# ELG4177 - DIGITAL SIGNAL PROCESSING
## Lab7

By:Hitham Jleed

http://www.site.uottawa.ca/~hjlee103/

uOttawa

Assignment #7

# FINITE WORD EFFECTS

# FACTORS INFLUENCING FINITE PRECISION EFFECTS

1. The structure used for implementation (Direct, transpose, etc.)
2. The word-length and data-type (fixed-point, floating-point, etc.) (1-s complement, 2-s complement, etc.)
3. The multiplication-type (rounding, truncation, etc.)

# FINITE PRECISION EFFECT: FIR Vs IIR FILTERS

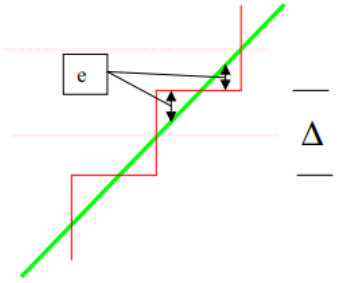
## Remarks on FIR filters:

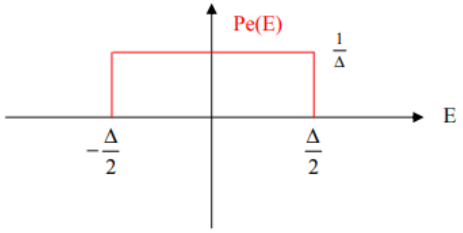
1. Filter coefficient quantization: coefficient quantization is quite serious for FIR filters due to the typically large dynamic range of coefficients of FIR filters.
2. Round-off noise: Round-off noise is not as serious for FIR filters as it is for IIR filters.
3. Limit cycles: Non-recursive (FIR) filters do not have limit cycles.
4. For FIR filters, the direct form is generally preferred.

## Remarks on IIR filters:

1. Filter coefficient quantization: coefficient quantization can make a stable IIR filter unstable! (The implementation of an IIR filters using a cascade of second order sections prevents that.)
2. Round-off noise: For a cascade of second order sections the round-off noise depends on
   (a) the poles-zero pairing,
   (b) the ordering of the sections

# Quantization Noise




PDF of Quantization Error

$$e[n] = v_q[n] - v[n]$$

$$pe = \begin{cases} 1/\Delta & -\Delta/2 < e < \Delta/2 \\ 0 & Otherwise \end{cases}$$

where v[n] is the un-quantized discrete time signal

e[n] is the quantization error

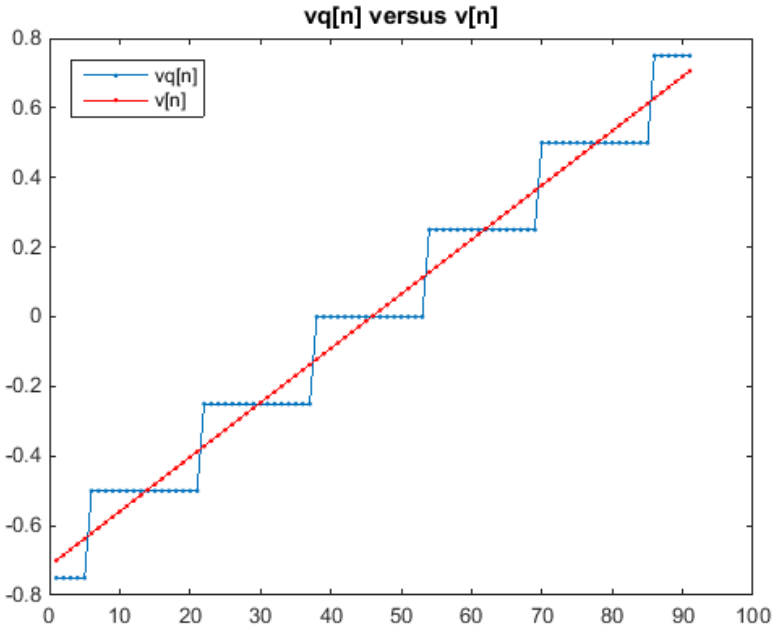
vq[n] is the quantized discrete time signal.

a) Generate an input signal v[n] from –0.7 to 0.7 using a step size of $2^{-6}$

```
v = -0.7:(2^(-6)):0.71;
```

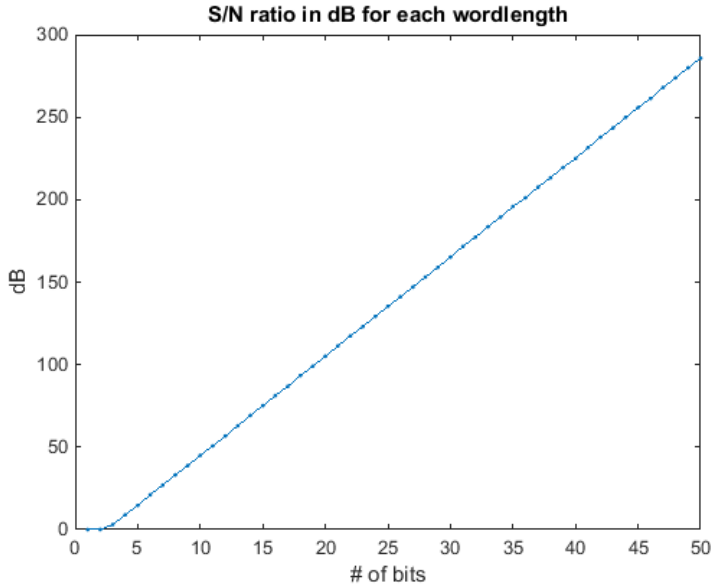
using the function *fixedpointquant.m* : (posted in brightspace)

Use a word length of 3 bits with the function, with a rounding and a saturation for the quantizer. Compute e[n]=vq[n]-v[n], and plot vq[n] versus v[n], and e[n] versus v[n]. What are the statistical properties of the error e[n] (mean, max., distribution) ?



b) Generate a Gaussion random input signal with zero mean and variance S of 0.01, using *randn* with 10000 samples.

```
S=0.01;   v = randn(1000,1)*sqrt(S);
b=1:50; % b = bit size
```

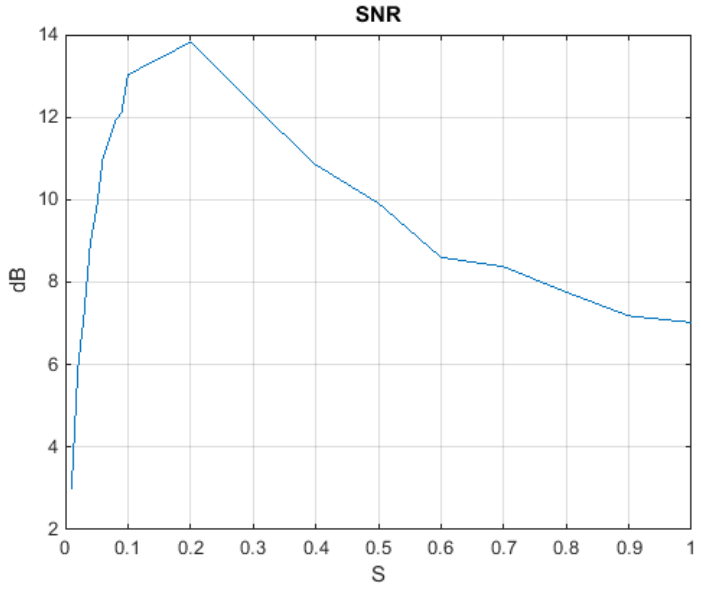




Hint: s=constant, b=vector

# c) For small values of variance S and a fixed number of bits

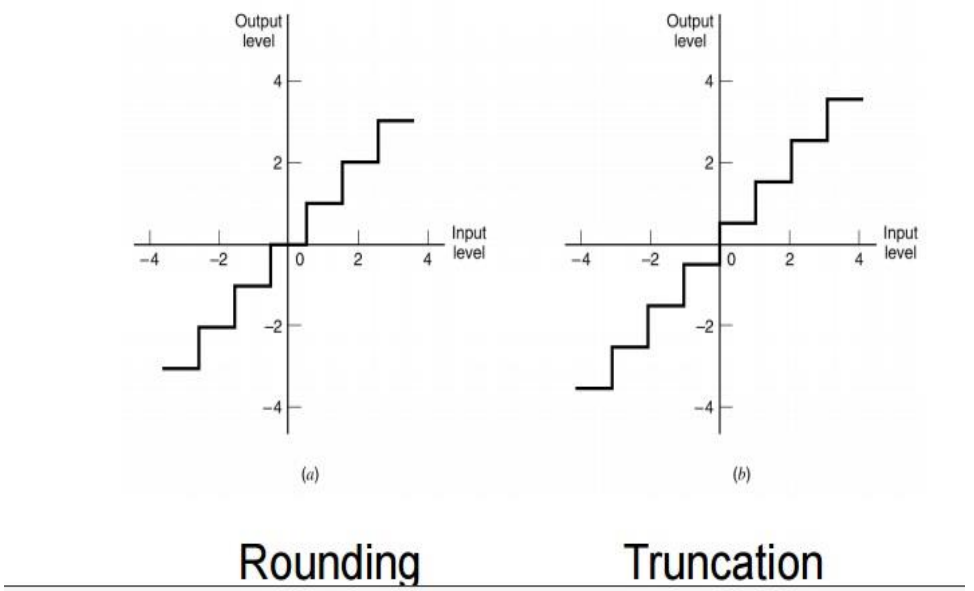
```
% S = various variances
S = [0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 0.2 0.3
0.4 0.5 0.6 0.7 0.8 0.9 1.0];
b=3; % b = bit size
```




Hint: s=vector, b=constant

# Rounding VS Truncation

d) Repeat a) by using a 2's complement truncation quantizer instead of a rounding quantizer. Compare the mean, the maximum and the distribution of the error signal e[n] with the one found in a).



## Coefficient Sensitivity

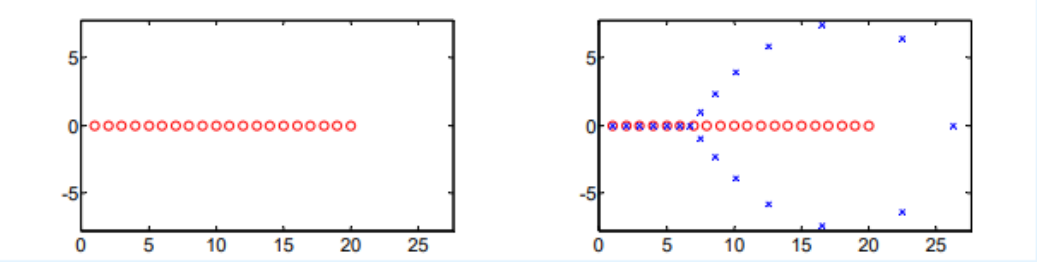
The roots of high order polynomials can be very sensitive to small changes in coefficient values.

Wilkinson's polynomial: (famous example)

$$f(x) = \prod_{n=1}^{20} (x - n) = x^{20} - 210x^{19} + 20615x^{18} - \dots$$

has roots well separated on the real axis.

Multiplying the coefficient of $x^{19}$ by $1.000001$ moves the roots a lot.




Moral: Avoid using direct form for filters orders over about 10.

# Cascaded Biquads

Avoid high order polynomials by factorizing into quadratic terms:

$$\frac{B(z)}{A(z)} = g\frac{\prod\left(1+b_{k,1}z^{-1}+b_{k,2}z^{-2}\right)}{\prod(1+a_{k,1}z^{-1}+a_{k,2}z^{-2})} = g\prod_{k=1}^{K}\frac{1+b_{k,1}z^{-1}+b_{k,2}z^{-2}}{1+a_{k,1}z^{-1}+a_{k,2}z^{-2}}$$

where $K = \max\left(\left\lceil\frac{M}{2}\right\rceil, \left\lceil\frac{N}{2}\right\rceil\right)$.

The term $\frac{1+b_{k,1}z^{-1}+b_{k,2}z^{-2}}{1+a_{k,1}z^{-1}+a_{k,2}z^{-2}}$ is a biquad (bi-quadratic section).

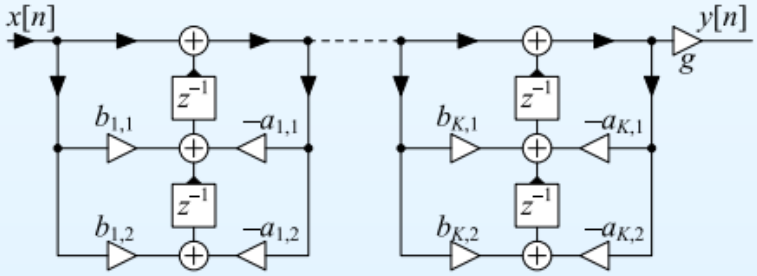
We need to choose:

(a) which poles to pair with which zeros in each biquad

(b) how to order the biquads

Direct Form II

Transposed



# MATLAB routines

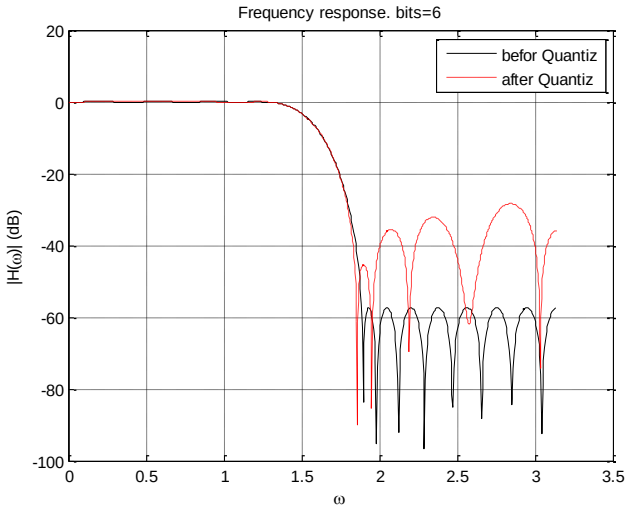
| | |
|---|---|
| residuez | $\frac{b(z^{-1})}{a(z^{-1})} \rightarrow \sum_k \frac{r_k}{1-p_k z^{-1}}$ |
| zp2tf , tf2zp | $\{z_m, p_k, g\} \leftrightarrow \prod_l \frac{b(z^{-1})}{a(z^{-1})}$ |
| tf2sos,sos2tf | $\frac{b(z^{-1})}{a(z^{-1})} \leftrightarrow \prod_l \frac{b_{0,l}+b_{1,l}z^{-1}+b_{2,l}z^{-2}}{1+a_{1,l}z^{-1}+a_{2,l}z^{-2}}$ |
| zp2sos,sos2zp | $\{z_m, p_k, g\} \leftrightarrow \prod_l \frac{b_{0,l}+b_{1,l}z^{-1}+b_{2,l}z^{-2}}{1+a_{\in 1,l}z^{-1}+a_{2,l}z^{-2}}$ |
| zp2ss,ss2zp | $\{z_m, p_k, g\} \leftrightarrow \begin{cases} x' = Ax + Bu \\ y = Cx + Du \end{cases}$ |
| tf2ss,ss2tf | $\frac{b(z^{-1})}{a(z^{-1})} \leftrightarrow \begin{cases} x' = Ax + Bu \\ y = Cx + Du \end{cases}$ |
| poly | $\text{poly}(\mathbf{A}) = \det(z\mathbf{I}-\mathbf{A})$ |

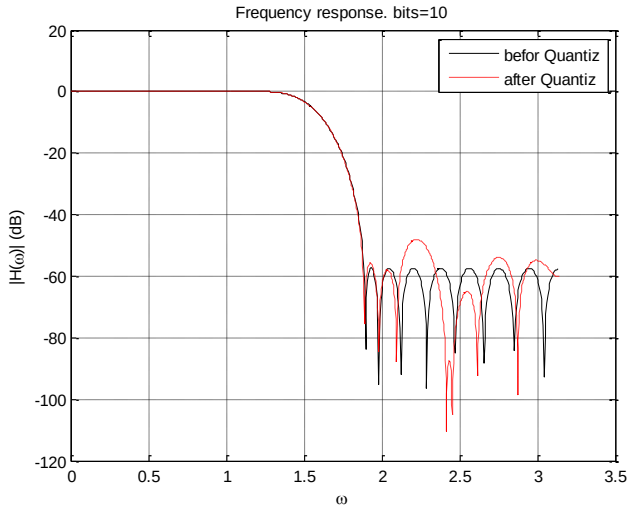
# fixed point implementations of filters

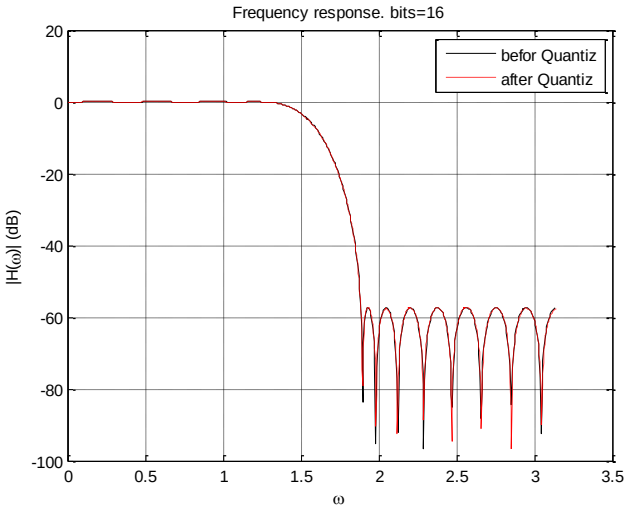
Quantize filter coefficients using quantizecoeffs.m function, uploaded in brightspace.

1. Using the FIR half-band low-pass filter. For e), f), and g)

```
% Direct form FIR implementation
```

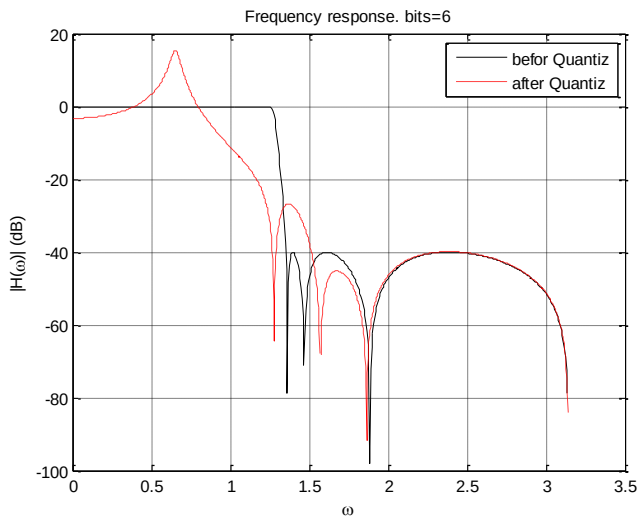


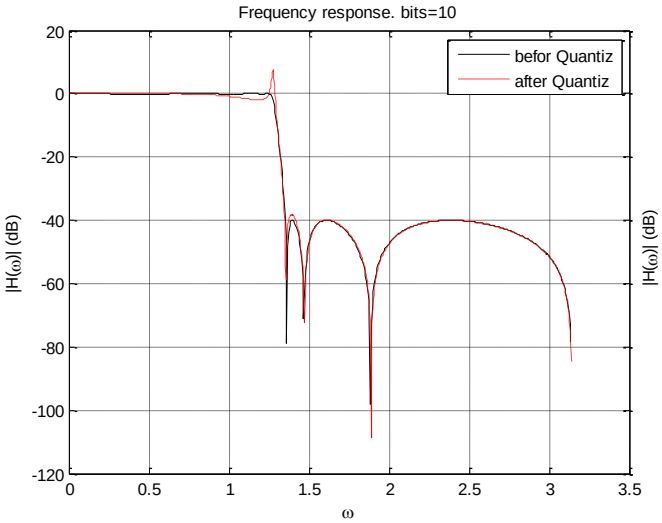



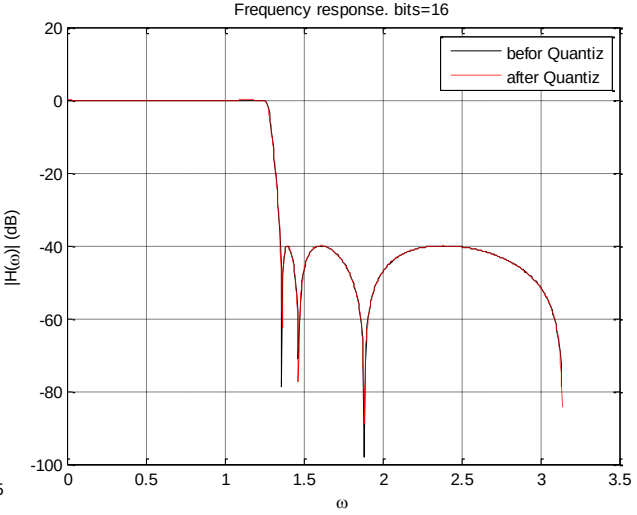



# 2. Quantize the coefficients of the IIR filter (Direct Form)

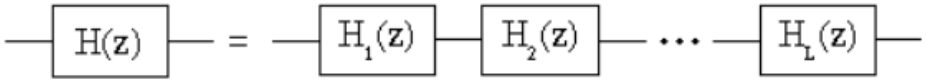
```
[b,a]=ellip(7, 0.1, 40, 0.4);                    % direct form

[num,numgain]=quantizecoeffs(b,bits);
[den,dengain]=quantizecoeffs(a,bits);
 num=num*numgain/dengain;
[h,w]=freqz(num,den,1000);
```



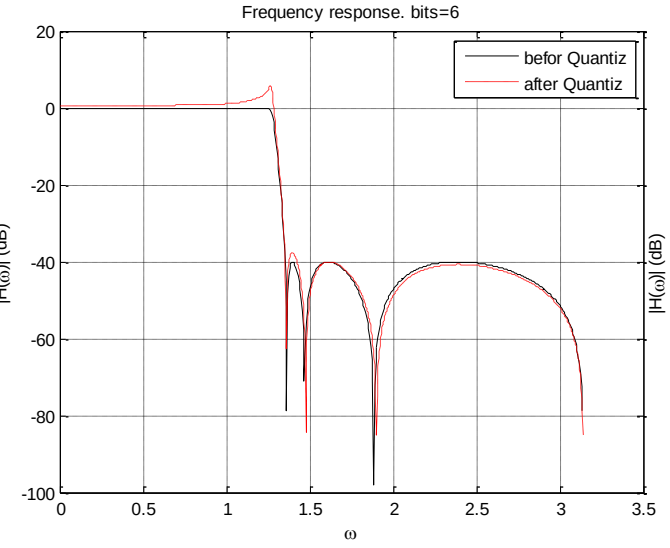



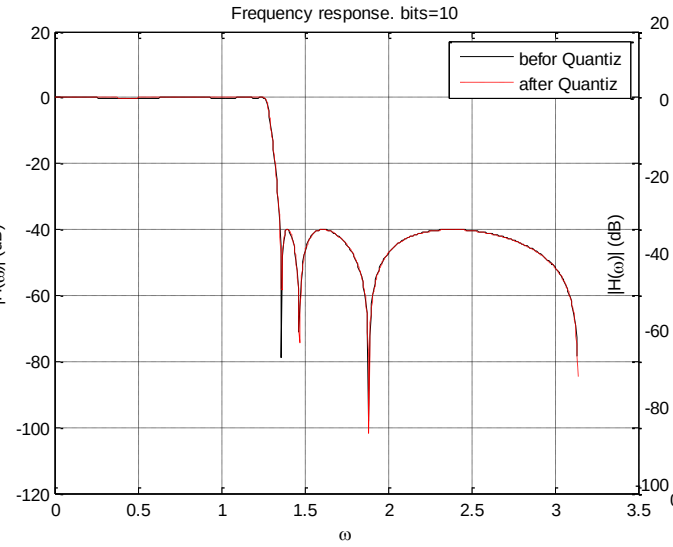
# 2. Quantize the coefficients of the IIR filter (Cascade Form)

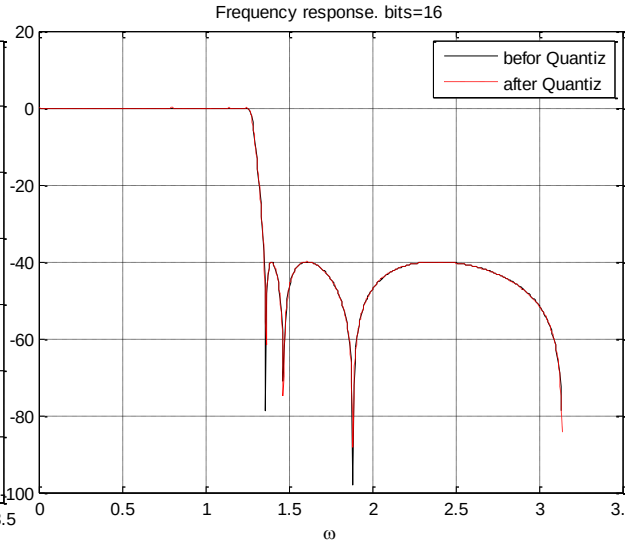



Sos= $\begin{vmatrix} b0 & b1 & b2 & a0 & a1 & a2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b0 & b1 & b2 & a0 & a1 & a2 \end{vmatrix}$

```
[b,a]=ellip(7, 0.1, 40, 0.4);   % direct form
[Z,p,k] = tf2zp(b,a);           % zero-pole form
[sos,gn]=zp2sos(Z,p,k);         % cascade form
```




```
b2=gn*multiconv(sos(:,1:3));
a2=multiconv(sos(:,4:6));
```

```
function C=multiconv(Y)
    C   =   conv(Y(1,:), Y(2,:));
    for i = 3:size(Y,1)
        C   =   conv(C, Y(i,:));
    end
end
```

uOttawa

# END Assignment7