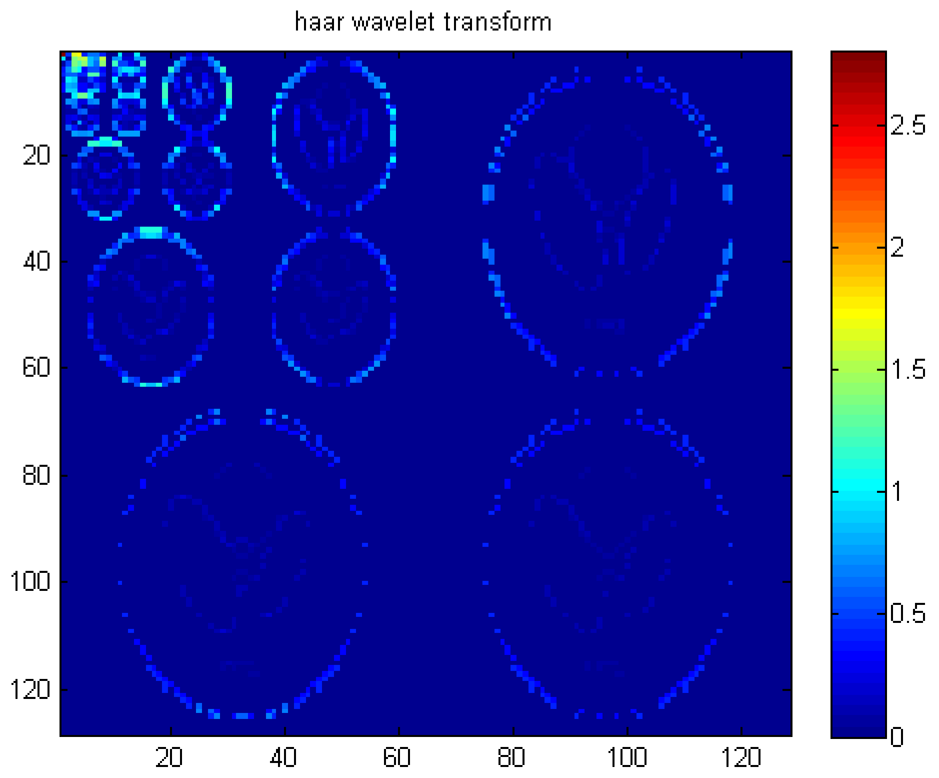
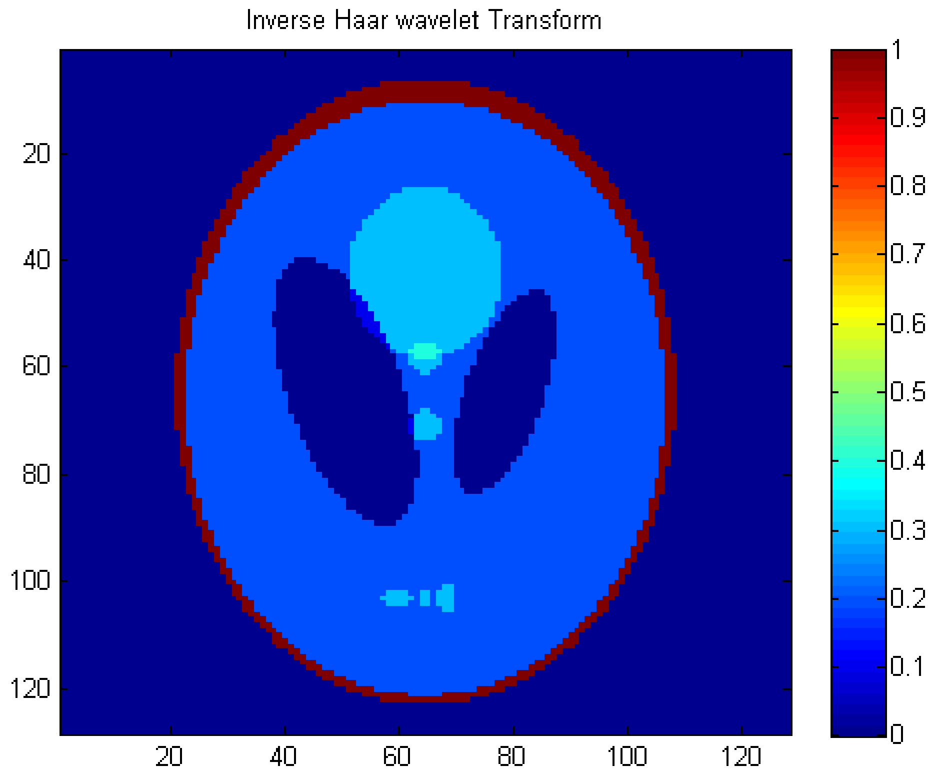
```
function [  ] = q1( )
    %finding Wavelet Transform
    s_image=128;
    f = double(phantom(s_image));
    w=haar_LLevel(f,log2(s_image));
    figure;imagesc(log(1+abs(w)));colorbar;title('haar wavelet transform');
    f_out=invhaar_LLevel(w,log2(s_image));
    figure;imagesc(f_out);colorbar;title('Inverse Haar wavelet Transform');

end

function[w]=haar_LLevel(f,steps)
    s1=size(f,1);
    w=haar_oneLevel(f);
    for k=1:steps-1
        w(1:s1/2,1:s1/2)=haar_oneLevel(w(1:s1/2,1:s1/2));
        s1=s1/2;
    end

end

function w = haar_oneLevel(x)
    [M,N] = size(x);
    if M~=N
       error('image must be square');
    end

    if 2^round(log2(M))~=M
       error('sidelength must be power of two');
    end

    h00 = [1 1; 1 1]/2;
    h01 = [-1 1; -1 1]/2;
    h10 = [-1 -1; 1 1]/2;
    h11 = [1 -1; -1 1]/2;

    w00 = conv2(x,h00,'same');
    w00 = w00(1:2:end,1:2:end);

    w01 = conv2(x,h01,'same');
    w01 = w01(1:2:end,1:2:end);

    w10 = conv2(x,h10,'same');
    w10 = w10(1:2:end,1:2:end);

    w11 = conv2(x,h11,'same');
    w11 = w11(1:2:end,1:2:end);

    w = [w00 w01; w10 w11];

end

function [x] = invhaar_oneLevel(w)

[M,N] = size(w);
if M~=N
   error('image must be square');
```

```
end

if 2^round(log2(M))~=M
   error('sidelength must be power of two');
end

wup = kron(w,[0 0; 0 1]);

h00 = [1 1; 1 1]/2;
h01 = [1 -1; 1 -1]/2;
h10 = [1 1; -1 -1]/2;
h11 = [1 -1; -1 1]/2;


w00 = wup(1:M,1:M);
x00 = conv2(w00,h00,'same');

w01 = wup(1:M,((1:M)+M));
x01 = conv2(w01,h01,'same');

w10 = wup(((1:M)+M),1:M);
x10 = conv2(w10,h10,'same');

w11 = wup(((1:M)+M),((1:M)+M));
x11 = conv2(w11,h11,'same');

x = (x00+x01+x10+x11);
end

function[f]=invhaar_LLevel(w,num_steps)
    f=w;
    [s1,s2]=size(w);
    for i=num_steps:-1:1
        s_temp=power(2,i-1);
        %display(s_temp);
        f(1:s1/s_temp,1:s2/s_temp)=invhaar_oneLevel(f(1:s1/s_temp,1:s2/s_temp));
    end

end
```

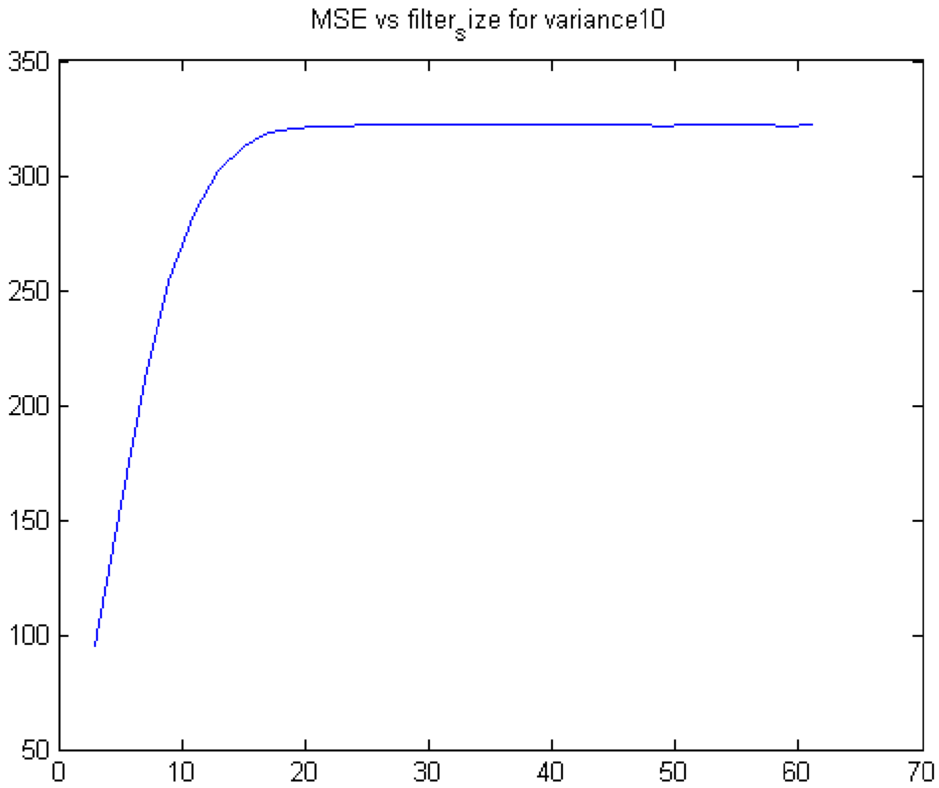




Published with MATLAB® R2013a

Solution 2 A

```
function[]=q2a()
        variance=10;
%     Comment - optimum neighborhood for removing gaussian noise is 3*3 size, which results
in minimum MSE
%     Thus we choose a 3*3 gaussian filter
    %variance=10;
    for i=1:30
       filter_size(i)=2*i+1;
        MSE(i)=q2_sub(filter_size(i),variance);
        x(i)=i;
    end
    figure;plot(filter_size,MSE);title(['MSE vs filter_size for variance'
num2str(variance)]);
end

function[Avg_MSE]=q2_sub(filter_size,variance)
    D0=filter_size;
    f=double(imread('CircleSquare.tif'));
    MSE(1:10)=0;x(1:10)=0;%initialisation
    fmax = max(f(:));
    I = 100;
    a = 1.1;
    f = f/fmax*I;
    fmin = 0;
    fmax = max(f(:));
    orignal_image=f;
    for i=1:10
       % generating noise
        snr = I^2/variance; % = I^2/sigma^2

        sigma = sqrt(I^2/snr);
        gaussian_noise = randn(size(f))*sigma;
        noisy_image=f+gaussian_noise;
       [filtered_image,MSE(i)]=gaussian_filter(noisy_image,orignal_image,D0,sigma);
%        figure;imagesc(f);
%        figure;imagesc(noisy_image);
%        figure;imagesc(filtered_image);title(['MSE=' num2str(MSE)]);
%        pause(20);
       x(i)=i;
    end
    Avg_MSE=sum(MSE(:))/size(MSE,1);
%     fpritnf('Average Error =%f',Avg_MSE);
%     figure;plot(x,MSE);
end

function[imout,MSE]=gaussian_filter(noisy_image,orignal_image,D0,sigma)

    f=double(noisy_image);
    h=fspecial('gaussian',D0,double(sigma));
    imout=conv2(f,h,'same');
    diff=orignal_image-imout;
    MSE=sum(diff(:).*diff(:) )/(size(noisy_image,1)*size(noisy_image,2));
end
```



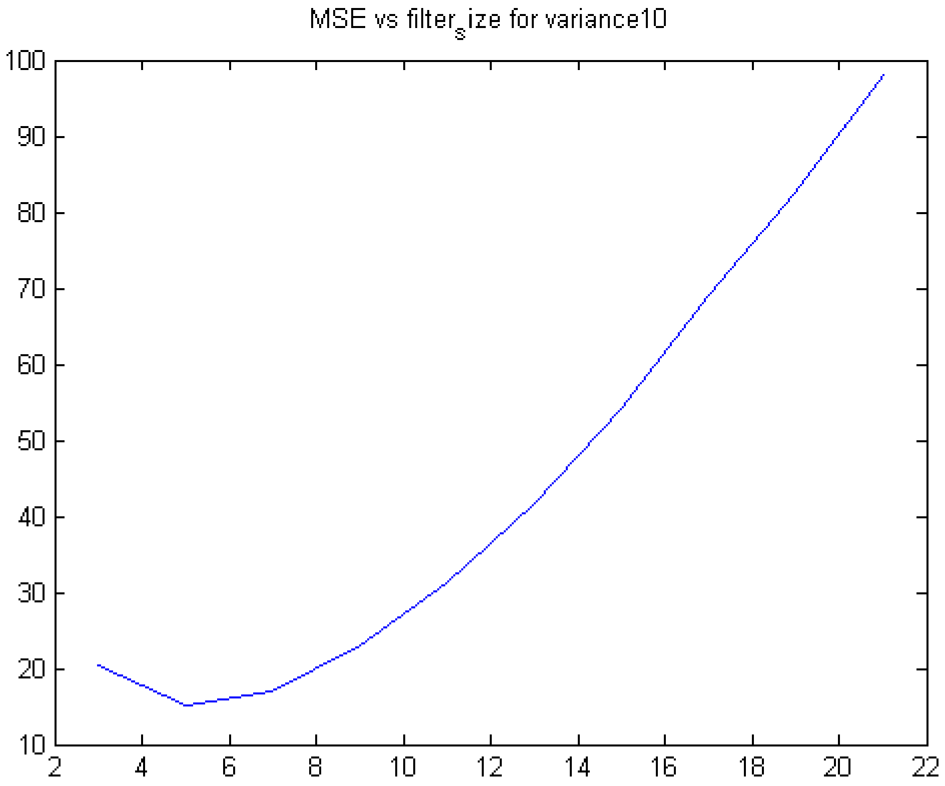
Published with MATLAB® R2013a

Comment – The MSE increases as the filter size of Gaussian Filter Increases and saturates after reaching size of 15.

Question 2B

```
function[]=q2b()
    %Median Filter
    variance=10;
    for i=1:10
       filter_size(i)=2*i+1;
        MSE(i)=q2b_sub(filter_size(i),variance);
        x(i)=i;
            end
    figure;plot(filter_size,MSE);title(['MSE vs filter_size for variance'
num2str(variance)]);
end

function[Avg_MSE]=q2b_sub(filter_size,variance)
    D0=filter_size;
    f=double(imread('CircleSquare.tif'));
    MSE(1:10)=0;x(1:10)=0;%initialisation
    fmax = max(f(:));
    I = 100;
    a = 1.1;
    f = f/fmax*I;
    fmin = 0;
    fmax = max(f(:));
    orignal_image=f;
     for i=1:10
        snr = I^2/variance; % = I^2/sigma^2
        sigma = sqrt(I^2/snr);
        gaussian_noise = randn(size(f))*sigma;
        noisy_image=f+gaussian_noise;

[filtered_image,MSE(i)]=median_filter(noisy_image,orignal_image,D0,sigma);
%        figure;imagesc(f);
%        figure;imagesc(noisy_image);
%        figure;imagesc(filtered_image);title(['MSE=' num2str(MSE(i))]);
%        pause(20);
       x(i)=i;
     end
    Avg_MSE=sum(MSE(:))/size(MSE,1);
end

function[f2,MSE]=median_filter(noisy_image,orignal_image,D0,sigma)
    neighborhood_size=D0;
    f=double(noisy_image);


    [s1,s2]=size(orignal_image);

f2(1:s1+2*floor(neighborhood_size/2),1:s2+2*floor(neighborhood_size/2))=0;

g=padarray(noisy_image,[floor(neighborhood_size/2),floor(neighborhood_size
/2)],'symmetric');

    for m=1+floor(neighborhood_size/2):s1+floor(neighborhood_size/2)
        for n= 1+floor(neighborhood_size/2):s2+floor(neighborhood_size/2)
             sub=g(m-
floor(neighborhood_size/2):m+floor(neighborhood_size/2),n-
floor(neighborhood_size/2):n+floor(neighborhood_size/2));
             f2(m,n)=median(sub(:));
        end
    end



f2=f2(1+floor(neighborhood_size/2):s1+floor(neighborhood_size/2),1+floor(n
eighborhood_size/2):s2+floor(neighborhood_size/2));
```

```
    diff=orignal_image-f2;
    MSE=sum(diff(:).*diff(:))/(s1*s2);
%     figure;imagesc(orignal_image);
%     figure;imagesc(noisy_image);
%     figure;imagesc(f2);
%     pause(20);
%
end
```



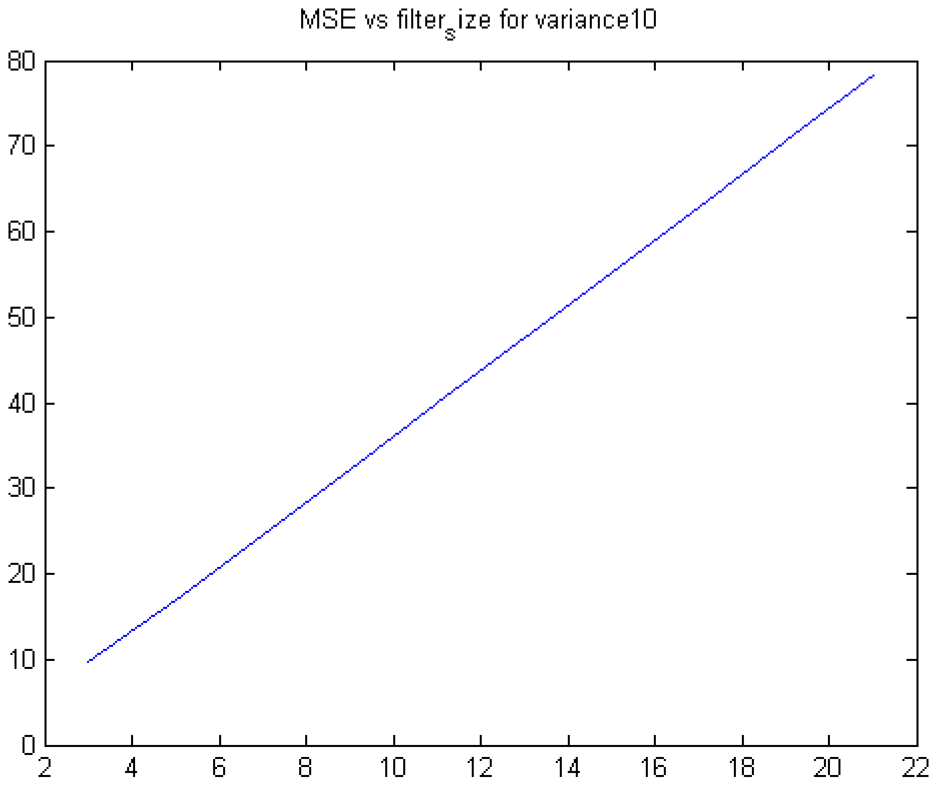
Published with MATLAB® R2013a

Comment – The MSE decreases as size approaches value of 5 and then increases again.

Question 2 C

```
function[]=q2c()
    variance=10;
    for i=1:10
       filter_size(i)=2*i+1;
        MSE(i)=MSE_bilateral_fn(variance,filter_size(i));
        x(i)=i;
        display(i);
    end
    figure;plot(filter_size,MSE);title(['MSE vs filter_size for variance'
num2str(variance)]);
end

function[MSE_avg]=MSE_bilateral_fn(variance, neighborhood_size)
%     tic;
     image=double(imread('CircleSquare.tif'));
    intensity_cutoff=100;
    num_iterations=10;MSE_avg=0;
    [s1,s2]=size(image);
    f=double(image);
    MSE(1:10)=0;x(1:10)=0;%initialisation
    fmax = max(f(:));
    I = 100;
    a = 1.1;
    f = f/fmax*I;
    MSE(1:num_iterations)=0;
    %neighborhood_size=3;
    f2(1:s1+2*floor(neighborhood_size/2),1:s2+2*floor(neighborhood_size/2))=0;

    for k=1:num_iterations
%         fprintf('slice number=%d \n',k);
        snr = I^2/variance; % = I^2/sigma^2
        sigma = sqrt(I^2/snr);
        gaussian_noise = randn(size(f))*sigma;
        noisy_image=f+gaussian_noise;
        g=noisy_image;

g=padarray(g,[floor(neighborhood_size/2),floor(neighborhood_size/2)],'symmetric');


        for i=1+floor(neighborhood_size/2):s1+floor(neighborhood_size/2)
            for j=1+floor(neighborhood_size/2):s2+floor(neighborhood_size/2)
                sub_g=g(i-
floor(neighborhood_size/2):i+floor(neighborhood_size/2),j-
floor(neighborhood_size/2):j+floor(neighborhood_size/2));
                count=0;%will be zero for the central pixel anyways
                sum_of_pixels=0;
                for m=1:2*floor(neighborhood_size/2)+1
                    for n=1:2*floor(neighborhood_size/2)+1
```

```
                       w=1*(sqrt((sub_g(m,n)-g(i,j))^2)<intensity_cutoff);
                       if(w==1)
                          sum_of_pixels=sum_of_pixels+sub_g(m,n); count=count+1;
                       end
                    end
                end
                f2(i,j)=sum_of_pixels/count;
%                 display(w);pause(5);

            end
        end

f2=f2(1+floor(neighborhood_size/2):s1+floor(neighborhood_size/2),1+floor(neighborho
od_size/2):s2+floor(neighborhood_size/2));
       % display(size(f2));
        diff=f-f2;
        MSE(k)=sum(diff(:).*diff(:))/(s1*s2);
        %figure;imshow(uint8(f2));
    end
    MSE_avg=sum(MSE(:))/num_iterations;
%     toc;
end
```



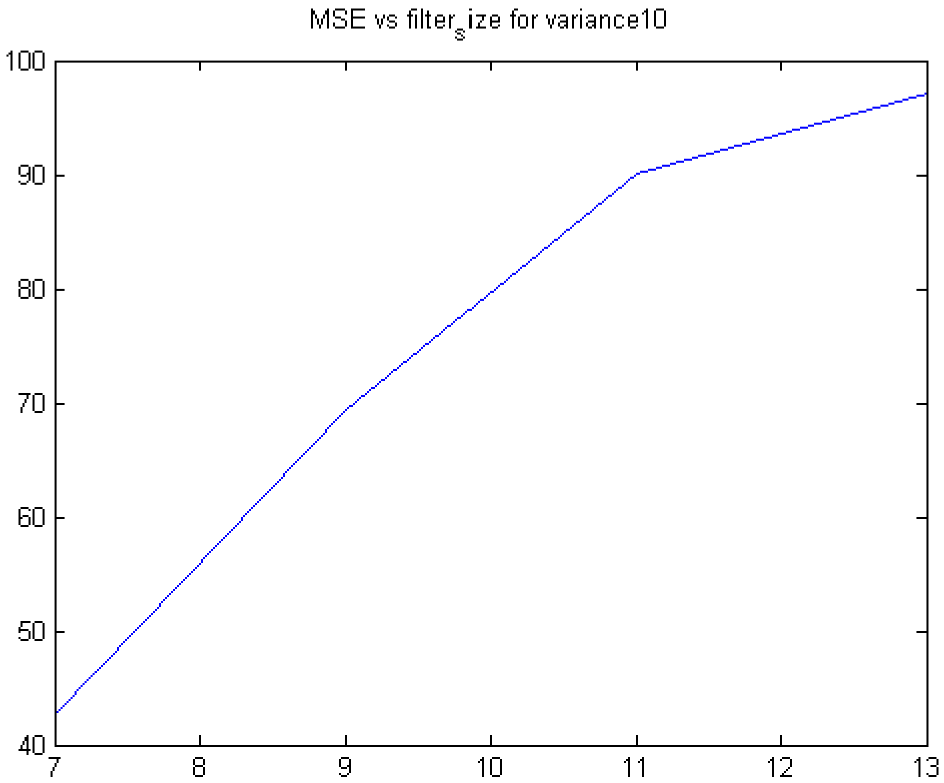
Published with MATLAB® R2013a

Comment - The MSE increases as the filter size increases

Question 2D

```
function[]=q2d()
    variance=10;
    n_size=15;
    for i=1:4
       p_size=5+2*i;
       display(i);
       MSE(i)=q2d_sub(variance,n_size,p_size);
       x(i)=p_size;
    end
    figure;plot(x,MSE);title(['MSE vs filter_size for variance'
num2str(variance)]);
end

function[MSE_avg]=q2d_sub(noise_variance, n_size,p_size)
    f=double(imread('CircleSquare.tif'));
    f=f(1:4:end,1:4:end);% done so as to reduce the time
    MSE(1:10)=0;x(1:10)=0;%initialisation
    fmax = max(f(:));
    I = 100;
    a = 1.1;
    f = f/fmax*I;
    orignal_image=f;
    filtered_image=f;filtered_image(:)=0;
     for i=1:10
        snr = I^2/noise_variance; % = I^2/sigma^2
        sigma = sqrt(I^2/snr);
        gaussian_noise = randn(size(f))*sigma;
%         display(size(f));display(size(gaussian_noise));
        noisy_image=f+gaussian_noise;

[filtered_image,MSE(i)]=nlm_filter(noisy_image,orignal_image,n_size,p_size
);
       if(i==1)
%             figure;imagesc(noisy_image);title('noisy_image');
%             figure;imagesc(orignal_image);title('orignal_image');
%             figure;imagesc(filtered_image);title('filtered image');
%             pause(5);
%
       end
              %figure;imagesc(filtered_image);title('filtered
image');pause(2);
     end
    MSE_avg=sum(MSE(:))/size(MSE,1);
end

function[imout,MSE]=nlm_filter(noisy_image,orignal_image,n_size,p_size)
    threshold=100;
    [s1,s2]=size(orignal_image);
    imout(1:s1,1:s2)=0;
    %n_size=7;p_size=3;
    %f1=figure;

%    f=padarray(f,[floor(n_size/2),floor(n_size/2)]);

noisy_image=padarray(noisy_image,[floor(n_size/2),floor(n_size/2)],'symmet
ric');
    for m=1+floor(n_size/2):s1+floor(n_size/2)
        for n=1+floor(n_size/2):s1+floor(n_size/2)
%             sub_g=f(m-floor(n_size/2):m+floor(n_size/2),n-
floor(n_size/2):n+floor(n_size/2));
            patch_mn=noisy_image(m-floor(p_size/2):m+floor(p_size/2),n-
floor(p_size/2):n+floor(p_size/2));

           % f(m-floor(p_size/2):m+floor(p_size/2),n-
floor(p_size/2):n+floor(p_size/2))=1;
%            f(m-floor(n_size/2):m+floor(n_size/2),n-
floor(n_size/2):n+floor(n_size/2))=2;
%            f(m-floor(p_size/2):m+floor(p_size/2),n-
floor(p_size/2):n+floor(p_size/2))=4;
%            figure(f1);imagesc(f);pause(0.1);
           sum=0;count=0;
            for s=1+floor(p_size/2):n_size-floor(p_size/2)
                for t=1+floor(p_size/2):n_size-floor(p_size/2)
                    a=m-1-floor(n_size/2);b=n-1-floor(n_size/2);
%                     f(a+s-floor(p_size/2):a+s+floor(p_size/2),b+t-
floor(p_size/2):b+t+floor(p_size/2))=1;
                    patch_st=noisy_image(a+s-
floor(p_size/2):a+s+floor(p_size/2),b+t-
floor(p_size/2):b+t+floor(p_size/2));
                    diff=abs(patch_mn-patch_st);
%                     display(size(diff));pause(10);
                    if(diff(:).*diff(:)<threshold)
                       sum=sum+noisy_image(a+s,b+t);count=count+1;
                    end
%                     figure(f1);imagesc(f);pause(0.01);
                end
            end
            imout(m-floor(n_size/2),n-floor(n_size/2))=sum/count;
        end
    end
%
noisy_image=noisy_image(1+floor(n_size/2):s1+floor(n_size/2),1+floor(n_siz
e/2):s2+floor(n_size/2));
    %display(size(imout));display(size(orignal_image));
    diff2=orignal_image-imout;
    MSE=diff2(:).*diff2(:)/(s1*s2);
    %display(size(MSE));
    %display(size(diff2));
    MSE_sum=0;
    for k=1:size(MSE,1)
        MSE_sum=MSE_sum+MSE(k,1);
    end
    MSE=MSE_sum;
%     figure;imagesc(orignal_image);
%     figure;imagesc(imout);
end
```

```
i =

     1


i =

     2


i =

     3


i =

     4
```



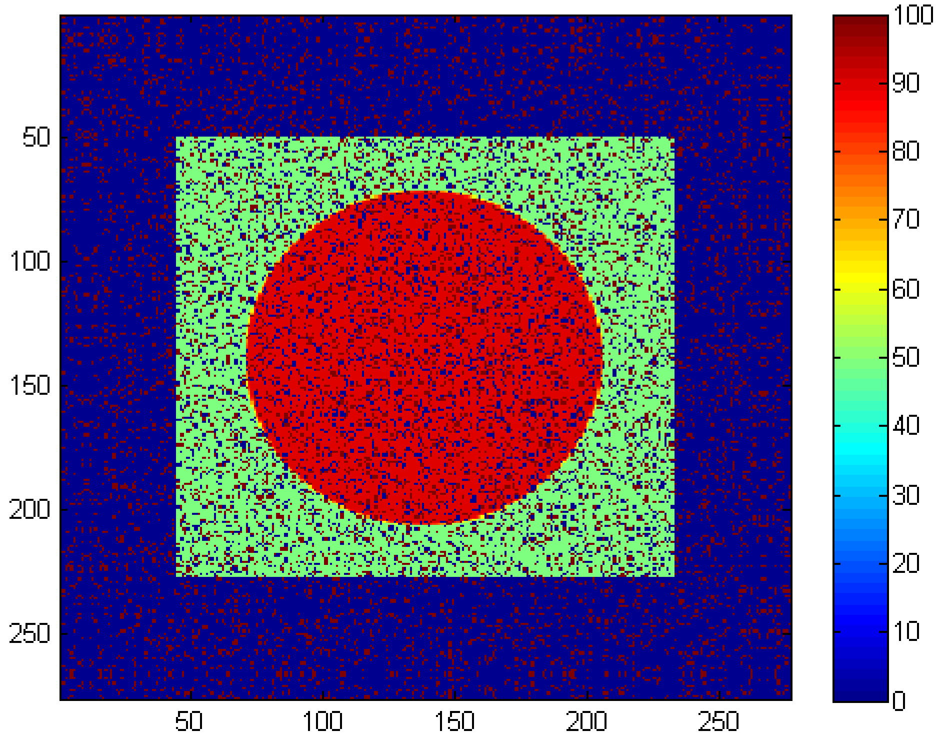
Published with MATLAB® R2013a

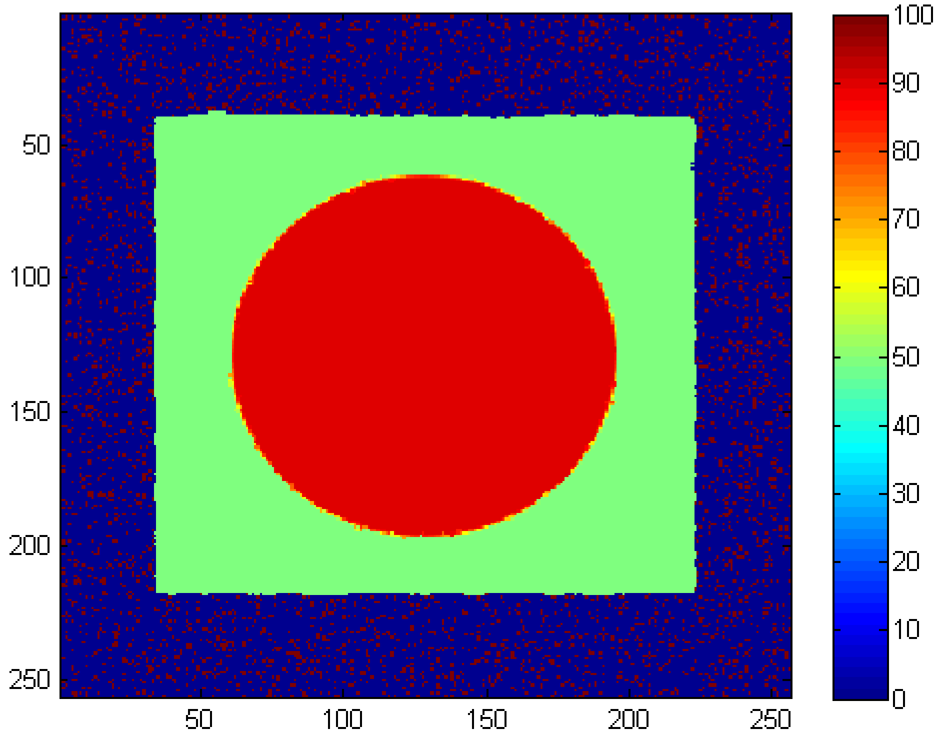
Comment- The MSE increases as the filter size increases. Only four data points are used because this non local means is a computationally heavy method

Question 2E

```
function[]=q2e()
    prob=0.25;
    image=imread('CircleSquare.tif');

    window_size_max=21;I=100;
    [s1,s2]=size(image);
     noisy_image=imnoise(image,'salt & pepper',prob);
     image=I*double(image)/max(double(image(:)));
     noisy_image=I*double(noisy_image)/max(double(noisy_image(:)));


noisy_image=padarray(noisy_image,[floor(window_size_max/2),floor(window_size_max/2)
],'symmetric');
    figure;imagesc(noisy_image);colorbar;
     imout=noisy_image;

     for i=1+floor(window_size_max/2):s1+floor(window_size_max/2)

         for j=1+floor(window_size_max/2):s2+floor(window_size_max/2)
```

```
             window_size=3;
            while(window_size<window_size_max)
               sub=noisy_image(i-floor(window_size/2): i+floor(window_size/2),j-
floor(window_size/2): j+floor(window_size/2));
               zmed=median(sub(:));
               zmin=min(sub(:));
               zmax=max(sub(:));
               A1=zmed-zmin;A2=zmed-zmax;
               if(A1>0&&A2<0)
                  B1=noisy_image(i,j)-zmin;
                  B2=noisy_image(i,j)-zmax;
                  if(B1>0&&B2<0)
                      imout(i,j)=noisy_image(i,j);window_size=window_size_max;
                  else
                       imout(i,j)=zmed;window_size=window_size_max;
                  end
               else
                   if(window_size<window_size_max)
                      window_size=window_size+2;
                   else
                       imout(i,j)=noisy_image(i,j);window_size=window_size_max;
                   end
               end
            end
         end
     end


imout_final=imout(1+floor(window_size_max/2):s1+floor(window_size_max/2),1+floor(wi
ndow_size_max/2):s2+floor(window_size_max/2));
     figure;imagesc(imout_final);colorbar;

end
```





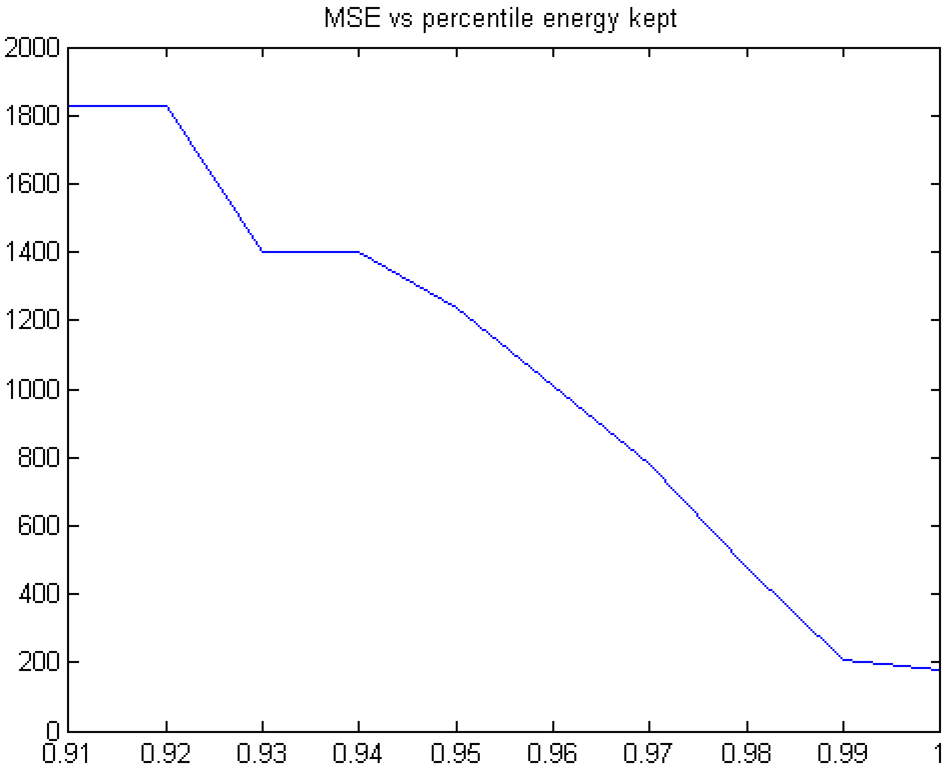
Published with MATLAB® R2013a

Question 2F

```
function[]=q2f()
    noise_variance=10;
    %thresholding using Fourier Coefficients
    for i=1:10
        percentile=0.9+0.01*i;
        MSE(i)=q2b_sub(percentile,noise_variance);
        x(i)=percentile;
    end
    figure;plot(x,MSE);title('MSE vs percentile energy kept');
end

function[Avg_MSE]=q2b_sub(percentile_thresh,noise_variance)
    f=double(imread('CircleSquare.tif'));
    MSE(1:10)=0;x(1:10)=0;%initialisation
    fmax = max(f(:));
    I = 100;
    a = 1.1;
    f = f/fmax*I;
    fmin = 0;
    fmax = max(f(:));
    orignal_image=f;
     for i=1:10
        snr = I^2/noise_variance; % = I^2/sigma^2
        sigma = sqrt(I^2/snr);
        gaussian_noise = randn(size(f))*sigma;
        noisy_image=f+gaussian_noise;

[filtered_image,MSE(i)]=fourier_thresholding_filter(noisy_image,orignal_im
age,percentile_thresh);
%        figure;imagesc(f);
%        figure;imagesc(noisy_image);
%        figure;imagesc(filtered_image);title(['MSE=' num2str(MSE(i))]);
%        pause(20);
       x(i)=i;
       if(i==1)
%             figure;imagesc(noisy_image);title('noisy_image');
%             figure;imagesc(orignal_image);title('orignal_image');
            figure;imagesc(filtered_image);title('filtered image');
             pause(1);
            close all;
       end
     end
    Avg_MSE=sum(MSE(:))/size(MSE,1);
```

```
end

function[filtered_image,MSE]=fourier_thresholding_filter(noisy_image,orign
al_image,percentile_thresh)
% percentile_thresh is the percent of energy which would remain in
filtered
% image
    F_noisy_image=fftshift(fft2(noisy_image));
    F_orignal_image=fftshift(fft2(orignal_image));
%     figure;imagesc(log(1+abs(F_noisy_image)));colorbar;
%     figure;imagesc(log(1+abs(F_orignal_image)));colorbar;
    X=abs(F_noisy_image).*abs(F_noisy_image);
    max_X=max(X(:));
%     figure;hist(X(:),100);title('histogram');
    [s1,s2]=size(noisy_image);
    sum_X=sum(X(:));
    X=X/sum_X;
    F_filtered_image=F_noisy_image;F_filtered_image(:,:)=0;
    default_radius=50;
    for default_radius=1:30
        energy_sum=0;
        for i=-default_radius:default_radius
            for j=-default_radius:default_radius
                D=sqrt((i)^2+(j)^2);
                if(D<default_radius)
                    energy_sum=energy_sum+X(floor(s1/2)+i,floor(s1/2)+j);
                end
            end
        end
%         display(energy_sum);
        if(energy_sum>percentile_thresh)
           break;
        end
    end
        for i=-default_radius:default_radius
            for j=-default_radius:default_radius
                D=sqrt((i)^2+(j)^2);
                if(D<default_radius)

F_filtered_image(floor(s1/2)+i,floor(s2/2)+j)=F_noisy_image(floor(s1/2)+i,
floor(s2/2)+j);
                end
            end
        end
    filtered_image=real(ifft2(fftshift(F_filtered_image)));
```

```
%
    diff=orignal_image-filtered_image;
    MSE=sum(diff(:).*diff(:))/(s1*s2);

end
```



Published with MATLAB® R2013a

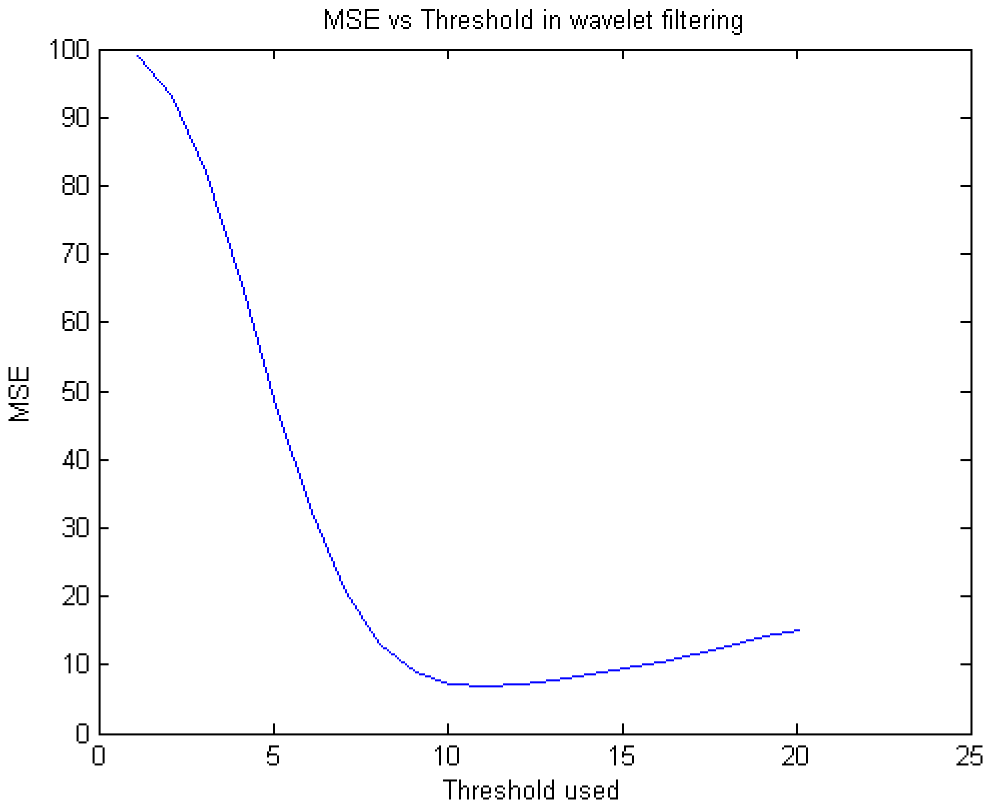
Comment- the MSE decreases as we keep more coefficients to save more energy in the filtered image but the drawback is the fringing effect and addition of artefacts in the filtered image

Question 2 g

```
function [  ] = q2g()
    % as the threshold reaches 10 the MSE decreases after that is starts
    % increasing again. This value 10 is due to the specific noise level
    s_image=128;
    variance=10;
    %f = double(phantom(s_image));
    %threshold=0.01;
    for i=1:20
        threshold=0.1+1*i;
       MSE(i)= q2g_sub(threshold,variance);
       x(i)=threshold;
    end
    figure;plot(x,MSE);title('MSE vs Threshold in wavelet filtering');
    xlabel('Threshold used');ylabel('MSE');
    %display(MSE);
end

function[Avg_MSE]=q2g_sub(absolute_thresh,noise_variance)
    f=double(imread('CircleSquare.tif'));
    MSE(1:10)=0;x(1:10)=0;%initialisation
    fmax = max(f(:));
    I = 100;
    a = 1.1;
    f = f/fmax*I;
    orignal_image=f;
    filtered_image=f;filtered_image(:)=0;
     for i=1:10
        snr = I^2/noise_variance; % = I^2/sigma^2
        sigma = sqrt(I^2/snr);
        gaussian_noise = randn(size(f))*sigma;
%         display(size(f));display(size(gaussian_noise));
        noisy_image=f+gaussian_noise;

[filtered_image,MSE(i)]=wavelet_thresholding_filter(noisy_image,orignal_image,absolute_thresh
);
       if(i==1)
%             figure;imagesc(noisy_image);title('noisy_image');
%             figure;imagesc(orignal_image);title('orignal_image');
%             figure;imagesc(filtered_image);title('filtered image');
%             pause(2);
%             close all;
       end

     end
    Avg_MSE=sum(MSE(:))/size(MSE,1);
end

function[filtered_image,MSE]=wavelet_thresholding_filter(noisy_image,orignal_image,absolute_t
hresh)
    [s1,s2]=size(orignal_image);
    w=haar_LLevel(noisy_image,log2(s1));
    for i=1:s1
        for j=1:s1%assuming s1 and s2 are same, since haar wavelet is applicable to images of
size as powers of 2
            if(abs(w(i,j))<absolute_thresh)
                w(i,j)=0;
            end
        end
    end
%     figure;plot(w(:));
    filtered_image=invhaar_LLevel(w,log2(s1));
%     figure;imagesc(noisy_image);title('noisy_image');
%     figure;imagesc(orignal_image);title('orignal_image');
%     figure;imagesc(filtered_image);title('filtered image');
%     pause(5);
%     close all;
    diff=filtered_image-orignal_image;
    MSE=diff(:).*diff(:)/(s1*s1);
    MSE=sum(MSE(:));
```

```
end

function[w]=haar_LLevel(f,steps)
    s1=size(f,1);
    w=haar_oneLevel(f);
    for k=1:steps-1
        w(1:s1/2,1:s1/2)=haar_oneLevel(w(1:s1/2,1:s1/2));
        s1=s1/2;
    end

end

function w = haar_oneLevel(x)
    [M,N] = size(x);
    if M~=N
       error('image must be square');
    end

    if 2^round(log2(M))~=M
       error('sidelength must be power of two');
    end

    h00 = [1 1; 1 1]/2;
    h01 = [-1 1; -1 1]/2;
    h10 = [-1 -1; 1 1]/2;
    h11 = [1 -1; -1 1]/2;

    w00 = conv2(x,h00,'same');
    w00 = w00(1:2:end,1:2:end);

    w01 = conv2(x,h01,'same');
    w01 = w01(1:2:end,1:2:end);

    w10 = conv2(x,h10,'same');
    w10 = w10(1:2:end,1:2:end);

    w11 = conv2(x,h11,'same');
    w11 = w11(1:2:end,1:2:end);

    w = [w00 w01; w10 w11];

end

function [x] = invhaar_oneLevel(w)

[M,N] = size(w);
if M~=N
   error('image must be square');
end

if 2^round(log2(M))~=M
   error('sidelength must be power of two');
end

wup = kron(w,[0 0; 0 1]);

h00 = [1 1; 1 1]/2;
h01 = [1 -1; 1 -1]/2;
```

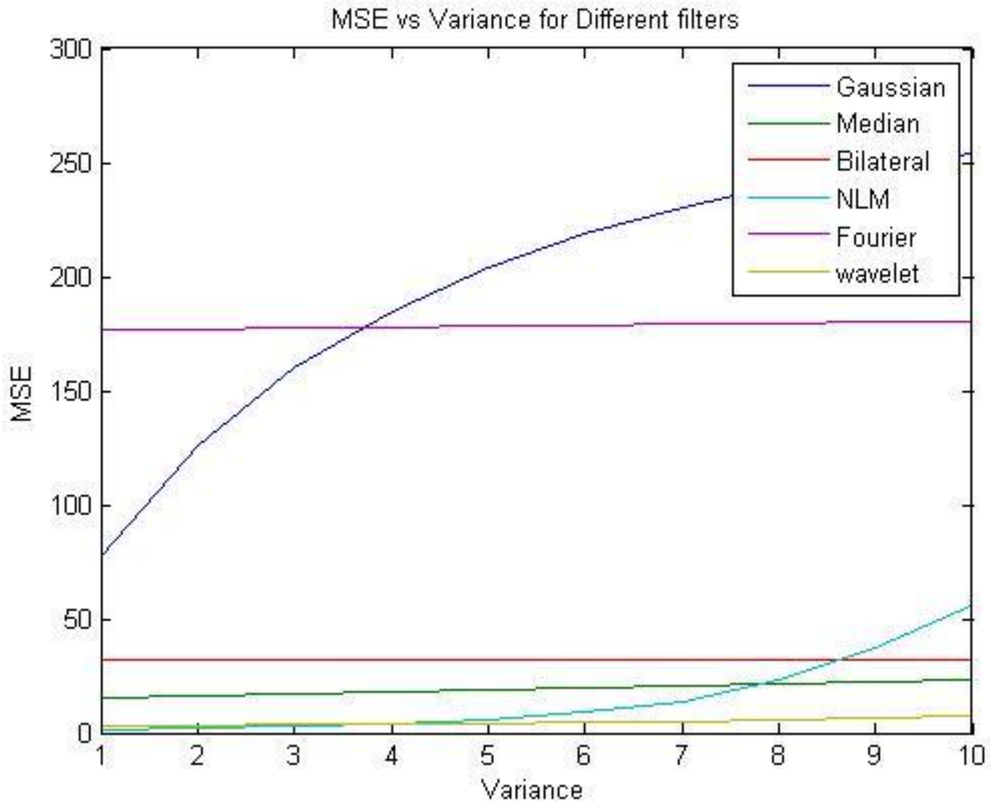
```
h10 = [1 1; -1 -1]/2;
h11 = [1 -1; -1 1]/2;


w00 = wup(1:M,1:M);
x00 = conv2(w00,h00,'same');

w01 = wup(1:M,((1:M)+M));
x01 = conv2(w01,h01,'same');

w10 = wup(((1:M)+M),1:M);
x10 = conv2(w10,h10,'same');

w11 = wup(((1:M)+M),((1:M)+M));
x11 = conv2(w11,h11,'same');

x = (x00+x01+x10+x11);
end

function[f]=invhaar_LLevel(w,num_steps)
    f=w;
    [s1,s2]=size(w);
    for i=num_steps:-1:1
        s_temp=power(2,i-1);
        %display(s_temp);
        f(1:s1/s_temp,1:s2/s_temp)=invhaar_oneLevel(f(1:s1/s_temp,1:s2/s_temp));
    end

end
```



Published with MATLAB® R2013a

Comment – The MSE decreases and then rises again with the increase in threshold with the minima at 10. This may be due to the fact of variance being 10

Question 2 i



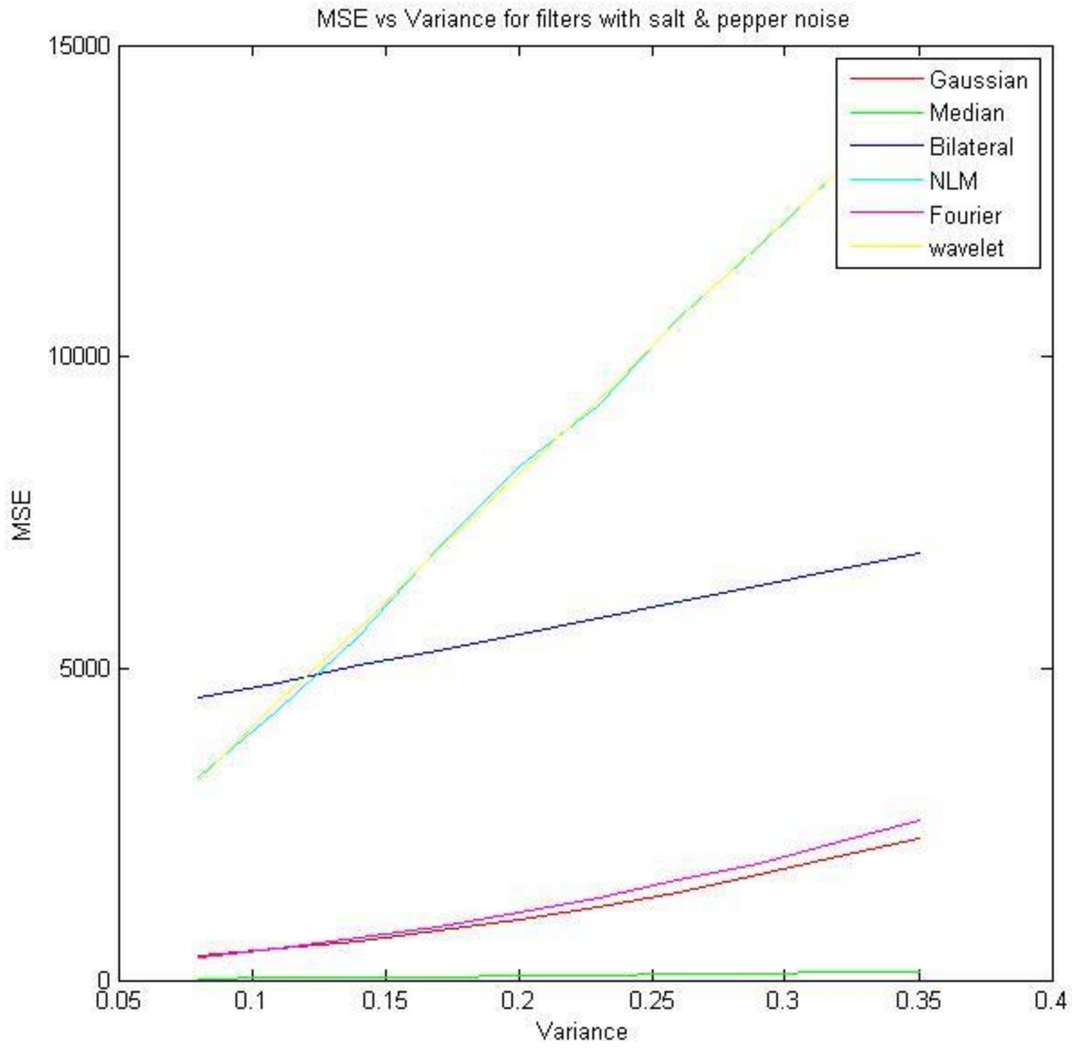
Plot of MSE versus Variance for different methods. Script used – q2i.m is attached in the zipped submission. Note to the evaluator – MATLAB parfor loops are used for faster computation. If the program is terminated by the user during a run, then the user would have to write matlabpool('close');on the command line before rerunning the program.

Question 2 j

- For Gaussian Noise the general order of preference for filtering comes out to be- Wavelet > Median>NLM>Bilateral>Fourier>Gaussian
- Wavelet filtering is the best method to remove noise because the resultant images had lowest MSE and did not contain artefacts compared to other methods

- In terms of MSE , Gaussian filter was the worst filter but in terms of visual perception Fourier domain thresholding was worse than Gaussian filtering in some cases because of addition of fringes and artefacts.
- In this case the filter with the minimum MSE also resulted in best looking images.

Question 3



- The order of performance of filters = Median>Gaussian>Fourier>Bilateral>NLM = wavelet
- The Median filter does the appropriate filtering and performs best
- Wavelet and NLM filters perform the worst
- Thus median filtering is the best filtering method for salt and pepper noise

Code for this question is attached as a part of the zip folder – q3.m and figure as Q3.fig