
  if N1 > 0
     for k=1:N1
       g11(k)=round(2^mm*g11(k))/(2^mm);
       g12(k)=round(2^mm*g12(k))/(2^mm);
       b11(k)=g12(k)*(g11(k)-1);
       b12(k)=-g11(k);
     end
  end
  if N2 > 0
    for k=1:N2
      g21(k)=round(2^mm*g21(k))/(2^mm);
      g22(k)=round(2^mm*g22(k))/(2^mm);
      b21(k)=g22(k)*(g21(k)-1);
      b22(k)=-g21(k);
    end
  end
 end
%
% Resulting amplitude response
% We exploit the routine allphaa.m giving
% the the phase response of a first- or second-
% order section
%
% Phase of the first allpass section
 [ph1,w]=allphaa(1,br,.0,1.,8000);
```

```
if N1 > 0
  for k=1:N1
    bb(1)=b11(k);bb(2)=b12(k);
    [ph,w]=allphaa(2,bb,.0,1.,8000);
    ph1=ph1+ph;
  end
end
ph2=zeros(size(w));
if N2 > 0
  for k=1:N2
    bb(1)=b21(k);bb(2)=b22(k);
    [ph,w]=allphaa(2,bb,.0,1.,8000);
    ph2=ph2+ph;
  end
end
figure(1)
plot(w/pi,ph1/pi,w/pi,ph2/pi);
xlabel('Angular frequency omega/pi');
ylabel('Phase/pi');
title('Phase responses of the  quantized allpass sections');
He=exp(j*ph1)+exp(j*(ph2));
H=abs(He/2);
AAp=input('desired maximum passband ripple in dB for plot ');
AAs=input('desired maximum stopband ripple in dB for plot ');
[HH,z]=freqz(B,A,8000);
figure(2)
plot(w/pi,20*log10(abs(H)),z/pi,20*log10(abs(HH)),'- -');
axis([0 1 -2*AAs 10]);grid
title('Solid and dashed lines for quantized and ideal filters')
ylabel('Amplitude in dB');xlabel('Angular frequency omega/pi');
figure(3)
subplot(211);
plot(w/pi,20*log10(abs(H)),z/pi,20*log10(abs(HH)),'- -');
axis([0 wp -1.5*AAp 0]);grid
title('Passband: Solid and dashed lines for quantized and ideal
filters')
ylabel('Amplitude in dB');xlabel('Angular frequency omega/pi');
subplot(212);
plot(w/pi,20*log10(abs(H)),z/pi,20*log10(abs(HH)),'- -');
axis([ws 1. -1.5*AAs -.5*As]);grid
title('Stopband: Solid and dashed lines for quantized and ideal
filters')
```

```
ylabel('Amplitude in dB');xlabel('Angular frequency omega/pi');
%
AAA1=[1 br(1)]
if N1 > 0
  for k=1:N1
    AAA1=conv(AAA1, [1 b11(k) b12(k)]);
  end
 end
AAA2=[1]
if N2 > 0
 for k=1:N2
   AAA2=conv(AAA2, [1 b21(k) b22(k)]);
 end
end
%
% Denominator
%
AA=conv(AAA1,AAA2);
%
% Numerator
%
for k=1:length(AAA1)
  BBB1(k)=AAA1(length(AAA1)+1-k);
end
for k=1:length(AAA2)
  BBB2(k)=AAA2(length(AAA2)+1-k);
end
BB=conv(AAA1,BBB2)+conv(AAA2,BBB1);
figure(4)
zplane(B,A);title('Pole-zero plot for the ideal filter')
figure(5)
zplane(BB,AA);title('Pole-zero plot for the quantized filter')
```

```
% Matlab-file alphaa.m
% can be found in ~ts/matlab/sldsp/alphaa.m
% Matlab-files allcoe.m and alltest.m use
% this file as a subroutine
%
 function [A,w]=allphaa(l,b,u1,u2,M)
%
% evaluates the phase response of an allpass
% filter at points
% w=u1*pi:(u2-u1)*pi/M:u2*pi
% l=1 for first-order; denominator is
% 1+b(1)z^(-1)
% l=2 for second-order; debominator is
% 1+b(1)z^(-1)+b(2)z^(-1)
% Tapio Saramaki 15.4.1997
%
% Pole location for the first-order section
%
 if l==1 h(1)=-b(1); end
%
% Pole locations in the form z=h(1)exp(+-j*pi*h(2))
% for the second-order section
%
 if l==2 h(1)=sqrt(b(2));
 h(2)=acos(-b(1)/(2*h(1)))/pi;end
%
% For more than one point
%
 if M > 1 w=u1*pi:(u2-u1)*pi/M:u2*pi;end
%
% For a single point
%
 if M==1 w=u1*pi; end
%
 if l==1
   A=-w;
   ph1=h(1)*sin(w);
   ph2=1-h(1)*cos(w);
   A=A-2*atan2(ph1,ph2);
 end
 if l==2
   A=-2*w;
```

```
  ph1=h(1)*sin(w-pi*h(2));
  ph2=1-h(1)*cos(w-pi*h(2));
  A=A-2*atan2(ph1,ph2);
  ph1=h(1)*sin(w+pi*h(2));
  ph2=1-h(1)*cos(w+pi*h(2));
  A=A-2*atan2(ph1,ph2);
end
```

```
%Matlab-file alltest.m for quantizing the coefficients
%of a parallel connection of two allpass filters
%The filter is of lowpass type and the order must
%be odd
%Several values of Ap, the passband ripple  in dB,
%can be used
%can be found in ~ts/matlab/sldsp
N=input('Filter order (odd)')
%Ap=input('Passband ripple in dB: ');
%As=input('Stopband ripple in dB: ');
wp=input('Passband edge/pi ');
ws=input('Stopband edge/pi ');
itype=input('1 for wave digital, 2 for direct-form');
mm=input('number of fractional bits: ');
%wp=0.2;ws=0.3;N=7;itype=2;mm=7;
Ap1=input('Start passband ripple in dB: ');
Ap3=input('Stop passband ripple in dB: ');
Ap2=input('Step passband ripple in dB: ');
Asopt=0;
Apstar=0;
Apopt=0;
Asopt=0;
for Ap=Ap1:Ap2:Ap3
wwp=tan(wp*pi/2);
wws=tan(ws*pi/2);
k4=sqrt(wws/wwp);
for i=1:4
k4=k4*k4;
k4=k4+sqrt(k4-1)*sqrt(k4+1);
end
atte=exp(N*(log(2*k4)/8))/4;
epss=sqrt(10^(Ap/10)-1)*atte;
epss=epss*epss;
As=10*log10(epss+1)
[B,A] = ellip(N,Ap,As,wp);
%
%Poles
pol=roots(A);
pol=sort(pol);
%
%First allpass section containing the real pole,
%the second innermost pole pair, the fourth innermost
```

```
%pole pair and so on. The denominator for the first
%order section is of the form 1+br(1)z^(-1). For
%the second-order sections of the form 1+b11(k)z^(-1)
%+b12(k)z^(-2) for k=1,2,...,N1. Here, N1=
%=floor(N/4)
%
N1=floor(N/4);
br(1)=-pol(1);
if N1 > 0
  for k=1:N1
  b=poly([pol(4*k) pol(4*k+1)]);
  b11(k)=b(2);b12(k)=b(3);
  end
end
%
%Second allpass section containing the  innermost pole
%pair, the third innermost pole pair and so on. For
%the second-order sections of the form 1+b21(k)z^(-1)
%+b22(k)z^(-2) for k=1,2,...,N2. Here, N2=
%=floor((N+2)/4)
%
N2=floor((N+2)/4);
if N2 > 0
  for k=1:N2
  b=poly([pol(4*k-2) pol(4*k-2+1)]);
  b21(k)=b(2);b22(k)=b(3);
  end
end
%itype=input('1 for wave digital, 2 for direct-form');
if itype==1
%
%for wave digital filters, the denominator for the first-order
%section is of the form 1-gamma*z^(-1). For the second-order
%section, it is of the form 1+gamma_2(gamma_1-1)z^(-1)
%-gamma_1*z^(-2).
gr(1)=-br(1);
if N1 > 0
  for k=1:N1
  g11(k)=-b12(k);
  g12(k)=-b11(k)/(1+b12(k));
  end
end
```

```
if N2 > 0
  for k=1:N2
  g21(k)=-b22(k);
  g22(k)=-b21(k)/(1+b22(k));
  end
end
end
%
%mm=input('number of fractional bits: ')
%
if itype==2
  br(1)=round(2^mm*br(1))/(2^mm);
  if N1 > 0
    for k=1:N1
    b11(k)=round(2^mm*b11(k))/(2^mm);
    b12(k)=round(2^mm*b12(k))/(2^mm);
    end
  end
  if N2 > 0
    for k=1:N2
    b21(k)=round(2^mm*b21(k))/(2^mm);
    b22(k)=round(2^mm*b22(k))/(2^mm);
    end
  end
end
%
if itype==1
  gr(1)=round(2^mm*gr(1))/(2^mm);
  br(1)=-gr(1)
  if N1 > 0
    for k=1:N1
    g11(k)=round(2^mm*g11(k))/(2^mm);
    g12(k)=round(2^mm*g12(k))/(2^mm);
    b11(k)=g12(k)*(g11(k)-1);
    b12(k)=-g11(k);
    end
  end
  if N2 > 0
    for k=1:N2
    g21(k)=round(2^mm*g21(k))/(2^mm);
    g22(k)=round(2^mm*g22(k))/(2^mm);
    b21(k)=g22(k)*(g21(k)-1);
```

```
    b22(k)=-g21(k);
    end
  end
end
%
%Resulting amplitude response
%We exploit the routine allphaa.m giving
%the the phase response of a first- or second-
%order section
%
%Stopband response
%
%Phase of the first allpass section
[ph1,w]=allphaa(1,br,ws,1.,2000);
if N1 > 0
  for k=1:N1
  bb(1)=b11(k);bb(2)=b12(k);
  [ph,w]=allphaa(2,bb,ws,1.,2000);
  ph1=ph1+ph;
  end
end
ph2=zeros(size(w));
if N2 > 0
  for k=1:N2
  bb(1)=b21(k);bb(2)=b22(k);
  [ph,w]=allphaa(2,bb,ws,1.,2000);
  ph2=ph2+ph;
  end
end
He=exp(j*ph1)+exp(j*(ph2));
H=abs(He/2);
Ass=20*log10(max(abs(H)));
%
%Passband response
%
%Phase of the first allpass section
[ph1,w]=allphaa(1,br,0.,wp,2000);
if N1 > 0
  for k=1:N1
  bb(1)=b11(k);bb(2)=b12(k);
  [ph,w]=allphaa(2,bb,0.,wp,2000);
  ph1=ph1+ph;
```

```
  end
end
ph2=zeros(size(w));
if N2 > 0
  for k=1:N2
  bb(1)=b21(k);bb(2)=b22(k);
  [ph,w]=allphaa(2,bb,0.,wp,2000);
  ph2=ph2+ph;
  end
end
He=exp(j*ph1)+exp(j*(ph2));
H=abs(He/2);
App=-20*log10(min(abs(H)));
Ap
As
App
Ass
Apstar
Asopt
Apopt
if Ass < Asopt & App < 0.5 Asopt=Ass;
Apopt=App; Apstar=Ap;end
end
Apstar
Asopt
Apopt
```

```
% Matlab-file iirnois.m for analysing the performance of
% a sixth-order elliptic filter implemented as a cascade
% of three second-order direct-form II blocks
% can be found in ~ts/matlab/sldsp
% subroutine dirgen.m is required
%
Ap=input('Passband ripple in dB: ');
As=input('Stopband ripple in dB: ');
wp=input('Passband edge/pi ');
[B,A] = ellip(6,Ap,As,wp);
[HH,w]=freqz(B,A,8*1024);
figure(1)
subplot(211)
plot(w/pi,20*log10(abs(HH)));
axis([0 1 -2*As 10]);
title('Amplitude response for the overall filter H(z)')
ylabel('Amplitude in dB');
xlabel('Angular frequency omega/pi');
subplot(212);plot(w/pi,20*log10(abs(HH)));
axis([0 wp -Ap 0]);
title('Passband details')
ylabel('Amplitude in dB');
xlabel('Angular frequency omega/pi');
figure(2)
zplane(B,A);title('Pole-zero plot for the overall filter H(z)')
%
%Poles and zeros
%
zer=roots(B);
pol=roots(A);
pol=sort(pol);
[Y,I]=sort(real(zer));
zer=zer(I);
%
it=input('1 for outermost pole pair last, 2 for first')
% Three numerator terms
% Note that zer(1) and zer(2) are complex conjugates
% The same is true for zer(3) and zer(4); and for
% zer(5) and zer(6)
if it==1 a1(1)=1;a1(3)=1;a1(2)=-2*real(zer(1));end
if it==2 a1(1)=1;a1(3)=1;a1(2)=-2*real(zer(5));end
a2(1)=1;a2(3)=1;a2(2)=-2*real(zer(3));
```

```
if it==1 a3(1)=1;a3(3)=1;a3(2)=-2*real(zer(5));end
if it==2 a3(1)=1;a3(3)=1;a3(2)=-2*real(zer(1));end
%
%Three denominator terms
%
if it==1
  b1(1)=1;b1(2)=-2*real(pol(1));
  b1(3)=real(pol(1))^2+imag(pol(1))^2;
end
if it==2
  b1(1)=1;b1(2)=-2*real(pol(5));
  b1(3)=real(pol(5))^2+imag(pol(5))^2;
end
b2(1)=1;b2(2)=-2*real(pol(3));
b2(3)=real(pol(3))^2+imag(pol(3))^2;
if it==1
  b3(1)=1;b3(2)=-2*real(pol(5));
  b3(3)=real(pol(5))^2+imag(pol(5))^2;
end
if it==2
  b3(1)=1;b3(2)=-2*real(pol(1));
  b3(3)=real(pol(1))^2+imag(pol(1))^2;
end
S=B(1);
disp('0 for no scaling, 1 for L_infty scaling, 2 for worst-case,')
it=input('3 for L_2 scaling')
%
% Scaling transfer functions
%
FN1(1)=S;
FD1=b1;
FN2=conv(FN1,a1);
FD2=conv(FD1,b2);
FN3=conv(FN2,a2);
FD3=conv(FD2,b3);
%
if it==1
  [H1,w]=freqz(FN1,FD1,8*1024);
  [H2,w]=freqz(FN2,FD2,8*1024);
  [H3,w]=freqz(FN3,FD3,8*1024);
  d1=max(abs(H1(1:8192)))
  d2=max(abs(H2(1:8192)))
```

```
  d3=max(abs(H3(1:8192)))
end
%
if it==2 | it==3
  [hh1,t]=impz(FN1,FD1,501);
  [hh2,t]=impz(FN2,FD2,501);
  [hh3,t]=impz(FN3,FD3,501);
end
%
if it==2
  d1=sum(abs(hh1));
  d2=sum(abs(hh2));
  d3=sum(abs(hh3));
end
%
if it==3
  d1=sqrt(sum(hh1.*hh1))
  d2=sqrt(sum(hh2.*hh2))
  d3=sqrt(sum(hh3.*hh3))
end
%
if it==1 | it==2 | it==3
  C1=1/d1;C2=1/(d2*C1);C3=1/(C1*C2*d3);
  S=C1*S;a1=C2*a1;a2=C3*a2;a3=a3/(C1*C2*C3);
end
FN1(1)=S;
FD1=b1;
FN2=conv(FN1,a1);
FD2=conv(FD1,b2);
FN3=conv(FN2,a2);
FD3=conv(FD2,b3);
FN4=conv(FN3,a3);
FD4=FD3;
AA1=a1;BB1=b1;CC1(1)=S;CC1(2)=1;
AA2=a2;BB2=b2;CC2(1)=1;CC2(2)=1;
AA3=a3;BB3=b3;CC3(1)=1;CC3(2)=1;
[H1,w]=freqz(FN1,FD1,8*1024);
[H2,w]=freqz(FN2,FD2,8*1024);
[H3,w]=freqz(FN3,FD3,8*1024);
[H4,w]=freqz(FN4,FD4,8*1024);
clear t;clear hh1;clear hh2;clear hh3;clear hh;
[hh1,t]=impz(FN1,FD1,101);
```

```
[hh2,t]=impz(FN2,FD2,101);
[hh3,t]=impz(FN3,FD3,101);
[hh,t]=impz(B,A,101);
%
%Amplitude responses
%
figure(3)
subplot(2,1,1);plot(w/pi,abs(H1));
title('Amplitude response for F1(z)')
ylabel('Amplitude');xlabel('Angular frequency omega/pi');
subplot(2,1,2);plot(w/pi,abs(H2));
title('Amplitude response for F2(z)')
ylabel('Amplitude');xlabel('Angular frequency omega/pi');
figure(4)
subplot(2,1,1);plot(w/pi,abs(H3));
title('Amplitude response for F3(z)')
ylabel('Amplitude');xlabel('Angular frequency omega/pi');
subplot(2,1,2);plot(w/pi,abs(H4));
title('Amplitude response for H(z)')
ylabel('Amplitude');xlabel('Angular frequency omega/pi');
%
%Impulse responses
%
figure(5)
subplot(2,1,1);impz(hh1);title('Impulse response for F1(z)');
ylabel('f1(n)');xlabel('n in samples');
text(15,.8*max(hh1),['sum of the absolute values of  f1(n)s ='...
,num2str(sum(abs(hh1)))]);
text(15,.55*max(hh1),['square root of the sum of the f1(n)^2s ='...
,num2str(sqrt(sum(hh1.*hh1)))]);
subplot(2,1,2);impz(hh2);title('Impulse response for F2(z)')
ylabel('f2(n)');xlabel('n in samples');
text(15,.8*max(hh2),['sum of the absolute values of  f2(n)s ='...
,num2str(sum(abs(hh2)))]);
text(15,.55*max(hh2),['square root of the sum of the f2(n)^2s ='...
,num2str(sqrt(sum(hh2.*hh2)))]);
figure(6)
subplot(2,1,1);impz(hh3);title('Impulse response for F3(z)')
ylabel('f3(n)');xlabel('n in samples');
text(15,.8*max(hh3),['sum of the absolute values of  f3(n)s ='...
,num2str(sum(abs(hh3)))]);
text(15,.55*max(hh3),['square root of the sum of the f3(n)^2s ='...
```

```
,num2str(sqrt(sum(hh3.*hh3)))]);
subplot(2,1,2);impz(hh);title('Impulse response for H(z)')
ylabel('h(n)');xlabel('n in samples');
text(15,.8*max(hh),['sum of the absolute values of  h(n)s ='...
,num2str(sum(abs(hh)))]);
text(15,.55*max(hh1),['square root of the sum of the h(n)^2s ='...
,num2str(sqrt(sum(hh.*hh)))]);
nn=input('number of samples: ')
mm=input('number of fractional bits')
nn1=1:nn;
disp('input signal of the form
d1*sin*(pi*d2*n+d3)+e1*sin*(pi*e2*n+e3)+')
disp('f1*2*(radn(1,n)-1)')
d1=input('d1= ');
d2=input('d2= ');
d3=input('d3= ');
e1=input('e1= ');
e2=input('e2= ');
e3=input('e3= ');
f1=input('f1= ');
nn1=1:nn;
in=d1*sin(pi*d2*nn1+d3)+e1*sin(pi*e2*nn1+e3);
in=in+f1*(2*rand(1,nn)-1);
%ity=input('1 for scaling the input, 2 not: ')
%if ity==1 f=abs(max(in));f=(1-2^(-mm))/f
%in=f*in; end
%mm fractional bits
in=round(2^(mm)*in)/(2^(mm));
out2=filter(B,A,in);
ntype=input('1 for truncation, 2 for rounding')
disp('1 rounding or quantization after multipliers,')
ktype=input('2 after adders')
out1=dirgen(in,AA1,BB1,CC1,mm,ktype,ntype);
out1=dirgen(out1,AA2,BB2,CC2,mm,ktype,ntype);
out1=dirgen(out1,AA3,BB3,CC3,mm,ktype,ntype);
nn3=1:length(in);
figure(7);subplot(211);
plot(nn3,out2);title('Ideal response');
ylabel('Amplitude');xlabel('n in samples');
subplot(212)
plot(nn3,out1);
if it==0 title(['No scaling, b ='...
```

```
, num2str(mm), ' bits, Actual simulated response']);
end
if it==1 title(['L_infty-norm scaling, b ='...
, num2str(mm), ' bits, Actual simulated response']);
end
if it==2 title(['Worst-case scaling, b ='...
, num2str(mm), ' bits, Actual simulated response']);
end
if it==3 title(['L_2-norm scaling, b ='...
, num2str(mm), ' bits, Actual simulated response']);
end
ylabel('Amplitude');xlabel('n in samples');
figure(8)
plot(nn3,out1-out2);
if it==0 title(['No scaling, b ='...
, num2str(mm), ' bits, Error signal']);
end
if it==1 title(['L_infty-norm scaling, b ='...
, num2str(mm), ' bits, Error signal']);
end
if it==2 title(['Worst-case scaling, b ='...
, num2str(mm), ' bits, Error signal']);
end
if it==3 title(['L_2-norm scaling, b ='...
, num2str(mm), ' bits, Error signal']);
end
ylabel('Amplitude');xlabel('n in samples');
nn1=0:nn-1;
spectr = abs(fft(out1-out2,nn));
figure(9)
plot(nn1(1:nn/2)/(nn/2),spectr(1:nn/2));
if it==0 title(['No scaling, b ='...
, num2str(mm), ' bits, FFT of the error signal']);
end
if it==1 title(['L_infty-norm scaling, b ='...
, num2str(mm), ' bits, FFT of the error signal']);
end
if it==2 title(['Worst-case scaling, b ='...
, num2str(mm), ' bits, FFT of the error signal']);
end
if it==3 title(['L_2-norm scaling, b ='...
, num2str(mm), ' bits, FFT of the error signal']);
```

```
end
xlabel('Angular frequency omega/pi');ylabel('Amplitude');
aaa=cov(out1-out2)/(2^(-2*mm)/12);
bbb=mean(out1-out2)/(2^(-mm-1));
ccc=10*log10(aaa);
text(.5,.8*max(spectr(1:nn/2)),['ouput variance='...
,num2str(aaa),'*2^(-2b)/12']);
text(.5,.6*max(spectr(1:nn/2)),['ouput mean='...
,num2str(bbb),'*2^(-b-1)']);
text(.5,.4*max(spectr(1:nn/2)),['noise gain='...
,num2str(ccc),' dB']);
```

```
%Matlab-file dirgen.m; can be found in ~ts/matlab/sldsp
%This file is used as a subroutine with iirnois.m
function [out]=dirgen(in,A,B,C,n,itype,nty)
%mimics 2's complement arithmetic implementation of a section
%C(1)H(z)C(2), where H(z)=N(z)/D(z), where
%N(z)=A(1)+A(2)z^(-1)+A(3)z^(-2)+...+A(M+1)z^(-M)
%and
%D(z)=1+B(2)z^(-1)+B(3)z^(-2)+...+B(M+1)z^(-M)
%in=input signal
%out=output signal
%n=number of fractional bits
%itype=1 for direct form II and quantization after each multiplier
%itype=2 for direct form II and quantization after additions
%itype=3 for transposed direct form II and quantization after
%each multiplier
%itype=4 for transposed direct form II and quantization after
%additions
%nty=1 for truncation; nty=2 for rounding
%
m=length(A)-1;
e=0;
if nty==2 e=.5;end
x=in;
for l=1:m x(length(in)+l)=0;end
for k=1:length(in)
kk=k+m;
if itype==1
%x(n)=in(n)
%w(n)=C(1)*x(n)-B(2)*w(n-1)-B(3)*w(n-2)-...-B(m+1)w(n-m)
%y(n)=A(1)*w(n)+A(2)*w(n-1)+A(3)*w(n-2)+...+A(m+1)w(n-m)
%out(n)=C(2)*y(n)
for l=1:m w(l)=0;end
w(kk)=floor(C(1)*x(k)*2^n+e)/(2^n);
for l=1:m
w(kk)=w(kk)+floor(-B(l+1)*w(kk-l)*2^n+e)/(2^n);
end
if w(kk) < -1; w(kk)=w(kk)+2;end
if w(kk) > 1-2^(-n); w(kk)=w(kk)-2;end
if w(kk) < -1; w(kk)=w(kk)+2;end
if w(kk) > 1-2^(-n); w(kk)=w(kk)-2;end
y(k)=floor(A(1)*w(kk)*2^n+e)/(2^n);
for l=1:m
```

```
y(k)=y(k)+floor(A(l+1)*w(kk-l)*2^n+e)/(2^n);
end
if y(k) < -1; y(k)=y(k)+2;end
if y(k) > 1-2^(-n); y(k)=y(k)-2;end
if y(k) < -1; y(k)=y(k)+2;end
if y(k) > 1-2^(-n); y(k)=y(k)-2;end
out(k)=floor(C(2)*y(k)*2^n+e)/(2^n);
if abs(C(2)-1) < 0.00000001 out(k)=y(k);end
if out(k) < -1; out(k)=out(k)+2;end
if out(k) > 1-2^(-n); out(k)=out(k)-2;end
if out(k) < -1; out(k)=out(k)+2;end
if out(k) > 1-2^(-n); out(k)=out(k)-2;end
end
if itype==2
%x(n)=in(n)
%w(n)=C(1)*x(n)-B(2)*w(n-1)-B(3)*w(n-2)-...-B(m+1)w(n-m)
%y(n)=A(1)*w(n)+A(2)*w(n-1)+A(3)*w(n-2)+...+A(m+1)w(n-m)
%out(n)=C(2)*y(n)
for l=1:m w(l)=0;end
w(kk)=C(1)*x(k);
for l=1:m
w(kk)=w(kk)-B(l+1)*w(kk-l);
end
w(kk)=floor(w(kk)*2^n+e)/(2^n);
if w(kk) < -1; w(kk)=w(kk)+2;end
if w(kk) > 1-2^(-n); w(kk)=w(kk)-2;end
if w(kk) < -1; w(kk)=w(kk)+2;end
if w(kk) > 1-2^(-n); w(kk)=w(kk)-2;end
y(k)=A(1)*w(kk);
for l=1:m
y(k)=y(k)+A(l+1)*w(kk-l);
end
y(k)=floor(y(k)*2^n+e)/(2^n);
if y(k) < -1; y(k)=y(k)+2;end
if y(k) > 1-2^(-n); y(k)=y(k)-2;end
if y(k) < -1; y(k)=y(k)+2;end
if y(k) > 1-2^(-n); y(k)=y(k)-2;end
out(k)=floor(C(2)*y(k)*2^n+e)/(2^n);
if abs(C(2)-1) < 0.00000001 out(k)=y(k);end
if out(k) < -1; out(k)=out(k)+2;end
if out(k) > 1-2^(-n); out(k)=out(k)-2;end
if out(k) < -1; out(k)=out(k)+2;end
```

```
if out(k) > 1-2^(-n); out(k)=out(k)-2;end
end
if itype==3
%y(n)=A(1)*xx(n)+A(2)*xx(n-1)-B(2)*y(n-1)+
%A(3)*xx(n-2)-B(3)*y(n-3)+...+
%A(m+1)*xx(n-m)-B(m+1)*y(n-m)
%xx(n)=C(1)*x(nn)
%out(n)=C(2)*y(n)
for l=1:m y(l)=0;xx(l)=0;end
xx(kk)=floor(C(1)*x(k)*2^n+e)/(2^n);
if abs(C(1)-1)< 0.00000001 xx(kk)=x(k);end
y(kk)=floor(A(1)*xx(kk)*2^n+e)/(2^n);
for l=1:m
y(kk)=y(kk)+floor(-B(m+2-l)*y(kk-(m+1-l))*2^n+e)/(2^n);
y(kk)=y(kk)+floor(A(m+2-l)*xx(kk-(m+1-l))*2^n+e)/(2^n);
end
if y(kk) < -1; y(kk)=y(kk)+2;end
if y(kk) > 1-2^(-n); y(kk)=y(kk)-2;end
if y(kk) < -1; y(kk)=y(kk)+2;end
if y(kk) > 1-2^(-n); y(kk)=y(kk)-2;end
out(k)=floor(C(2)*y(kk)*2^n+e)/(2^n);
if abs(C(2)-1) < 0.00000001 out(k)=y(kk);end
if out(k) < -1; out(k)=out(k)+2;end
if out(k) > 1-2^(-n); out(k)=out(k)-2;end
if out(k) < -1; out(k)=out(k)+2;end
if out(k) > 1-2^(-n); out(k)=out(k)-2;end
end
if itype==4
%y(n)=A(1)*xx(n)+A(2)xx(n-1)-B(2)*y(n-1)+
%A(3)*xx(n-2)-B(3)*y(n-3)
%xx(n)=C(1)*x(nn)
%out(n)=C(2)*y(n)
for l=1:m y(l)=0;xx(l)=0;end
xx(kk)=floor(C(1)*x(k+m)*2^n+e)/(2^n);
if abs(C(1)-1)< .00000001 xx(kk)=x(k);end
y(kk)=A(1)*xx(kk);
for l=1:m
y(kk)=y(kk)-B(m+2-l)*y(kk-(m+1-l));
y(kk)=y(kk)+A(m+2-l)*xx(kk-(m+1-l));
end
y(kk)=floor(y(kk)*2^n+e)/(2^n);
if y(kk) < -1; y(kk)=y(kk)+2;end
```

```
if y(kk) > 1-2^(-n); y(kk)=y(kk)-2;end
if y(kk) < -1; y(kk)=y(kk)+2;end
if y(kk) > 1-2^(-n); y(kk)=y(kk)-2;end
out(k)=floor(C(2)*y(kk)*2^n+e)/(2^n);
if abs(C(2)-1) < 0.00000001 out(k)=y(kk);end
if out(k) < -1; out(k)=out(k)+2;end
if out(k) > 1-2^(-n); out(k)=out(k)-2;end
if out(k) < -1; out(k)=out(k)+2;end
if out(k) > 1-2^(-n); out(k)=out(k)-2;end
end
end
```