Contents

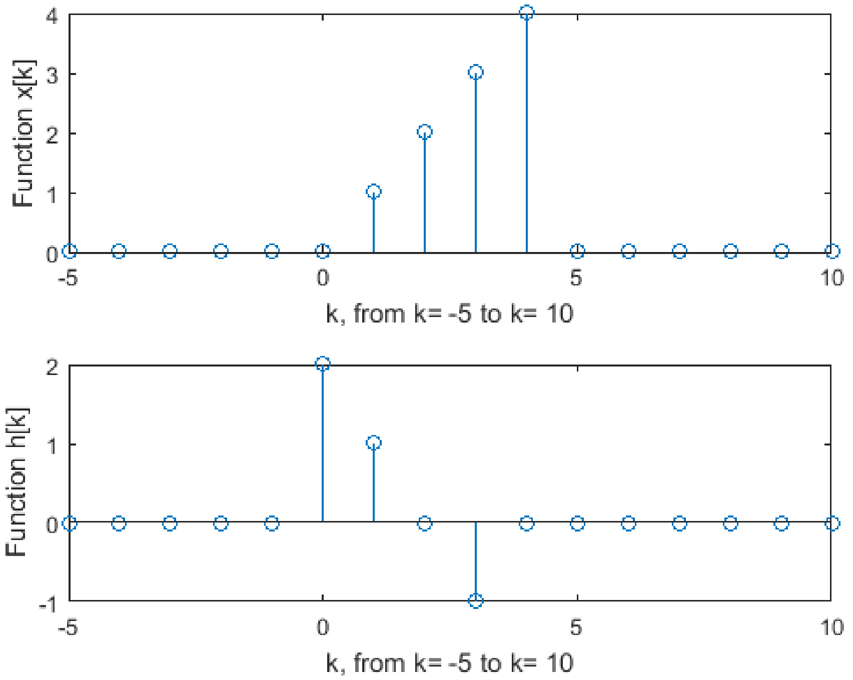
Main Function

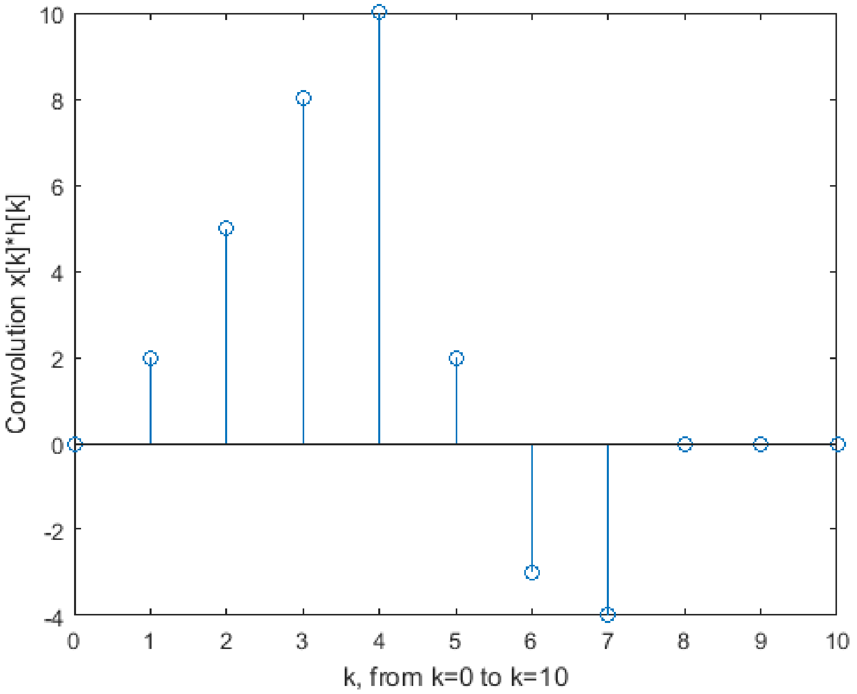
```
function ECE340_Lab2()
    warning('off','all');
    Q1_Convolution();
    Q2_Convolution_Filter();
    Q3_Aliasing_Effect();
    Q4_2D_Aliasing();
    OptionalQ_LowPassFilter_Aliasing();
end
```

Q1 Convolution

```
function Q1_Convolution()
    %Defining the discerete function x[k]
    function X=x(k)
        X=k.*(k>=0 & k<=4);
    end
    %Defining the discerete function h[k]
    function H=h(k)
        H=(2-k).*(k>=0 & k<=3);
    end

    %(1a)
    %Plotting the two functions
    figure           %Creating a figure
    subplot(2,1,1)   %Creating 1st subplot
    stem(-5:1:10,x(-5:1:10));        %plotting x[k]
    xlabel('k, from k= -5 to k= 10');  %Labelling axis
    ylabel('Function x[k]')
    hold on
    subplot(2,1,2)   %Creatting the 2nd subplot
    stem(-5:1:10,h(-5:1:10));         %plotting x[k]
    xlabel('k, from k= -5 to k= 10');   %Labelling axis
    ylabel('Function h[k]')

    %(1b)
    %Calculating and plotting the convolution
    conv_x_and_h=conv(x(0:5),h(0:5))
    %conv_x_and_h is [0 2 5 8 10 2 -3 -4 0 0 0]

    figure                   %Creating a new figure
    stem(0:10,conv_x_and_h)  %Plotting the convolution
    xlabel('k, from k=0 to k=10');   %Labelling axis
    ylabel('Convolution x[k]*h[k]')
    %(1c) See attachment
end
```

```
conv_x_and_h =

     0     2     5     8    10     2    -3    -4     0     0     0
```

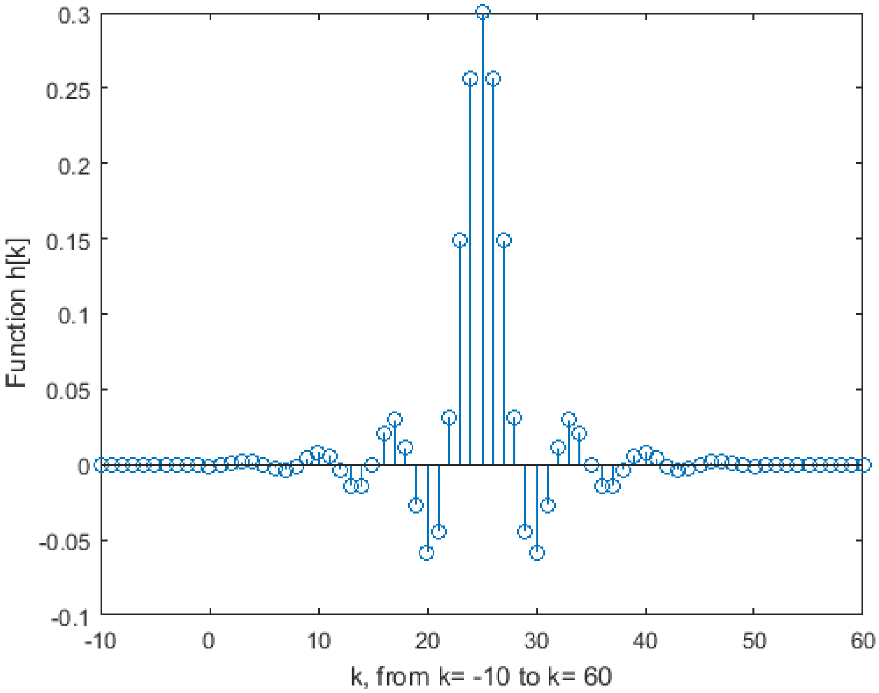




Q2 Convolution Audio Filter

```
function Q2_Convolution_Filter()
    %2a) Defining the discerete function h[k]
    function H=h(k)
        H=(0.3.*sinc(0.3.*(k-25)).* ...
       (0.54-0.46.*cos(2.*pi.*k./50))).*(k>=0 & k<=50);
    end

    %2b) Plotting the function h[k]
    figure           %Creating a new figure
    stem(-10:60,h(-10:60));        %plotting h[k]
    xlabel('k, from k= -10 to k= 60');  %Labelling axis
    ylabel('Function h[k]')

    %2c) reading the audio file, "baila.wav"
    [baila, baila_FS]=audioread('baila.wav');
    %2c+d) Calculating and saving the filtered file, "baila_filtered.wav"
    baila_filtered=conv(baila,h(0:50));
    audiowrite('baila_filtered.wav',baila_filtered,baila_FS);
    %Comments on Quality
    Quality_comment=sprintf(['The filtered audio file is ''sharper'' with less '...
       'noise. baila.wav sounds as if it was recorded in a bathroom \n'...
       'while baila_filtered.wav is crispy sharp.'])
end
```

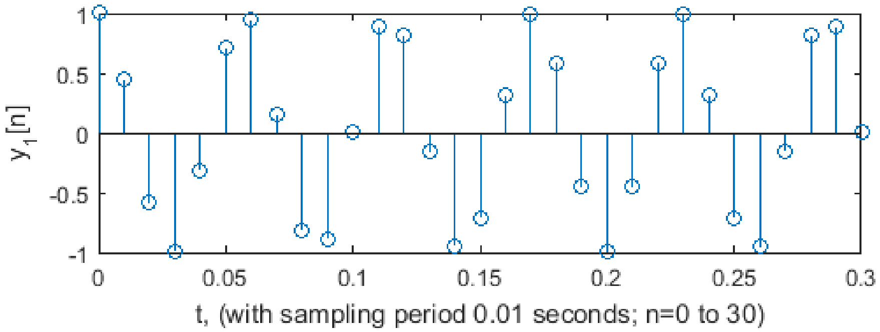
```
Quality_comment =

The filtered audio file is 'sharper' with less noise. baila.wav sounds as if it was recorded in a bathroom
while baila_filtered.wav is crispy sharp.
```

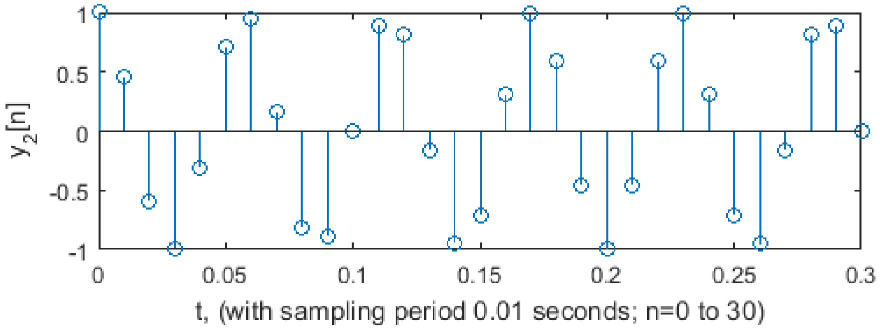


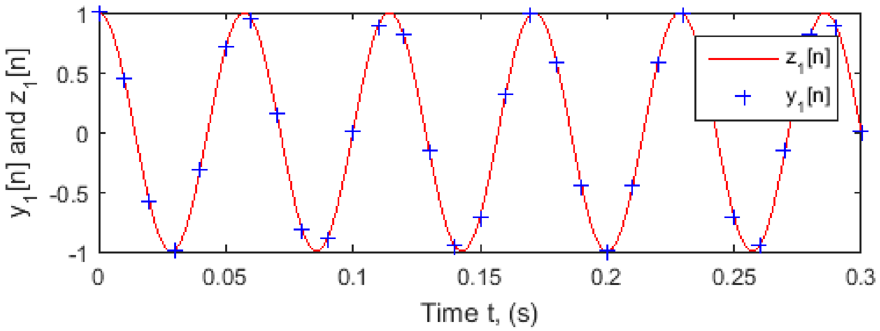
Q3 Aliasing Effect

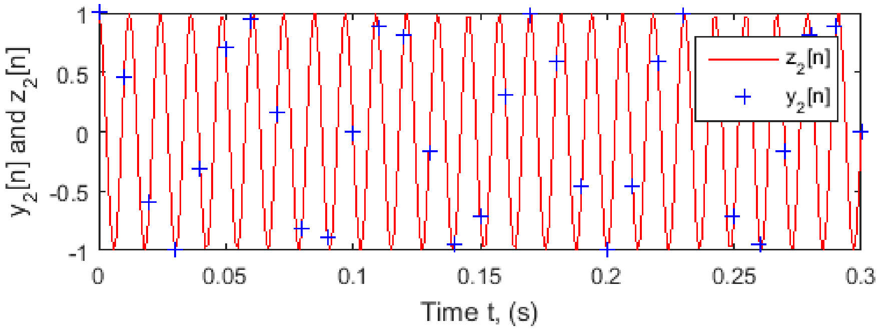
```
function Q3_Aliasing_Effect()
    function X1=x1(t)
        X1=cos(35.*pi.*t);
    end
    function X2=x2(t)
        X2=cos(165.*pi.*t);
    end
    function X3=x3(t)
        X3=cos(235.*pi.*t);
    end
    %(3a): y1 & y2
    %Defining Sampling Frequenct and thus sampling period
    f=100;
    t=1/f;

    %Calculating the digital signals y1[n] and y2[n] and stem-plotting them
    y1=x1((0:30)*t);
    y2=x2((0:30)*t);
    figure          %Creatting new figure
    subplot(2,1,1); %Creating first subplot for y1[n]
    stem((0:30)*t,y1);
    xlabel('t, (with sampling period 0.01 seconds; n=0 to 30)'); %Labels
    ylabel('y_1[n]');
    subplot(2,1,2); %Creating second subplot for y2[n]
    stem((0:30)*t,y2);
    xlabel('t, (with sampling period 0.01 seconds; n=0 to 30)'); %Labels
    ylabel('y_2[n]');
        Comment=sprintf(['The two sequences y_1[n] and y_2[n] have the same values but'...
        ' the two functions x1 and x2 are not the same, \nthey only appear'...
        ' to be the same due to alising and choice of sampling frequency.'])


    %(3b): z1 & z2
    %Defining Sampling Frequenct and thus sampling period
    f2=1000;
    t2=1/f2;
    %Calculating the digital signals y1[n] and y2[n] and stem-plotting them
    z1=x1((0:300)*t2);
    z2=x2((0:300)*t2);
    figure          %Creatting new figure
    subplot(2,1,1);
    plot((0:300)*t2,z1,'r-', (0:30)*t,y1,'b+');
    xlabel('Time t, (s)'); ylabel('y_1[n] and z_1[n]');
    legend('z_1[n]','y_1[n]');
    subplot(2,1,2);
    plot((0:300)*t2,z2,'r-', (0:30)*t,y2,'b+');
    xlabel('Time t, (s)'); ylabel('y_2[n] and z_2[n]');
    legend('z_2[n]','y_2[n]');
        Comment=sprintf(['At high enough frequency the differences in x1 and x2 '...
    'becomes clear. Eventhough y1 and y2 have the same values but z1 and\n'...
    ' z2 do not have the same values. So at high enough sampling rate'...
    ' the aliasing effect disappears.'])

    %(3c): z3
    y3=x3((0:50)*t);
    y1=x1((0:50)*t);
    figure          %Creatting new figure
    subplot(2,1,1); %Creating first subplot for y1[n] and y3[n]
    stem([(0:50)*t;(0:50)*t]',[y1' y3']);
    xlabel('t, (with sampling period 0.01 seconds; n=0 to 50)'); %Labels
    ylabel('y_1[n] & y_3[n]');
    legend('y_1[n]','y_3[n]');
    subplot(2,1,2); %Creating second subplot for difference y1[n]-y3[n]
    stem((0:50)*t,[y1-y3]);
    xlabel('t, (with sampling period 0.01 seconds; n=0 to 50)'); %Labels
    ylabel('(y_1[n] - y_3[n])');
    title('Negligible Difference in the two Sequences');
end
```

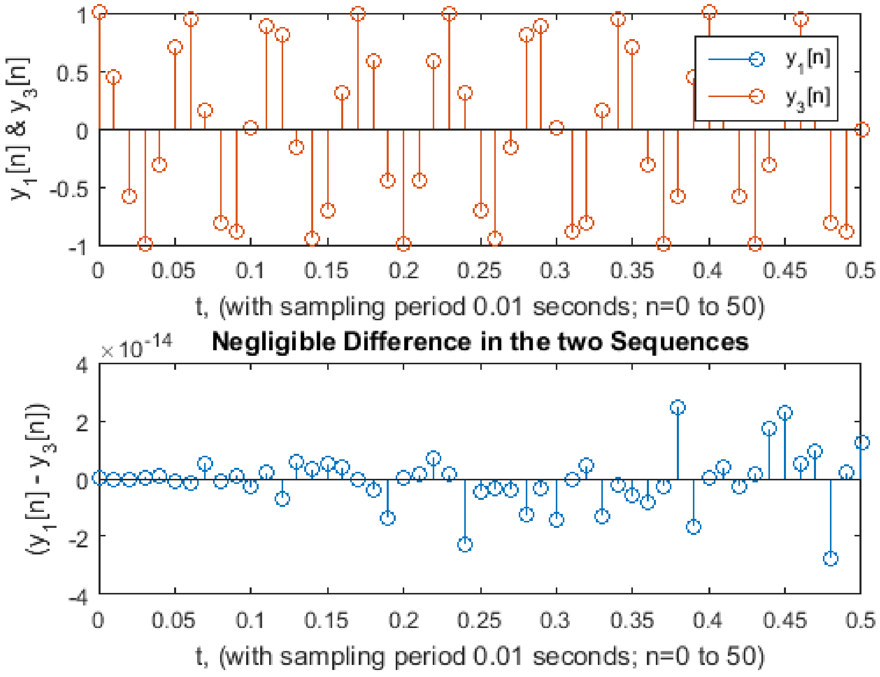
```
Comment =

The two sequences y_1[n] and y_2[n] have the same values but the two functions x1 and x2 are not the same,
they only appear to be the same due to alising and choice of sampling frequency.


Comment =

At high enough frequency the differences in x1 and x2 becomes clear. Eventhough y1 and y2 have the same values but z1 and
 z2 do not have the same values. So at high enough sampling rate the aliasing effect disappears.
```


$y_1[n]$
1
0.5
0
-0.5
-1
0
0.05
0.1
0.15
0.2
0.25
0.3
t, (with sampling period 0.01 seconds; n=0 to 30)


$y_2[n]$
1
0.5
0
-0.5
-1
0
0.05
0.1
0.15
0.2
0.25
0.3
t, (with sampling period 0.01 seconds; n=0 to 30)


$y_1[n]$ and $z_1[n]$
$z_1[n]$
$y_1[n]$
1
0.5
0
-0.5
-1
0
0.05
0.1
0.15
0.2
0.25
0.3
Time t, (s)


$y_2[n]$ and $z_2[n]$
$z_2[n]$
$y_2[n]$
1
0.5
0
-0.5
-1
0
0.05
0.1
0.15
0.2
0.25
0.3
Time t, (s)

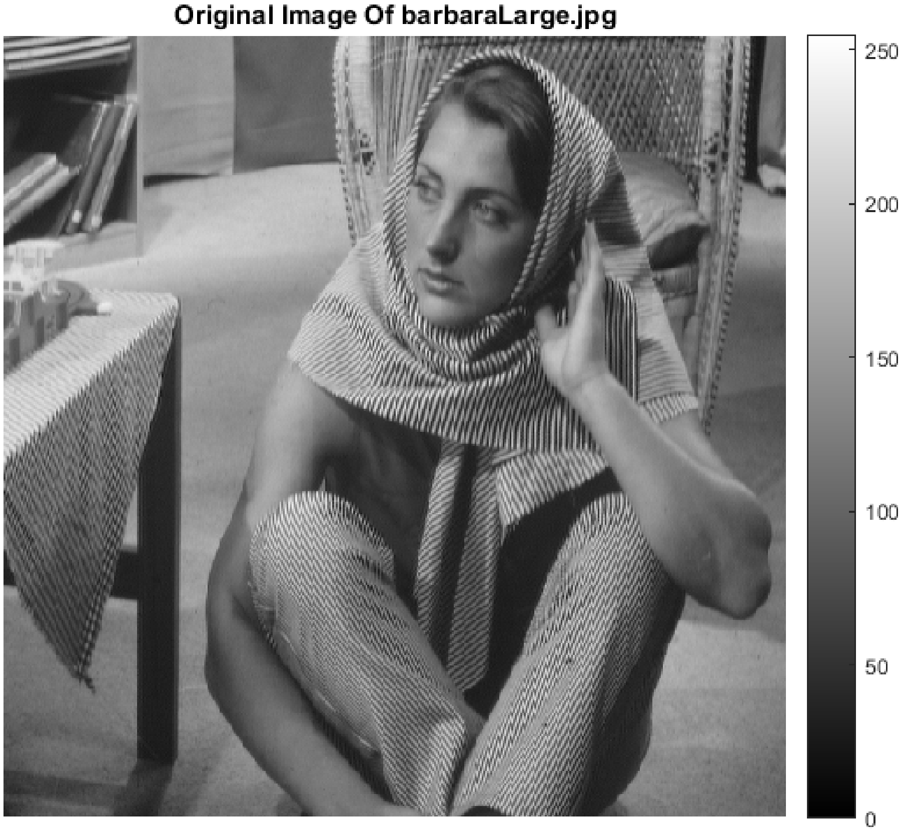


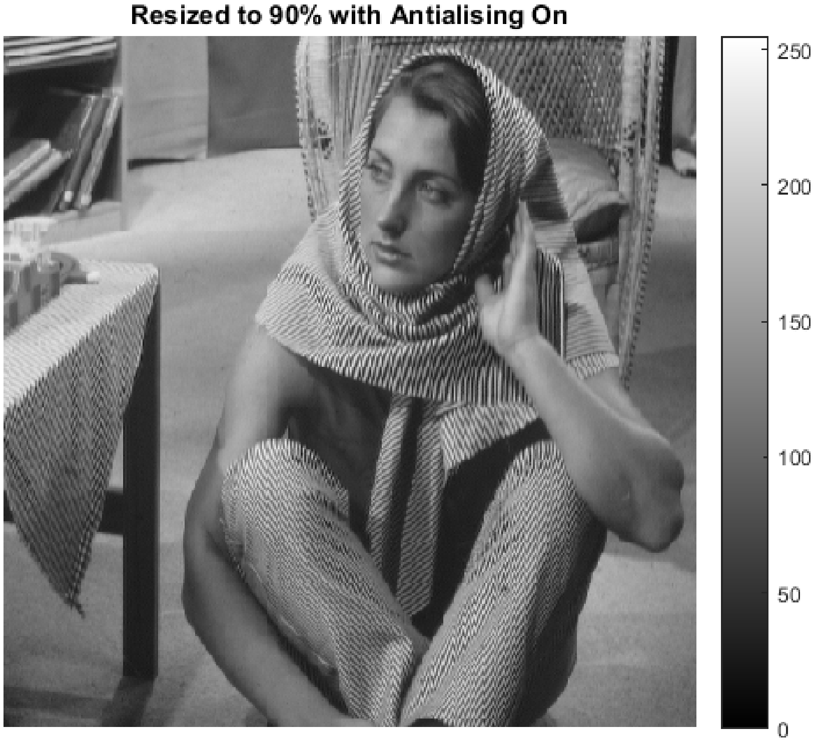
Q4 2D Aliasing in an image

```
function Q4_2D_Aliasing()
    %a) Reading the image file
    Img=imread('barbaraLarge.jpg');
    %b) Displaying the original image
    figure; imshow(Img), colorbar;
    title('Original Image Of barbaraLarge.jpg');
    %c) Resizing the image files with and without antializing enabled
    Img0_9_AntiAlisOn=imresize(Img, 0.9, 'nearest','antialiasing',1);
    Img0_7_AntiAlisOn=imresize(Img, 0.7, 'nearest','antialiasing',1);
    Img0_5_AntiAlisOn=imresize(Img, 0.5, 'nearest','antialiasing',1);
    Img0_9_AntiAlisOff=imresize(Img, 0.9, 'nearest','antialiasing',0);
    Img0_7_AntiAlisOff=imresize(Img, 0.7, 'nearest','antialiasing',0);
    Img0_5_AntiAlisOff=imresize(Img, 0.5, 'nearest','antialiasing',0);
    comments=sprintf(['When antialising is enabled the image quality is not '...
        'affected as much and there is less aliasing observable in high\n'...
        'frequency regions'])


    %d) Showing the images
    figure;imshow(Img0_9_AntiAlisOn), colorbar;
    title('Resized to 90% with Antialising On');
    figure;imshow(Img0_9_AntiAlisOff), colorbar;
    title('Resized to 90% with Antialising Off');
    figure;imshow(Img0_7_AntiAlisOn), colorbar;
    title('Resized to 70% with Antialising On');
    figure;imshow(Img0_7_AntiAlisOff), colorbar;
    title('Resized to 70% with Antialising Off');
    figure;imshow(Img0_5_AntiAlisOn), colorbar;
    title('Resized to 50% with Antialising On');
    figure;imshow(Img0_5_AntiAlisOff), colorbar;
    title('Resized to 50% with Antialising Off');
    comments=sprintf(['High frequency regions begin to show aliasing when the '...
        'sampling frequency is reduced. Having antialiasing enabled \n'...
        'reduces aliasing and the image quality is better.'])
end
```

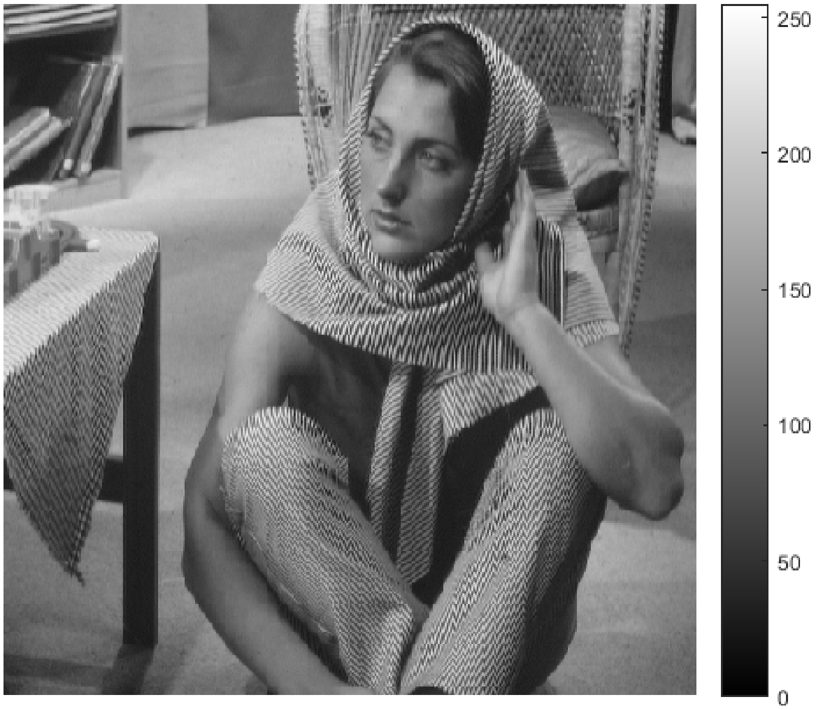
```
comments =

When antialising is enabled the image quality is not affected as much and there is less aliasing observable in high
frequency regions
```

```
comments =

High frequency regions begin to show aliasing when the sampling frequency is reduced. Having antialiasing enabled
reduces aliasing and the image quality is better.
```

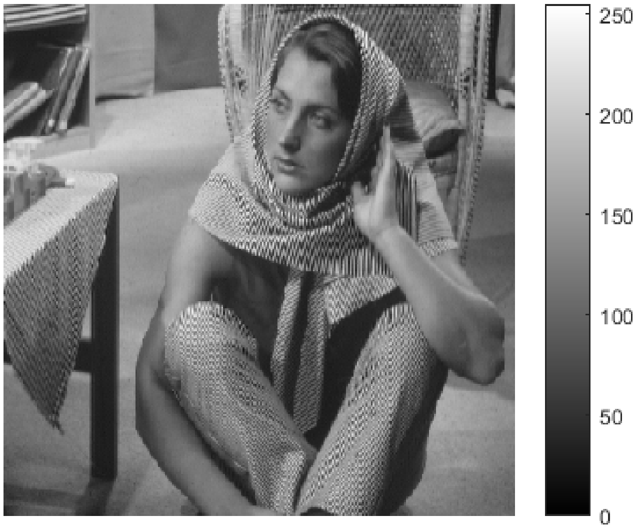




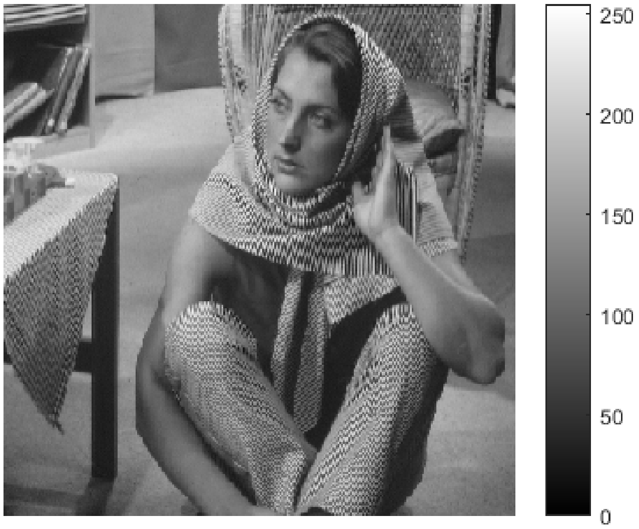
Resized to 90% with Antialising Off



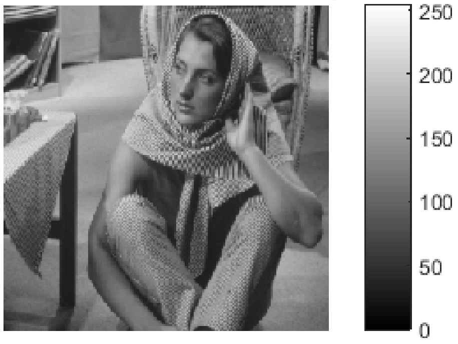
Resized to 70% with Antialising On



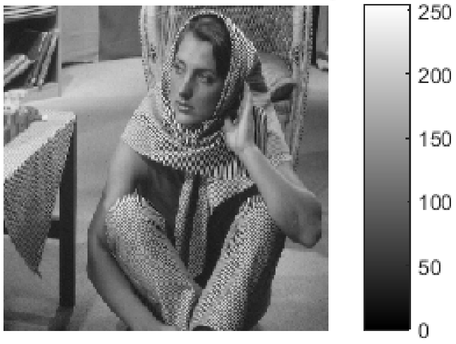
Resized to 70% with Antialising Off



Resized to 50% with Antialising On



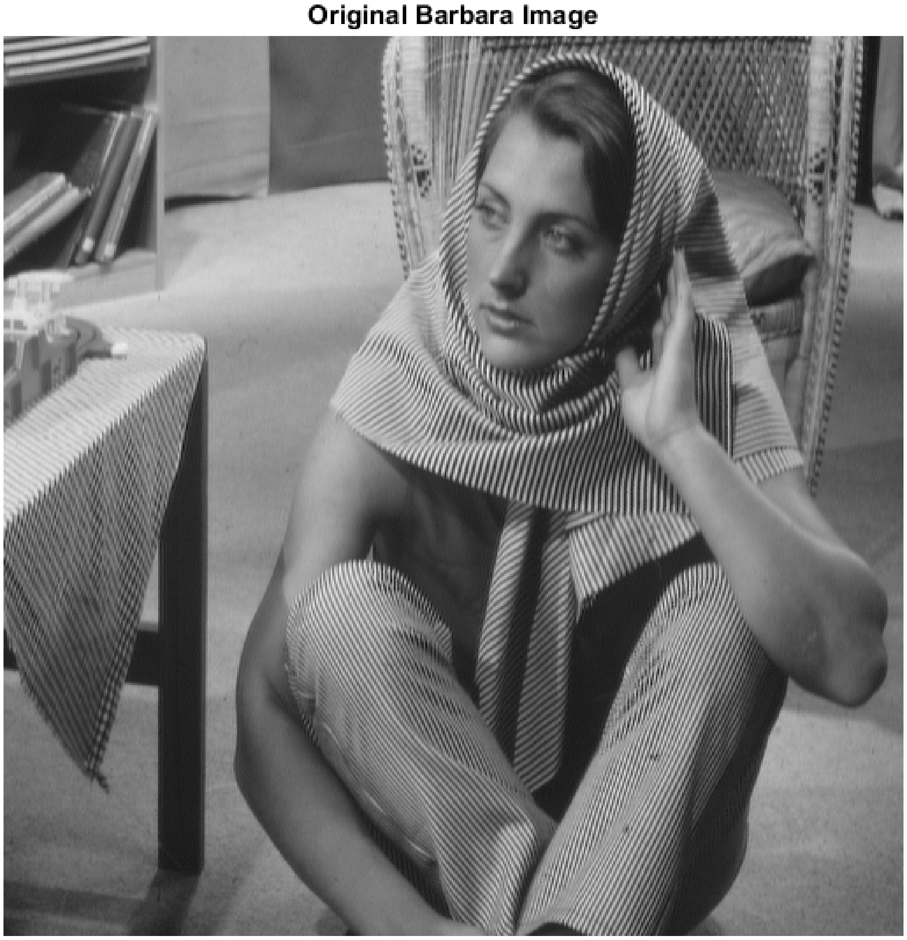
Resized to 50% with Antialising Off



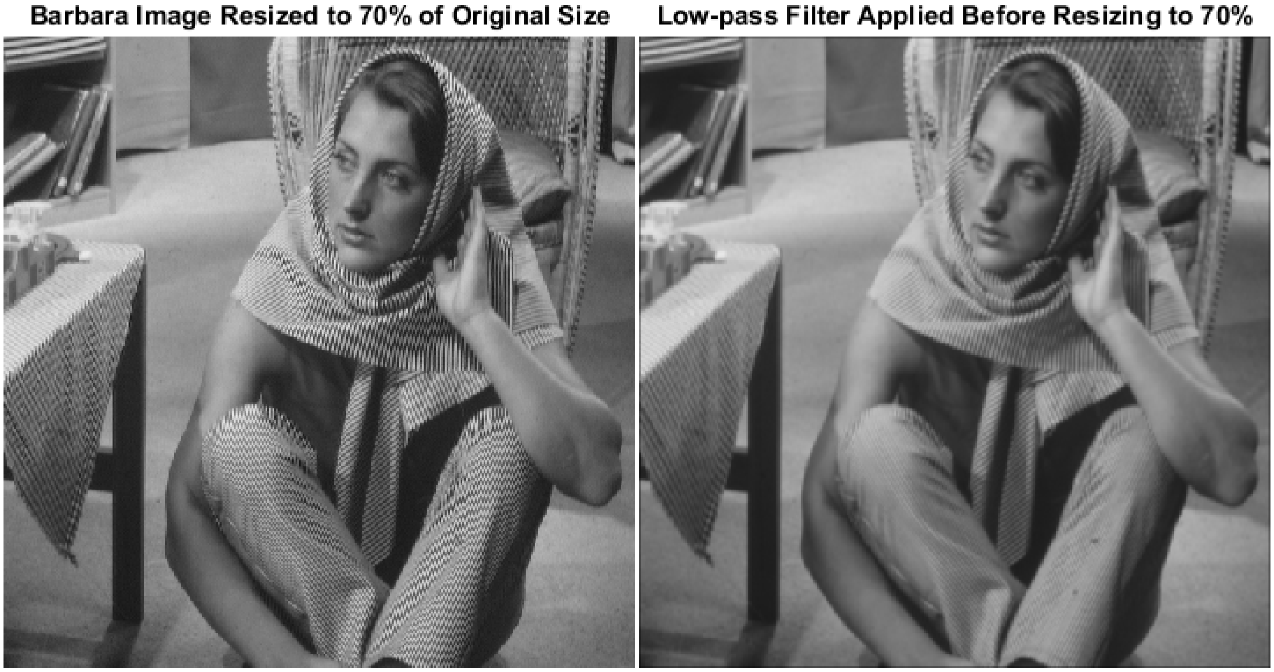
Optional Demo, Low pass filtering perior to resizing

```
function OptionalQ_LowPassFilter_Aliasing()
    I=imread('barbaraLarge.jpg');
    % Low pass filtering before downsampling
    % creates a 3x3 low pass filter kernel
    filt=fspecial('average',[3 3]);
    % applies the lpf by convolving the image with the filter kernel
    filt_img=imfilter(I,filt,'conv');

    %Resizing the image to 70% with and without low pass filter applied
    B_LPF=imresize(filt_img, 0.7 , 'nearest', 0);
    B=imresize(I, 0.7, 'nearest', 0);

    %Printing the figures
    figure,  imshow(I);
    title('Original Barbara Image');
    figure,  imshow(B);
    title('Barbara Image Resized to 70% of Original Size');
    figure,  imshow(B_LPF);
    title('Low-pass Filter Applied Before Resizing to 70%');
    comment=sprintf(['The filter is a 3by3 squar that averages the nearby values '...
        'and thus removes the high frequency components that have periods\n'...
        ' of smaller than 3 pixels. This lp filtering reduces the aliasing'...
        ' effect and improves the image quality during resizing.'])
end
```

```
comment =

The filter is a 3by3 squar that averages the nearby values and thus removes the high frequency components that have periods
 of smaller than 3 pixels. This lp filtering reduces the aliasing effect and improves the image quality during resizing.
```



Original Barbara Image



Barbara Image Resized to 70% of Original Size

Low-pass Filter Applied Before Resizing to 70%

Published with MATLAB® R2015a