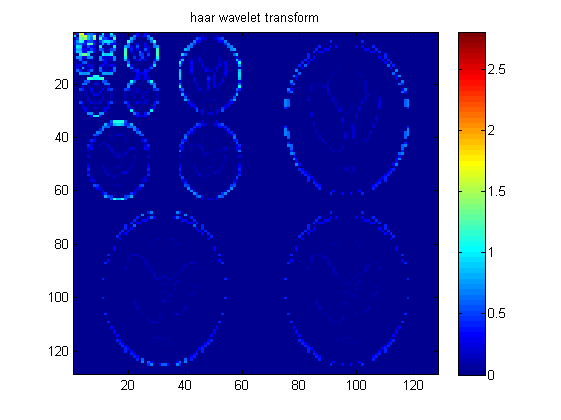
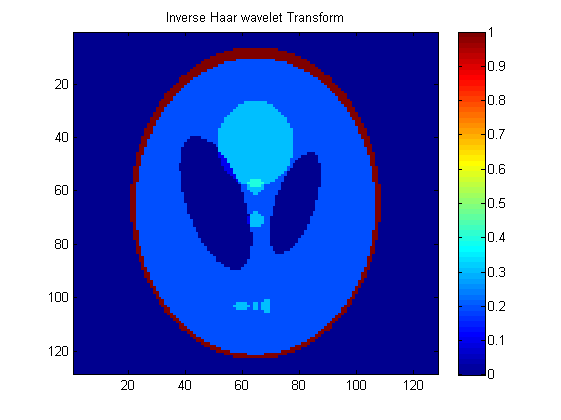
function [ ] = q1( )  
 %finding Wavelet Transform   
 s\_image=128;  
 f = double(phantom(s\_image));  
 w=haar\_LLevel(f,log2(s\_image));  
 figure;imagesc(log(1+abs(w)));colorbar;title('haar wavelet transform');  
 f\_out=invhaar\_LLevel(w,log2(s\_image));  
 figure;imagesc(f\_out);colorbar;title('Inverse 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

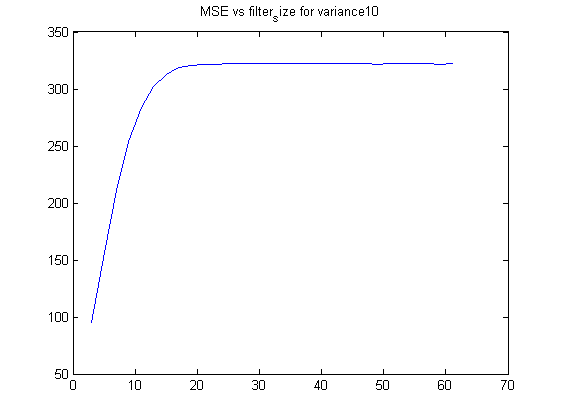




[*Published with MATLAB® R2013a*](http://www.mathworks.com/products/matlab)

**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

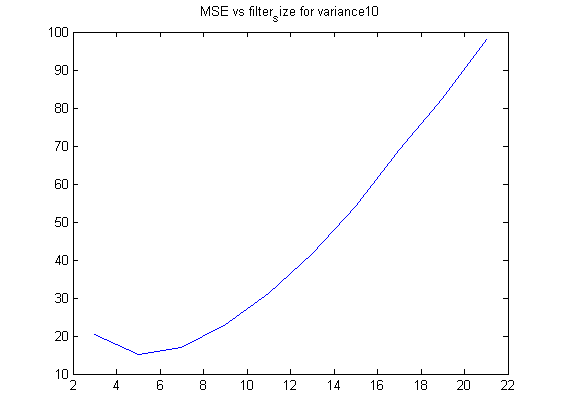


[*Published with MATLAB® R2013a*](http://www.mathworks.com/products/matlab)

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(neighborhood\_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

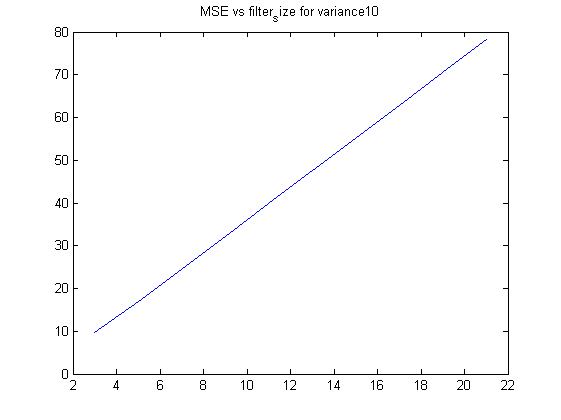


[*Published with MATLAB® R2013a*](http://www.mathworks.com/products/matlab)

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(neighborhood\_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



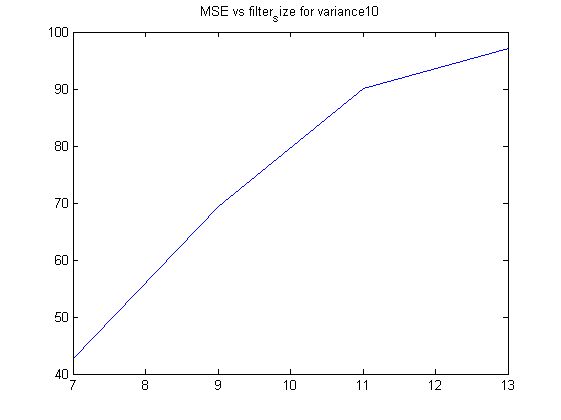
[*Published with MATLAB® R2013a*](http://www.mathworks.com/products/matlab)

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)],'symmetric');  
 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\_size/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

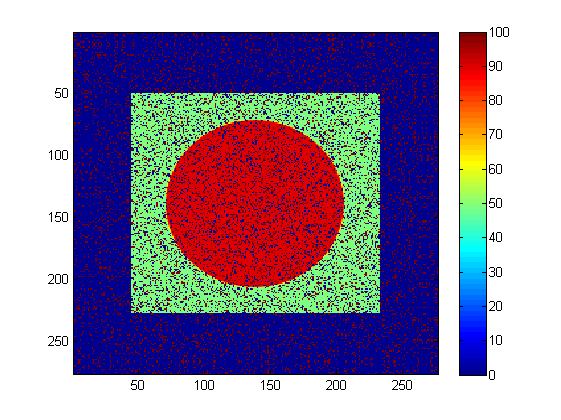


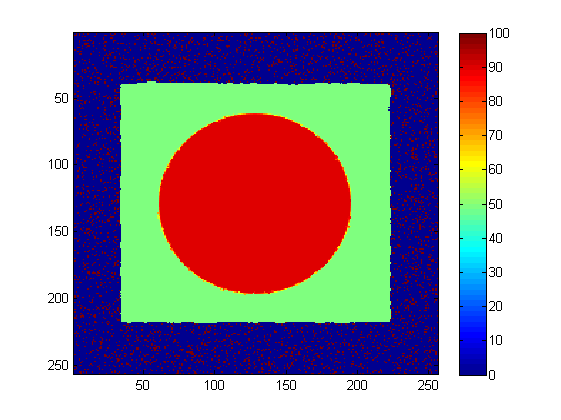
[*Published with MATLAB® R2013a*](http://www.mathworks.com/products/matlab)

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(window\_size\_max/2):s2+floor(window\_size\_max/2));  
 figure;imagesc(imout\_final);colorbar;  
  
end

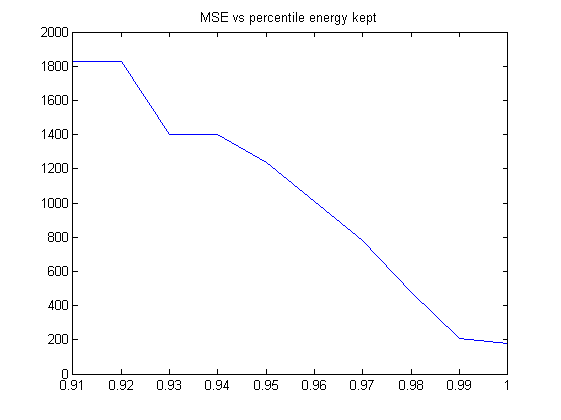




[*Published with MATLAB® R2013a*](http://www.mathworks.com/products/matlab)

**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\_image,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,orignal\_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

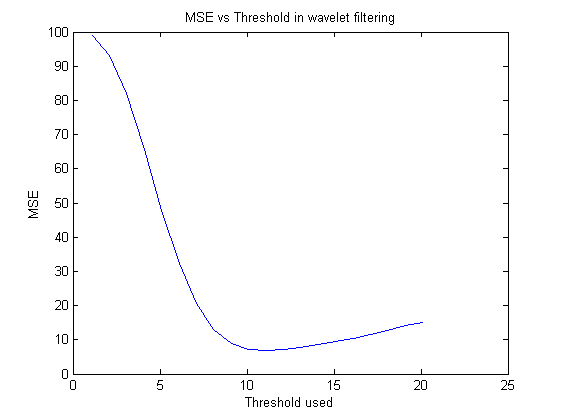


[*Published with MATLAB® R2013a*](http://www.mathworks.com/products/matlab)

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**

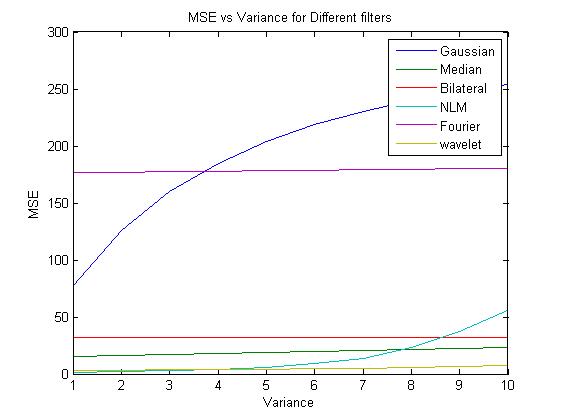
function [ ] = q2g()  
 % as the threshold reaches 10 the MSE decreases after that is starts  
 % increasing again. This value 10 is due to the specific noise level  
 s\_image=128;  
 variance=10;  
 %f = double(phantom(s\_image));  
 %threshold=0.01;  
 for i=1:20  
 threshold=0.1+1\*i;  
 MSE(i)= 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\_thresh)  
 [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*](http://www.mathworks.com/products/matlab)

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**

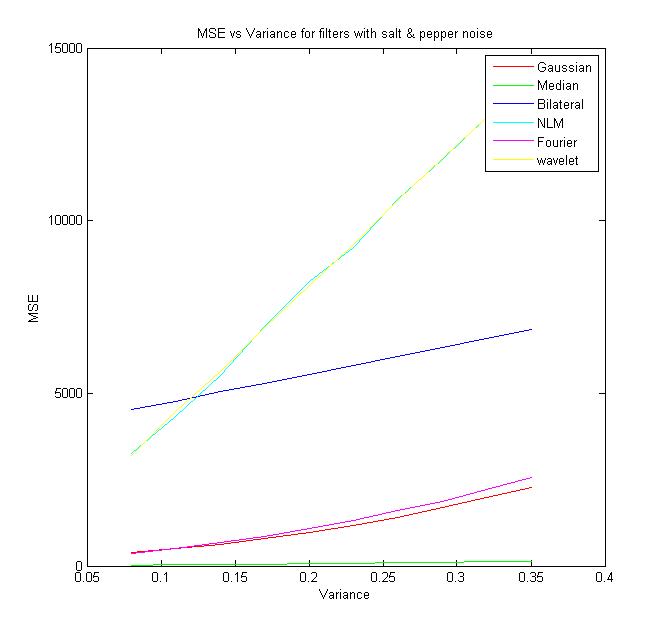
****

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