1. All computations were performed using the Regularization Tools provided by Hansen on the web. The Matlab file final1.m was used to complete the exercise. It is displayed below.

%set up the sharp and blurred NxN images

N=128;

[A,b,x]=blur(N);

%reshape and plot the sharp and blurred images

X=reshape(x,N,N);

B=reshape(b,N,N) ;

figure(1) ;

imagesc(X) ;

colormap(gray);

title(‘Original Image’);

figure(2);

imagesc(B);

colormap(gray);

title(‘Blurred Image’);

%set up the cgls computation

[X,rho,eta]=cgls(A,b,25,0);

%reshape and display images produced from the cgls computation

X5=reshape(X(:,5),N,N);

X10=reshape(X(:,10),N,N);

X15=reshape(X(:,15),N,N);

X20=reshape(X(:,20),N,N);

X25=reshape(X(:,25),N,N);

figure(3);

imagesc(X5);

colormap(gray);

title(‘Image after 5 CGLS Iterations’);

figure(4);

imagesc(X10);

colormap(gray);

title(‘Image after 10 CGLS Iterations’);

figure(5);

imagesc(X15);

colormap(gray);

title(‘Image after 15 CGLS Iterations’);

figure(6);

imagesc(X20);

colormap(gray);

title(‘Image after 20 CGLS Iterations’);

figure(7);

imagesc(X25);

colormap(gray);

title(‘Image after 25 CGLS Iterations’);

%add noise to the blurred image as instructed in the exercise

e=randn(size(b));

e=e/norm(e,’fro’);

b=b+.1\*norm(b,’fro’)\*e;

%reshape and display our new blurred image

B=reshape(b,N,N) ;

figure(8) ;

imagesc(B) ;

colormap(gray);

title(‘Blurred Image with Noise’);

%set up the cgls computations for our blurred image with noise

[X,rho,eta]=cgls(A,b,25,0);

%reshape and display images produced from this cgls calculation with noise

X5=reshape(X(:,5),N,N);

X10=reshape(X(:,10),N,N);

X15=reshape(X(:,15),N,N);

X20=reshape(X(:,20),N,N);

X25=reshape(X(:,25),N,N);

figure(9);

imagesc(X5);

colormap(gray);

title(‘Image after 5 CGLS Iterations with Noise’);

figure(10);

imagesc(X10);

colormap(gray);

title(‘Image after 10 CGLS Iterations with Noise’);

figure(11);

imagesc(X15);

colormap(gray);

title(‘Image after 15 CGLS Iterations with Noise’);

figure(12);

imagesc(X20);

colormap(gray);

title(‘Image after 20 CGLS Iterations with Noise’);

figure(13);

imagesc(X25);

colormap(gray);

title(‘Image after 25 CGLS Iterations with Noise’);

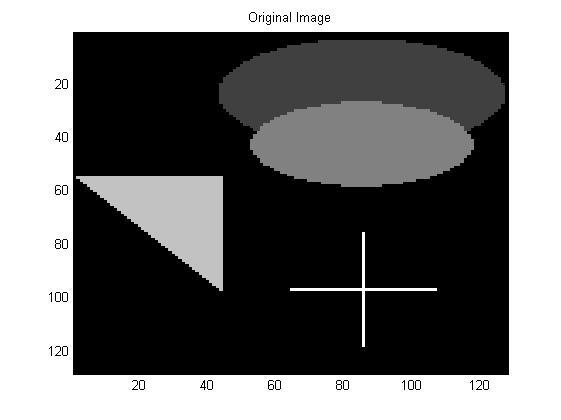


Figure 1

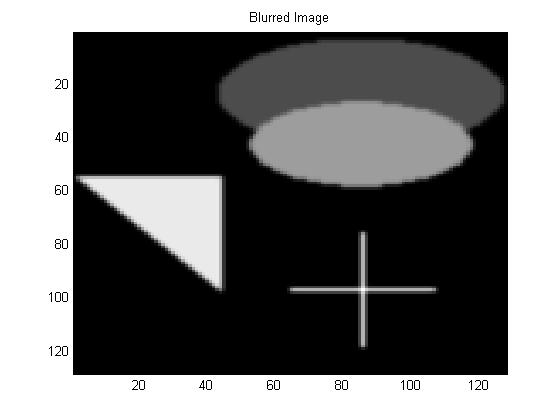


Figure 2

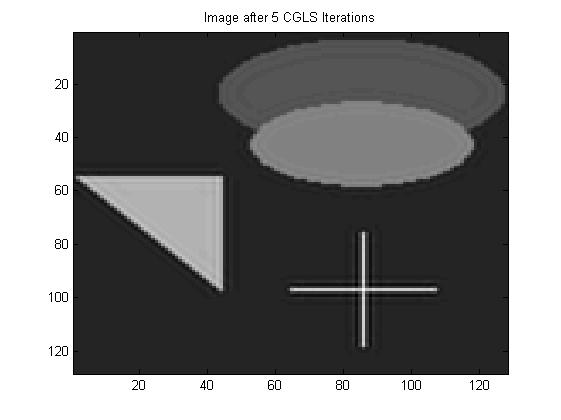


Figure 3

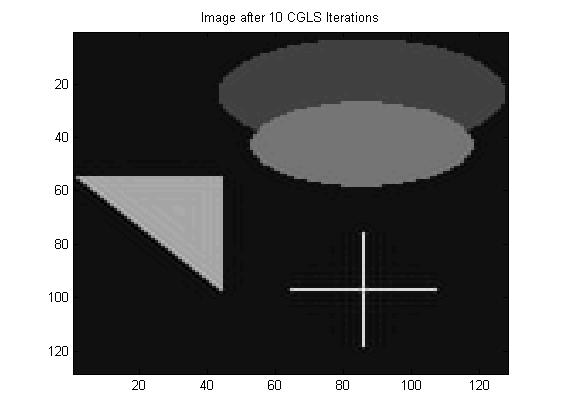


Figure 4

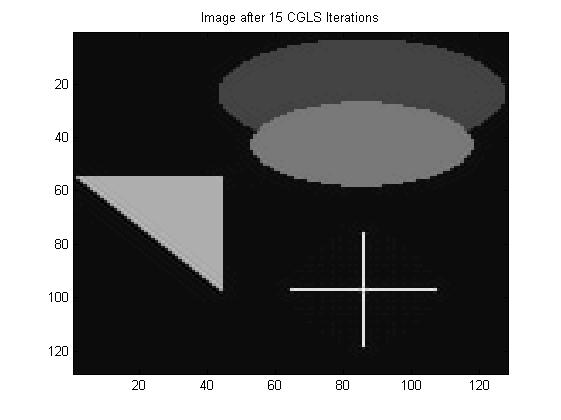


Figure 5

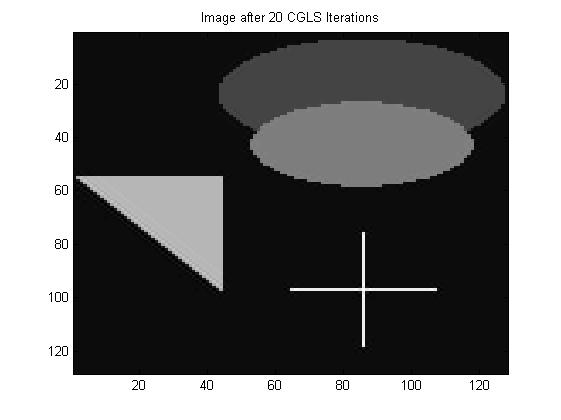


Figure 6

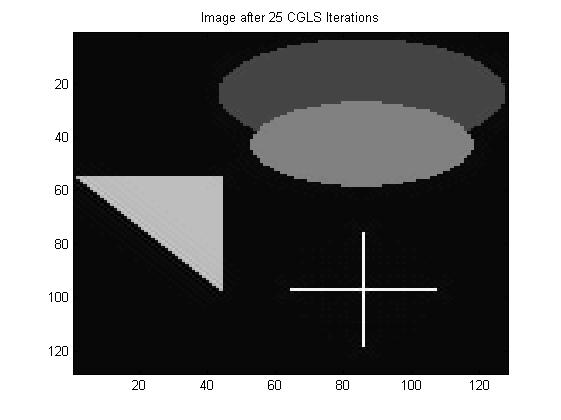


Figure 7

From these images, it is clear that as the number of iterations increases, the regularized image gets sharper. After 15 iterations it looks fairly close to the original image and most of the blur is no longer noticeable, except on the slanted edge of the triangle.

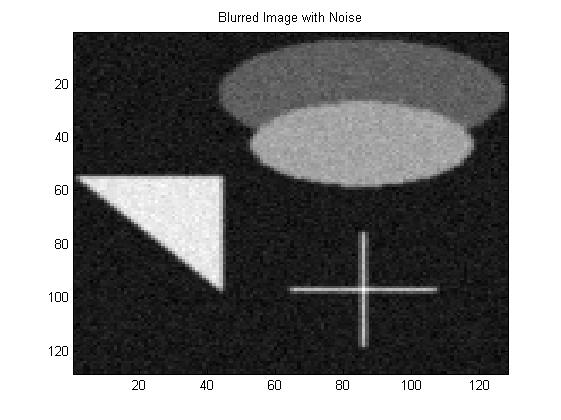


Figure 8

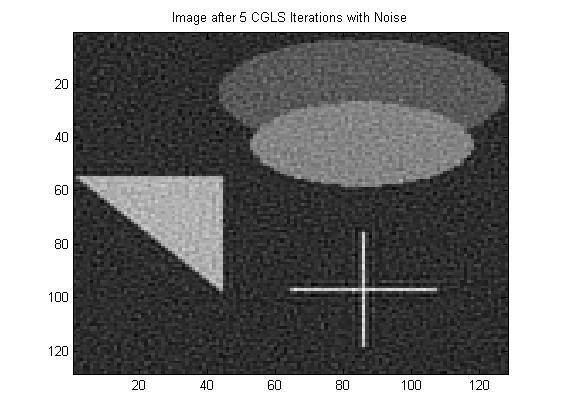


Figure 9

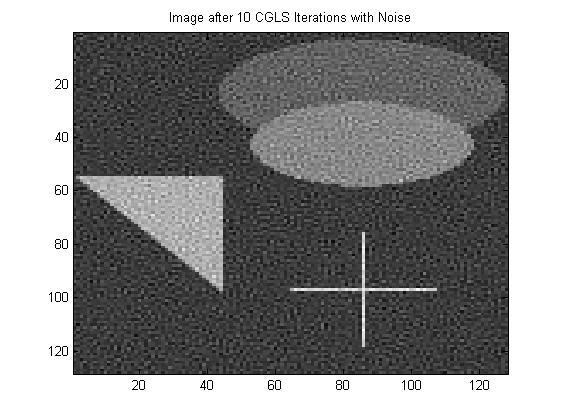


Figure 10

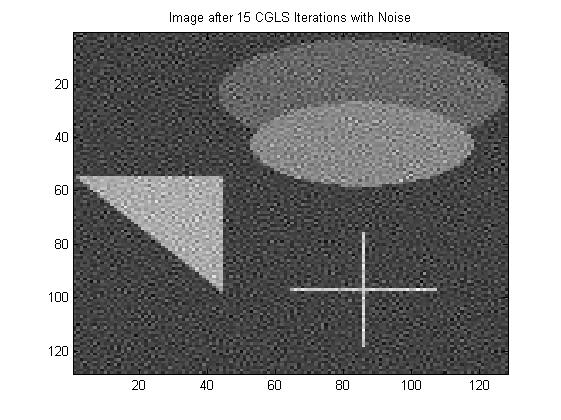


Figure 11

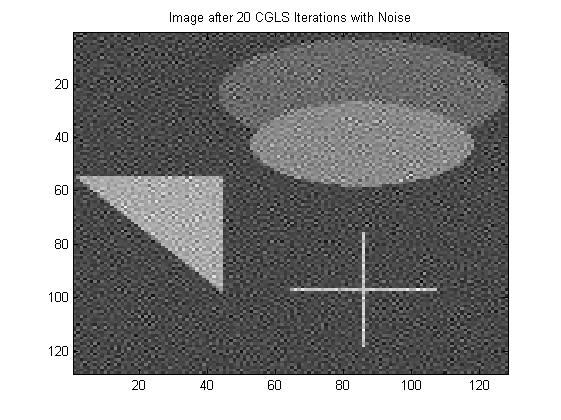


Figure 12

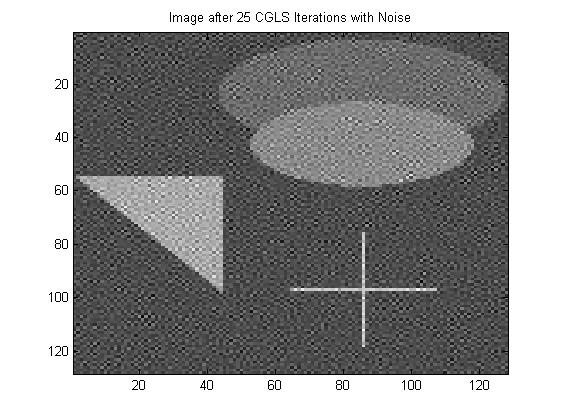


Figure 13

From the above images it can be seen that after 10 iterations, the noise begins to dominate the regularized image. This is due to the fact that for the CGLS algorithm, the number of iterations plays the role of the regularization parameter.

2. Noise in Images

1. The two different forms of noise are additive noise and multiplicative noise. Additive noise takes on the form

where µA is the additive noise. Additive noise has a statistical distribution,

often Gaussian, with a certain variation and mean.

Multiplicative noise takes on the form

where µM is the multiplicative noise. Multiplicative noise is a part of the integral equation and often has a Poisson distribution.

There can also be both multiplicative and additive noise in an image.

Gaussian noise is statistical noise that is additive. It has a normal or Gaussian amplitude distribution and is independent at each pixel.

Salt-and-pepper noise is impulsive noise that is usually caused by a detector

problem. It is characterized by completely black pixels in lighter regions of the

image or white pixels in darker regions of the image.

Shot noise usually occurs in cosmological images. It is caused by variation in the

number of photons sensed. It has a root-mean-square value that is proportional to

the square root of the image intensity. It is independent at each pixel and follows

a Poisson distribution.

Random noise is rare and can be due to the electrical signal in the detector. It can

be independent of the image.

Impulse noise can be positive or negative and either adds to or subtracts from

pixel values.

(b) SNR is defined in Decibels by

(c) Gaussian additive noise is primarily caused by Johnson-Nyquist, or thermal noise.

It is electronic noise caused by thermal agitation of electrons inside a conductor.

Poisson multiplicative noise is caused by background photons from natural and artificial sources. These photons cause corruption to each pixel value.

Salt and pepper noise can be caused by a variety of things. Most often it is due to dead pixels, but can also be caused by analog-to-digital converter errors and bit-errors in transmission.

Impulse noise occurs in semiconductors and is electronic noise. It is characterized by step transitions between voltages or current levels.

(d) For this problem I selected the pumpkins image from the HNO challenges. I first

added salt and pepper noise and removed this noise using an averaging filter and a

median filter. I then added Gaussian noise to the original image and removed this

noise using a Wiener filter. This was all performed in final2.m which is given

below

%set up original image

X=imread(‘pumpkins.tif’);

figure(1);

imagesc(X);

colormap(gray);

title(‘Original Image’);

%add salt and pepper noise to image

Xnoise=imnoise(X,’salt & pepper’) ;

figure(2) ;

imagesc(Xnoise) ;

colormap(gray) ;

title(‘Image with Salt and Pepper Noise’);

%remove salt and pepper noise with an averaging filter

X1=filter2(fspecial(‘average’),Xnoise) ;

figure(3) ;

imagesc(X1) ;

colormap(gray);

title(‘Image with Noise Removed using an Averaging Filter’);

%remove salt and pepper noise with a median filter

X2=medfilt2(Xnoise) ;

figure(4) ;

imagesc(X2) ;

colormap(gray);

title(‘Image with Noise Removed using a Median Filter’);

%add Gaussian noise to the original image

Xnoise2=imnoise(X,’gaussian’) ;

figure(5) ;

imagesc(Xnoise2);

colormap(gray);

title(‘Image with Gaussian Noise’);

%remove Gaussian noise with a Wiener filter

X3=wiener2(Xnoise2) ;

figure(6) ;

imagesc(X3) ;

colormap(gray);

title(‘Image with Noise Removed using a Wiener Filter’);

%remove Gaussian noise with a median filter

X4=medfilt2(Xnoise2) ;

figure(7) ;

imagesc(X4) ;

colormap(gray);

title(‘Image with Gaussian Noise Removed using a Median Filter’);

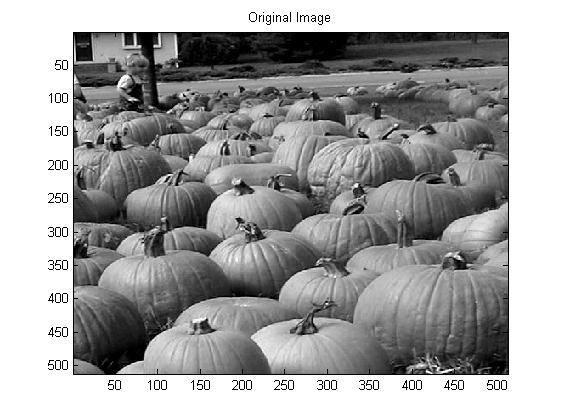


Figure 1

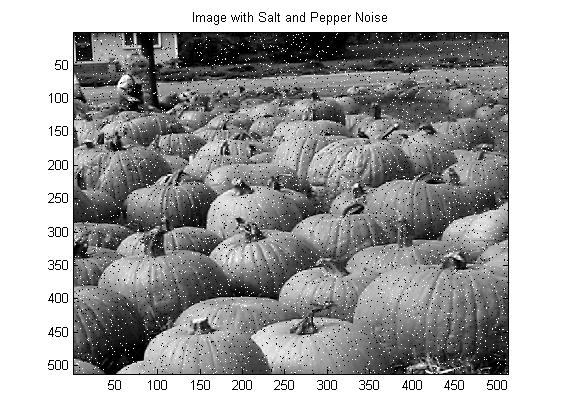


Figure 2

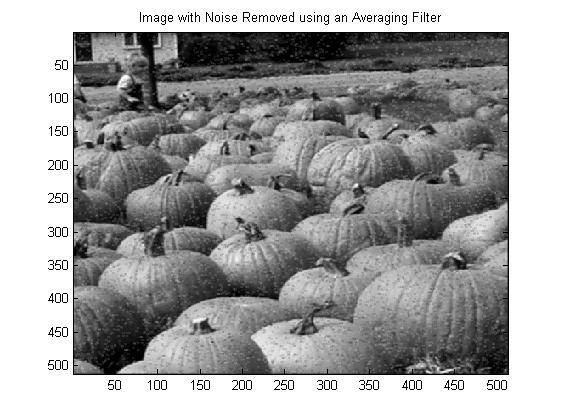


Figure 3

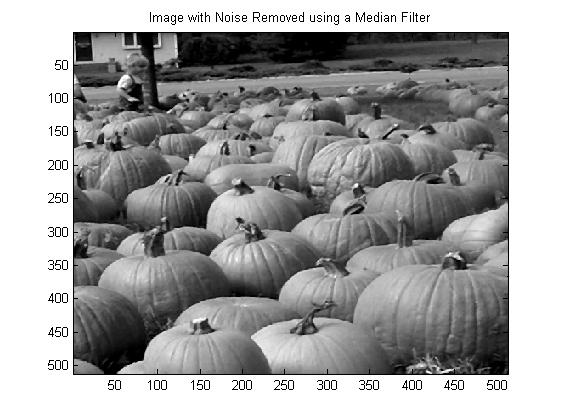


Figure 4

From the above figures it is obvious that the median filter does a much better job of removing salt and pepper noise than the averaging filter. It does a better job of preserving edges and the resulting image looks fairly close to the original image, whereas the image obtained using the averaging filter is clearly still noisy.

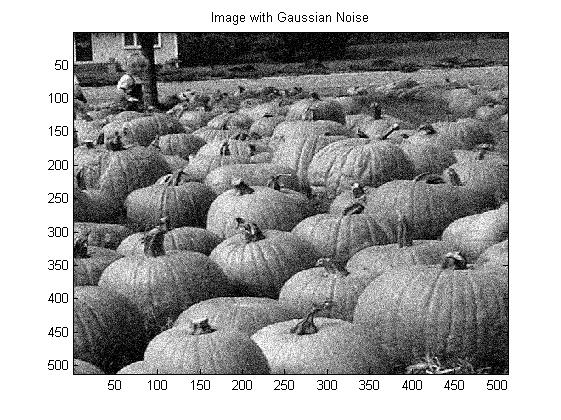


Figure 5

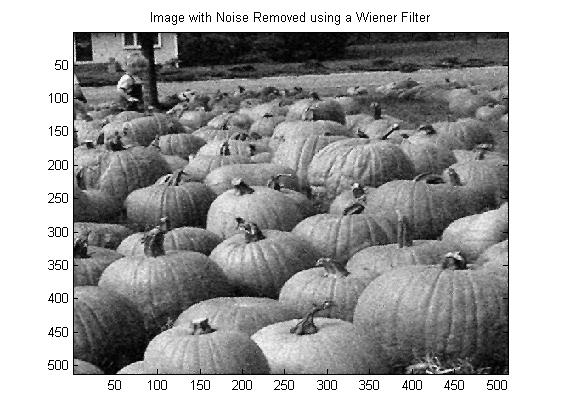


Figure 6

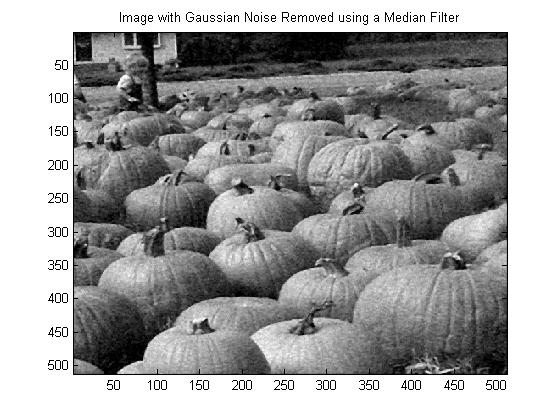


Figure 7

From the images above it can be seen that both the median and Wiener filters do a decent job of removing Gaussian noise from the image; however noise is still present in the filtered images. The median filter was able to remove salt and pepper noise better than Gaussian noise. It can also be seen, especially in the background of the image, that the Wiener filter does a slightly better job at removing Gaussian noise from the image. The type of Wiener filter used is an adaptive filter, which takes into account local image variance. Adaptive filters are often better than linear filters because they are more selective, which allows them to preserve images and other high-frequency regions.