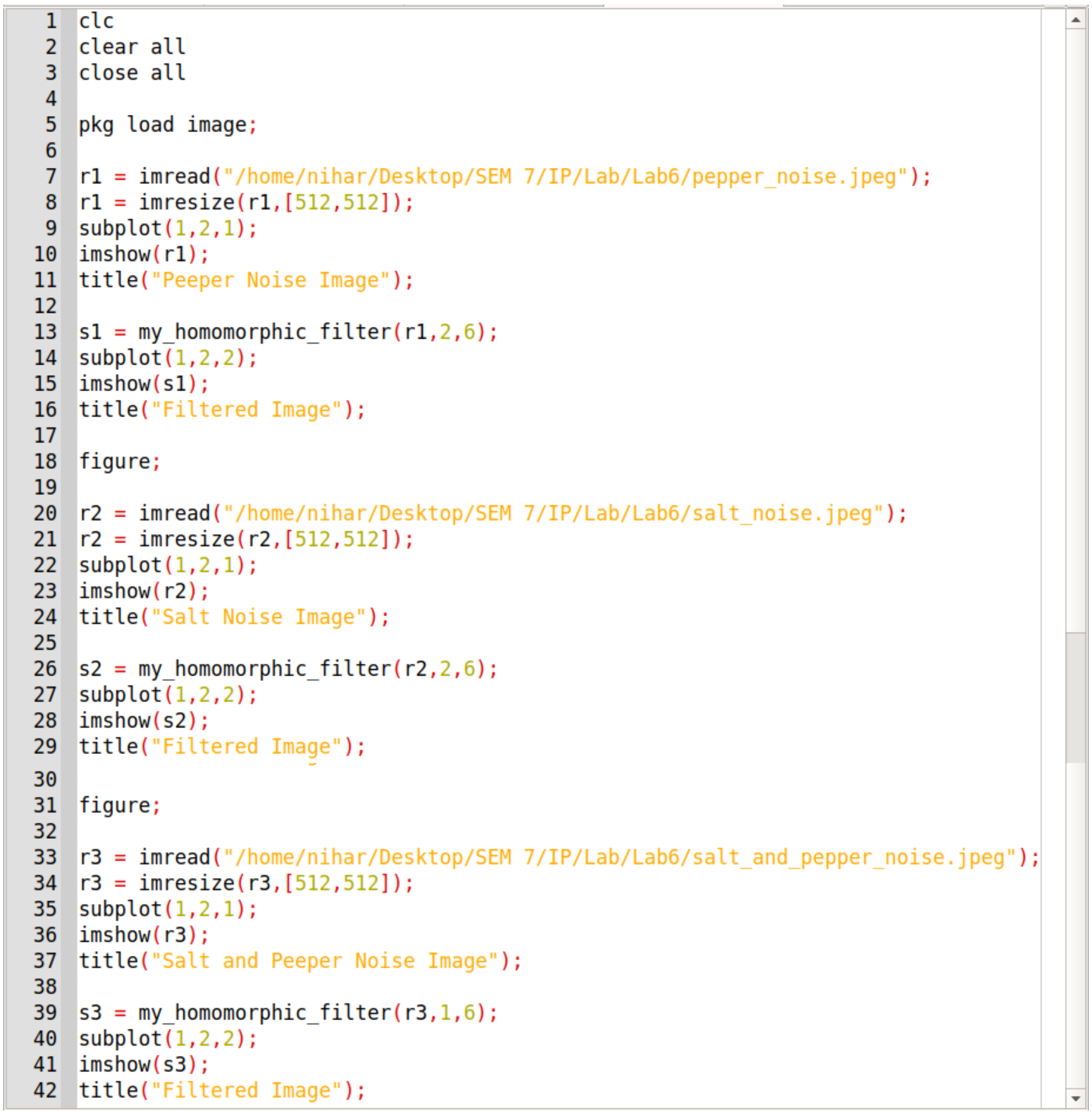
EXPERINMENT 7 & 8

Aim : Learn different filters and apply on image and also learn 1D DFT and IDFT.

❖ Exercises :

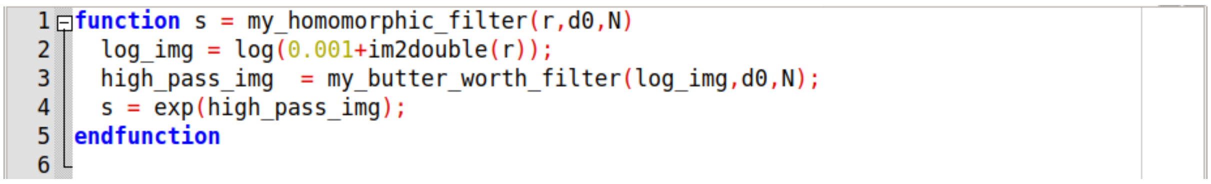
1. **Implement homomorphic filter and apply it to your photo with pepper noise, salt noise and salt and pepper noise. Conclude the results.**

Code :



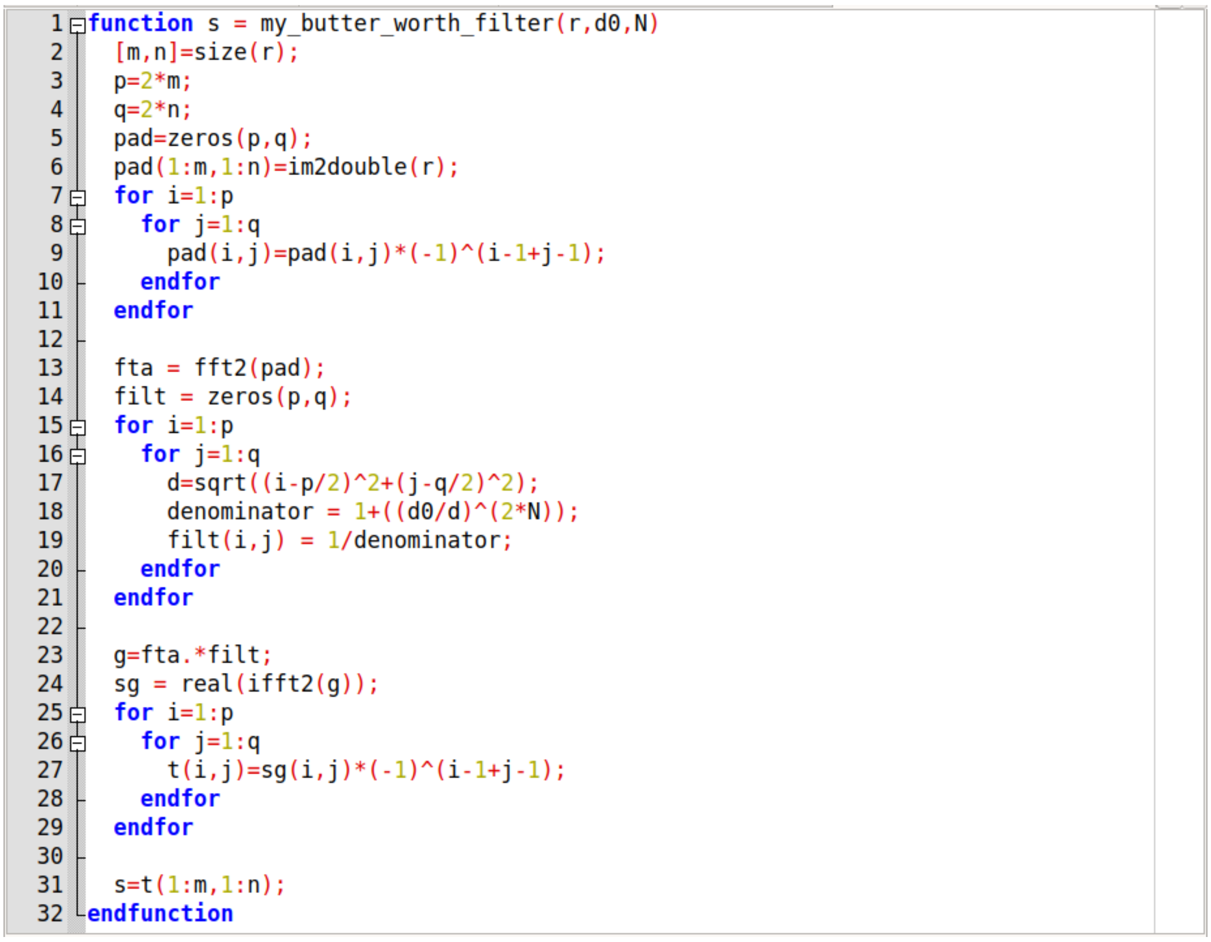
```
clc
clear all
close all

pkg load image;

r1 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/pepper_noise.jpeg");
r1 = imresize(r1,[512,512]);
subplot(1,2,1);
imshow(r1);
title("Peeper Noise Image");

s1 = my_homomorphic_filter(r1,2,6);
subplot(1,2,2);
imshow(s1);
title("Filtered Image");

figure;

r2 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/salt_noise.jpeg");
r2 = imresize(r2,[512,512]);
subplot(1,2,1);
imshow(r2);
title("Salt Noise Image");

s2 = my_homomorphic_filter(r2,2,6);
subplot(1,2,2);
imshow(s2);
title("Filtered Image");

figure;

r3 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/salt_and_pepper_noise.jpeg");
r3 = imresize(r3,[512,512]);
subplot(1,2,1);
imshow(r3);
title("Salt and Peeper Noise Image");

s3 = my_homomorphic_filter(r3,1,6);
subplot(1,2,2);
imshow(s3);
title("Filtered Image");
```

Function for Homomorphic Filter :



```
function s = my_homomorphic_filter(r,d0,N)
  log_img = log(0.001+im2double(r));
  high_pass_img  = my_butter_worth_filter(log_img,d0,N);
  s = exp(high_pass_img);
endfunction

```

Function for Butter Worth Filter :



```
function s = my_butter_worth_filter(r,d0,N)
  [m,n]=size(r);
  p=2*m;
  q=2*n;
  pad=zeros(p,q);
  pad(1:m,1:n)=im2double(r);
  for i=1:p
    for j=1:q
      pad(i,j)=pad(i,j)*(-1)^(i-1+j-1);
    endfor
  endfor

  fta = fft2(pad);
  filt = zeros(p,q);
  for i=1:p
    for j=1:q
      d=sqrt((i-p/2)^2+(j-q/2)^2);
      denominator = 1+((d0/d)^(2*N));
      filt(i,j) = 1/denominator;
    endfor
  endfor

  g=fta.*filt;
  sg = real(ifft2(g));
  for i=1:p
    for j=1:q
      t(i,j)=sg(i,j)*(-1)^(i-1+j-1);
    endfor
  endfor

  s=t(1:m,1:n);
endfunction
```

- Output of homomorphic filter depends on filter and value of parameters like d0, n(order of filter) and type of fitler.
- In my case image get overexposed so it can be used for low exposed image to enhance.

Output :

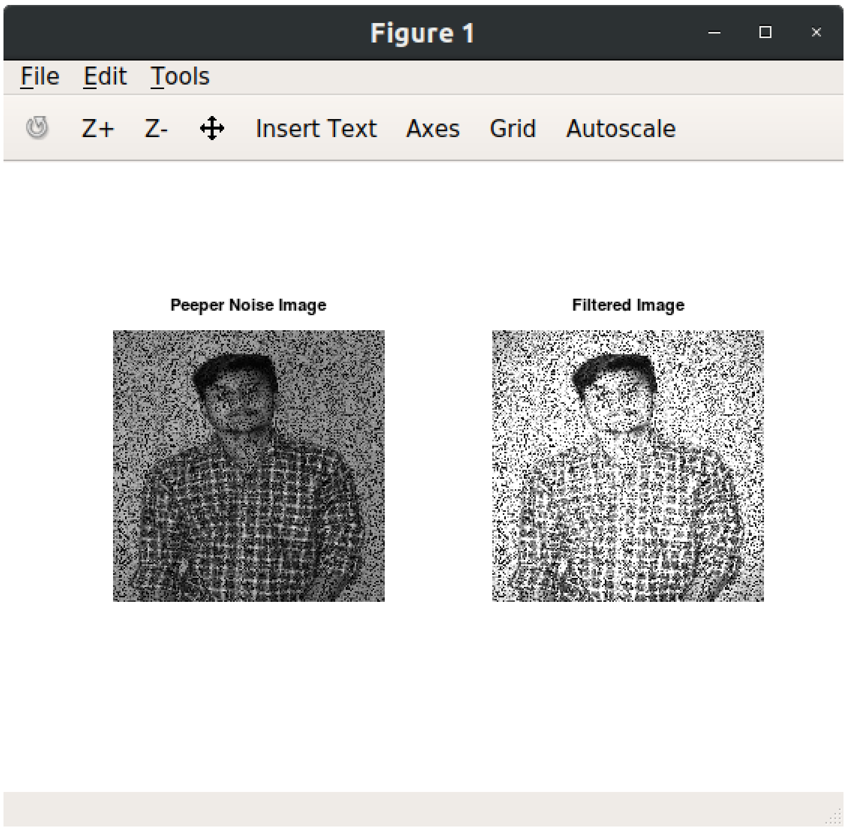


Figure 1
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Peeper Noise Image
Filtered Image

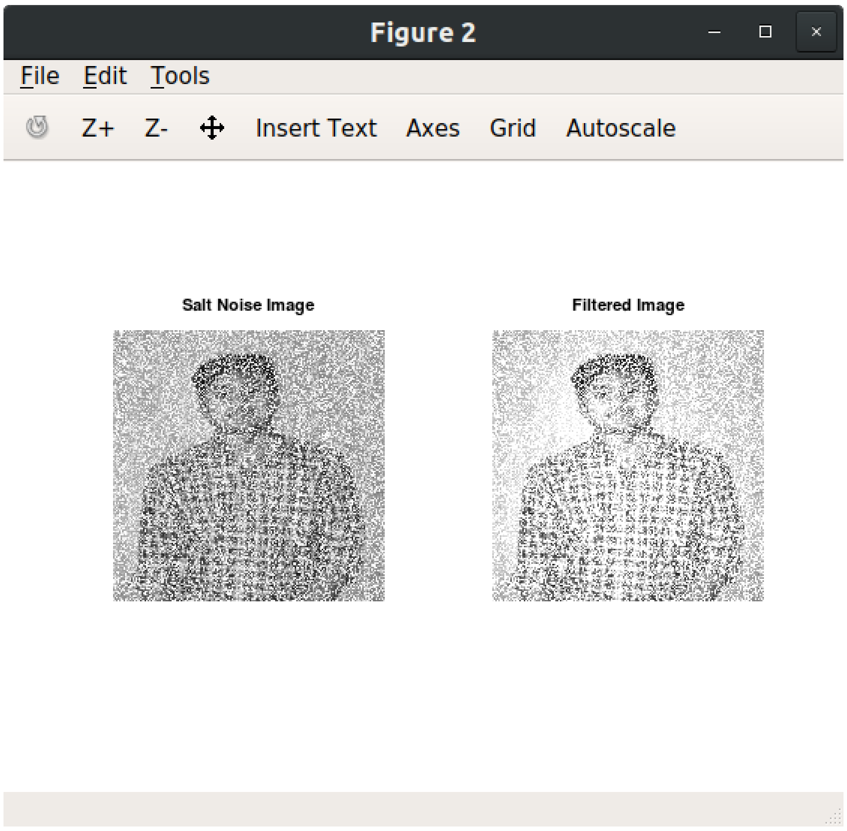


Figure 2
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Salt Noise Image
Filtered Image

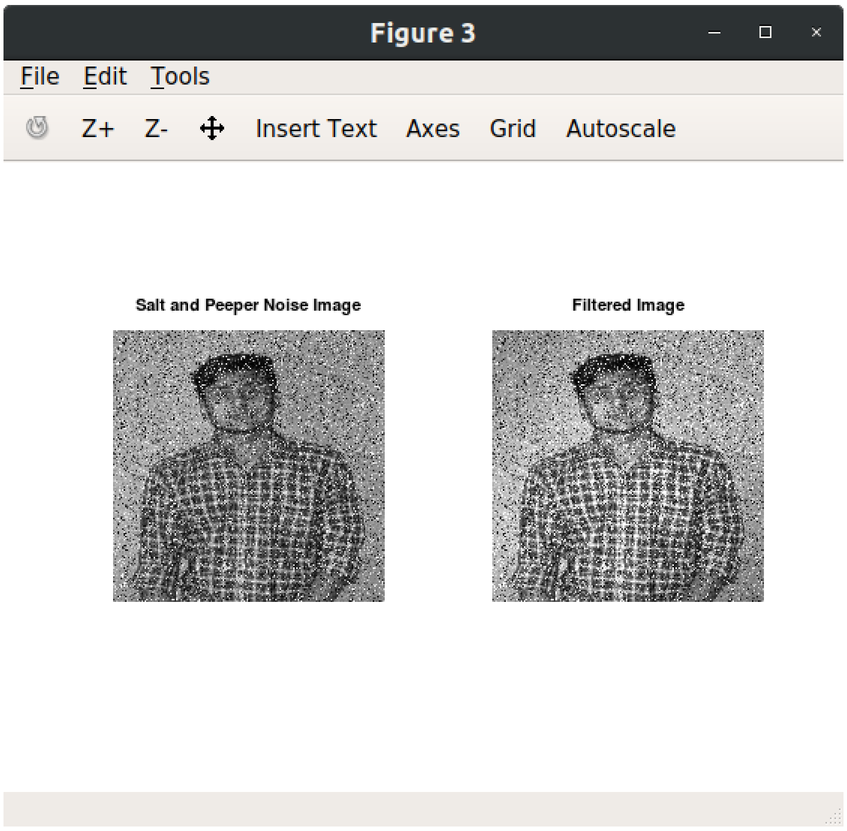


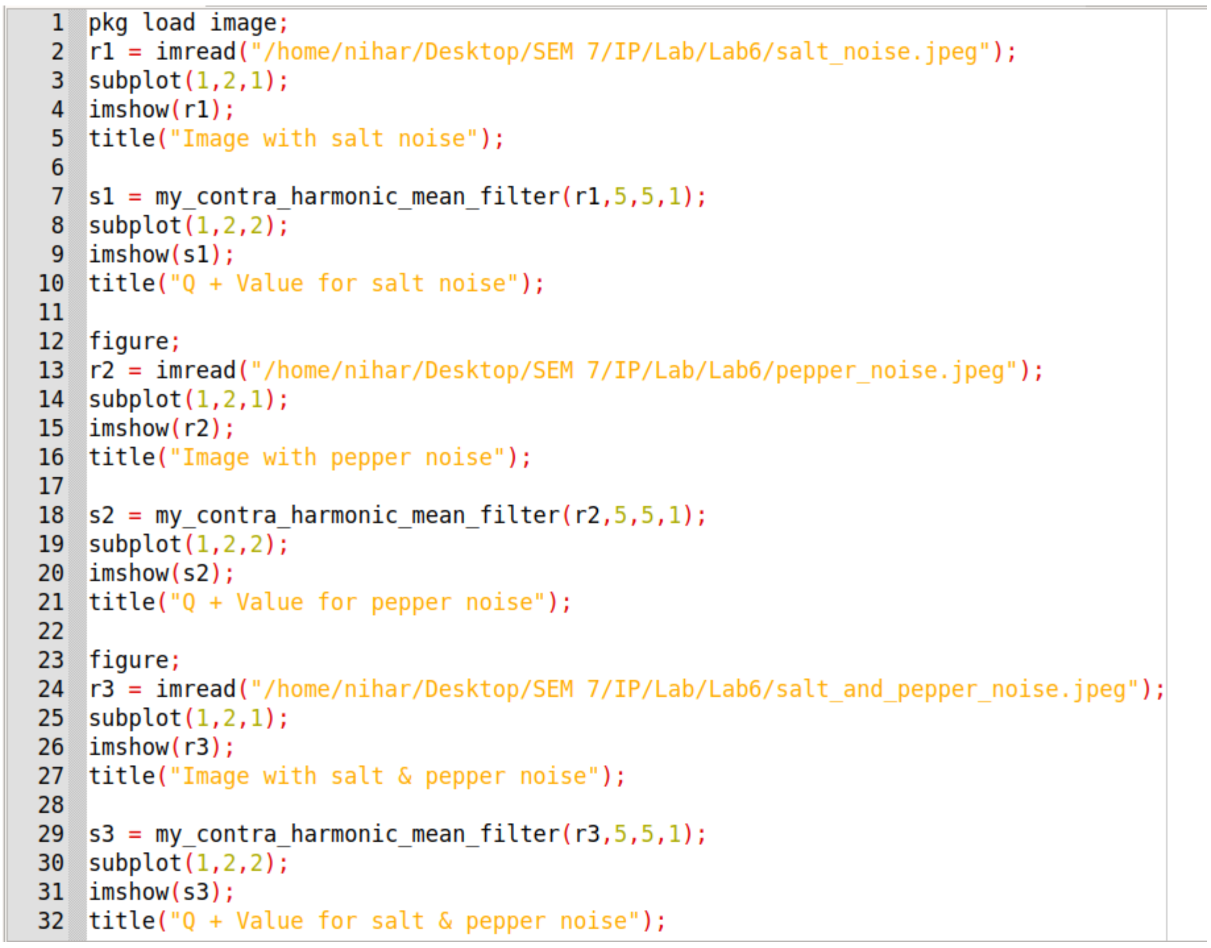
Figure 3
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Salt and Peeper Noise Image
Filtered Image

2. Use contra harmonic filter with Q +ve value for salt noise, pepper noise and salt and pepper noise. Comment on your outcome.

→ Solution :-

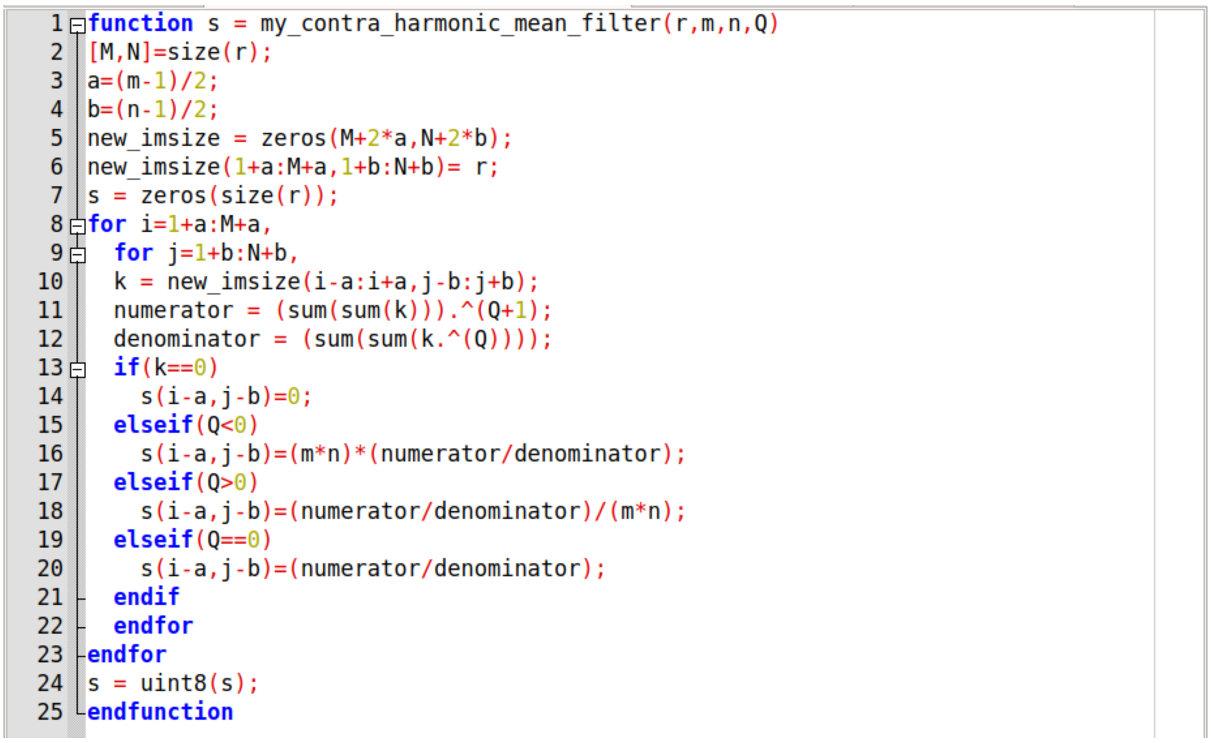
- For positive value of Q it removes pepper noise and increase noise for salt.
- For salt and pepper noise image is giving same effect as salt image because it has both salt and pepper noise. Pepper is noise is reduced but salt noise is increased.

Code :



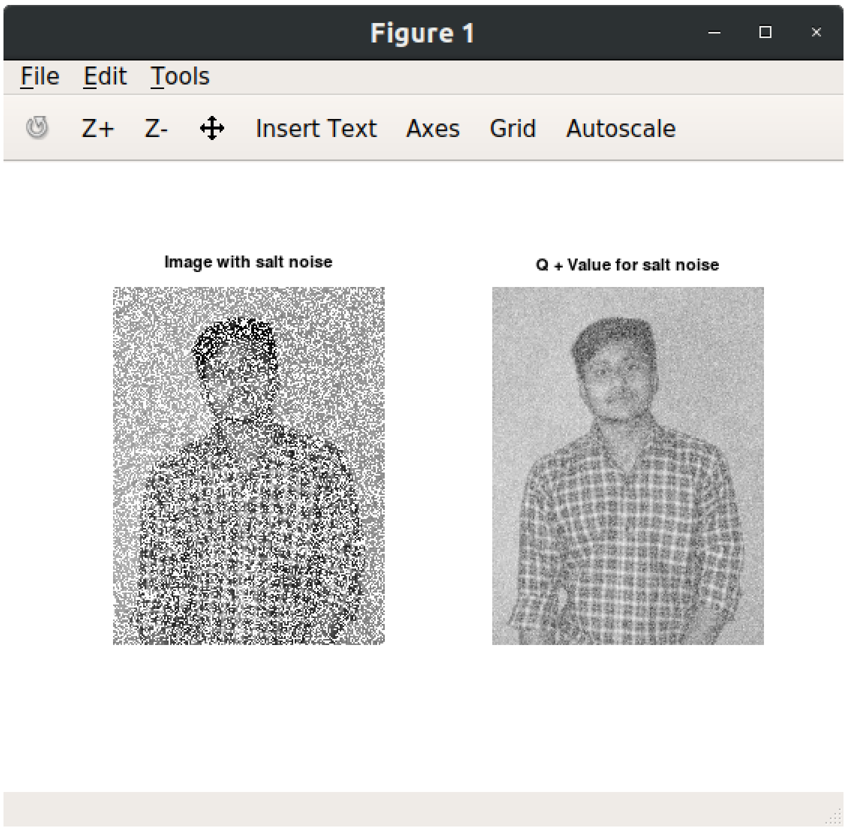
```
pkg load image;
r1 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/salt_noise.jpeg");
subplot(1,2,1);
imshow(r1);
title("Image with salt noise");

s1 = my_contra_harmonic_mean_filter(r1,5,5,1);
subplot(1,2,2);
imshow(s1);
title("Q + Value for salt noise");

figure;
r2 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/pepper_noise.jpeg");
subplot(1,2,1);
imshow(r2);
title("Image with pepper noise");

s2 = my_contra_harmonic_mean_filter(r2,5,5,1);
subplot(1,2,2);
imshow(s2);
title("Q + Value for pepper noise");

figure;
r3 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/salt_and_pepper_noise.jpeg");
subplot(1,2,1);
imshow(r3);
title("Image with salt & pepper noise");

s3 = my_contra_harmonic_mean_filter(r3,5,5,1);
subplot(1,2,2);
imshow(s3);
title("Q + Value for salt & pepper noise");
```

Function for Contra Harmonic Filter :



```
function s = my_contra_harmonic_mean_filter(r,m,n,Q)
[M,N]=size(r);
a=(m-1)/2;
b=(n-1)/2;
new_imsize = zeros(M+2*a,N+2*b);
new_imsize(1+a:M+a,1+b:N+b)= r;
s = zeros(size(r));
for i=1+a:M+a,
  for j=1+b:N+b,
  k = new_imsize(i-a:i+a,j-b:j+b);
  numerator = (sum(sum(k))).^(Q+1);
  denominator = (sum(sum(k.^(Q))));
  if(k==0)
    s(i-a,j-b)=0;
  elseif(Q<0)
    s(i-a,j-b)=(m*n)*(numerator/denominator);
  elseif(Q>0)
    s(i-a,j-b)=(numerator/denominator)/(m*n);
  elseif(Q==0)
    s(i-a,j-b)=(numerator/denominator);
  endif
  endfor
endfor
s = uint8(s);
endfunction
```

Output :



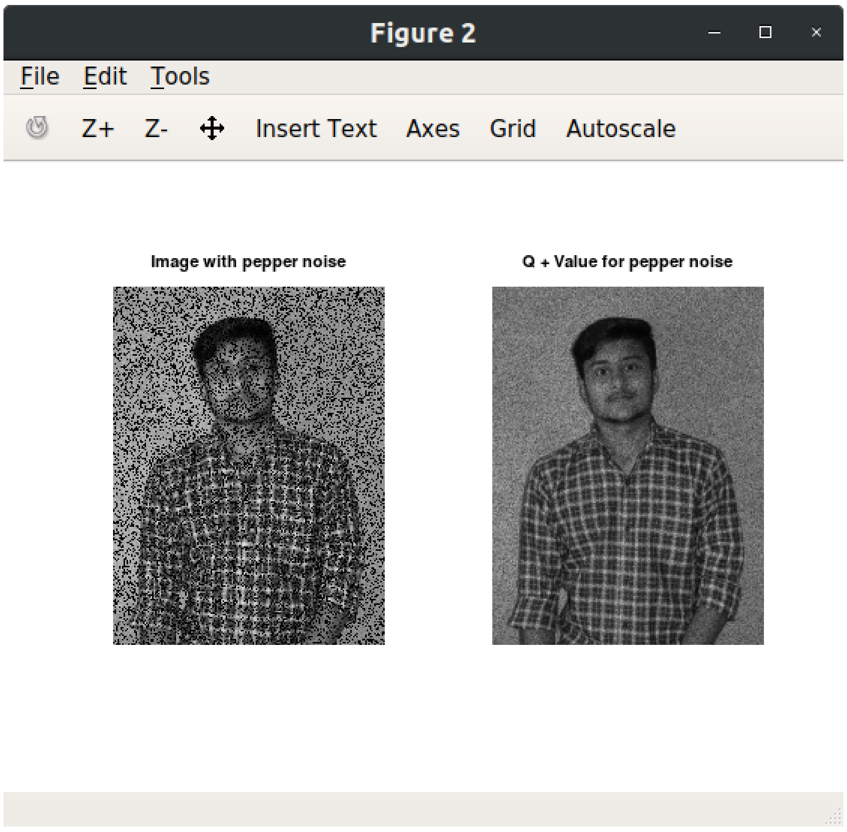


Figure 2
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Image with pepper noise
Q + Value for pepper noise

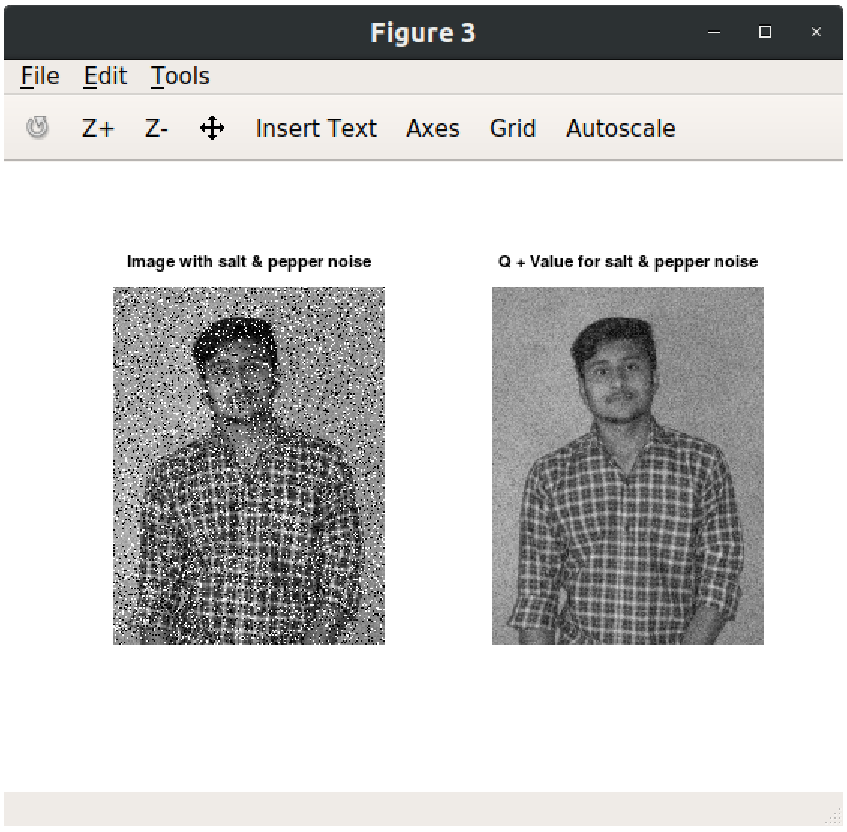


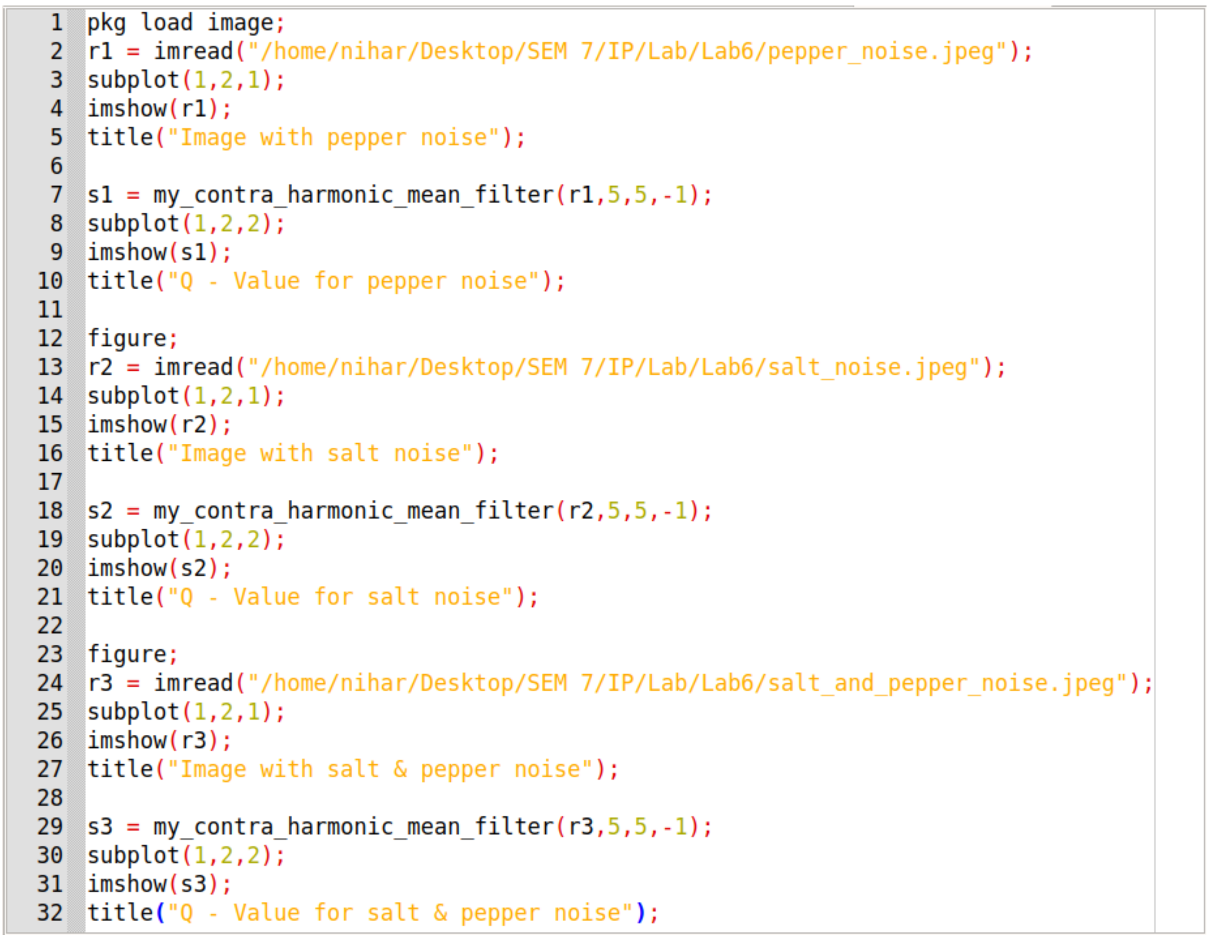
Figure 3
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Image with salt & pepper noise
Q + Value for salt & pepper noise

3. Use contra harmonic filter with Q values –ve for value for salt noise, pepper noise and salt and pepper noise. Comment on your outcome. .

→ Solution :-

- For negative value of Q it reduces salt noise and increases pepper noise.
- For salt and pepper noise image it increases pepper noise and decrease salt noise.
- For salt and pepper I passed first with negative Q value and then on that result positive Q valued contra harmonic filter is applied so noise get reduced.

Code :



```
pkg load image;
r1 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/pepper_noise.jpeg");
subplot(1,2,1);
imshow(r1);
title("Image with pepper noise");

s1 = my_contra_harmonic_mean_filter(r1,5,5,-1);
subplot(1,2,2);
imshow(s1);
title("Q - Value for pepper noise");

figure;
r2 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/salt_noise.jpeg");
subplot(1,2,1);
imshow(r2);
title("Image with salt noise");

s2 = my_contra_harmonic_mean_filter(r2,5,5,-1);
subplot(1,2,2);
imshow(s2);
title("Q - Value for salt noise");

figure;
r3 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/salt_and_pepper_noise.jpeg");
subplot(1,2,1);
imshow(r3);
title("Image with salt & pepper noise");

s3 = my_contra_harmonic_mean_filter(r3,5,5,-1);
subplot(1,2,2);
imshow(s3);
title("Q - Value for salt & pepper noise");
```

Output :

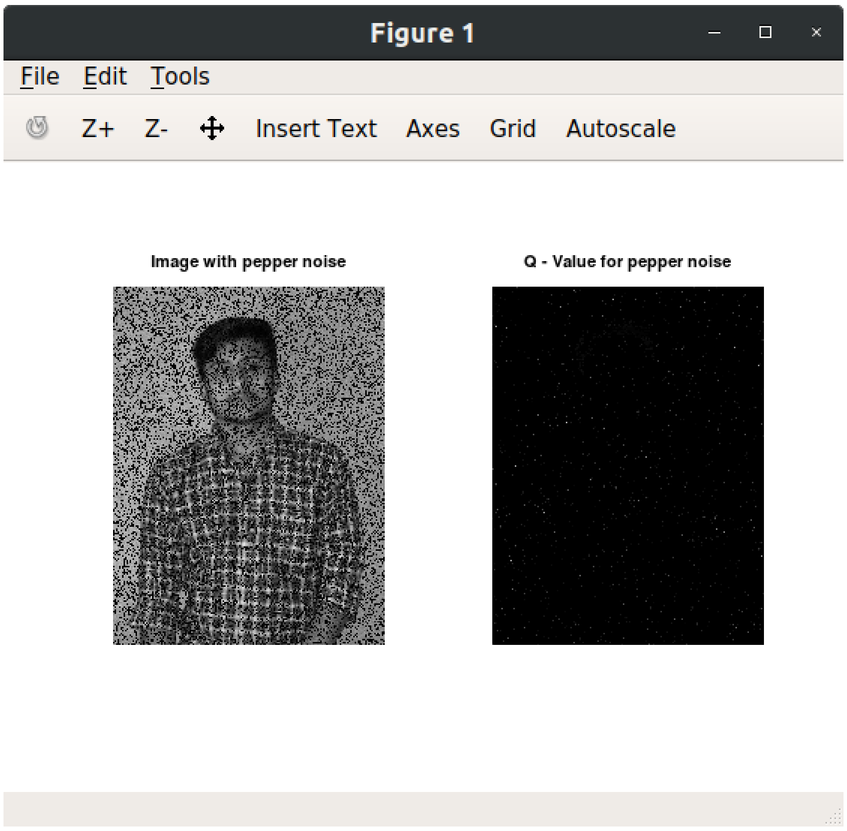


Figure 1
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Image with pepper noise
Q - Value for pepper noise

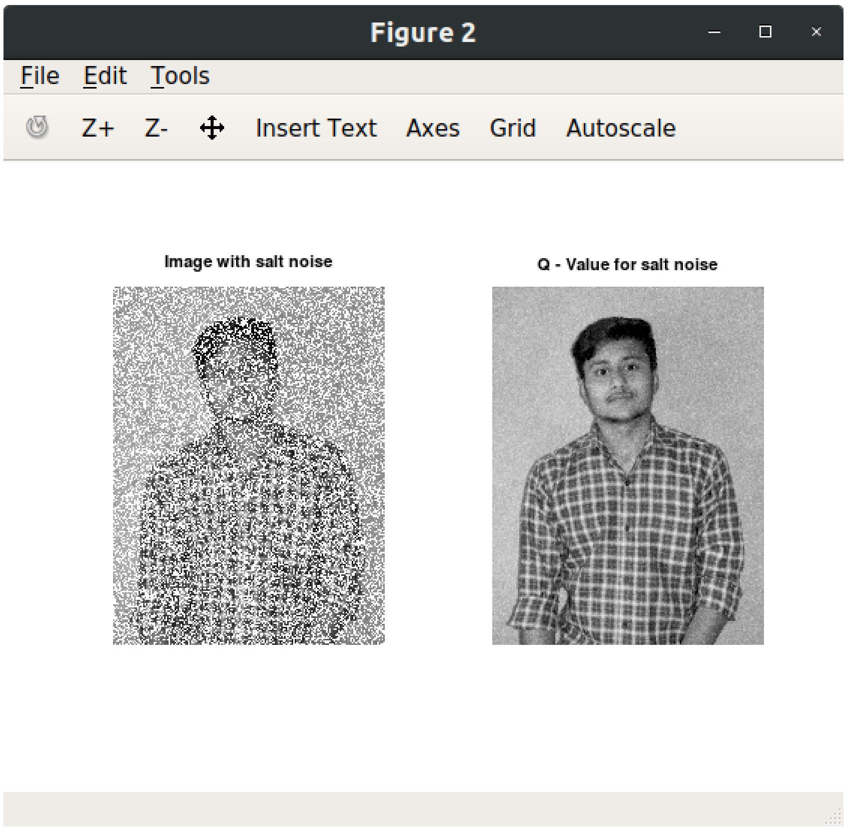


Figure 2
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Image with salt noise
Q - Value for salt noise

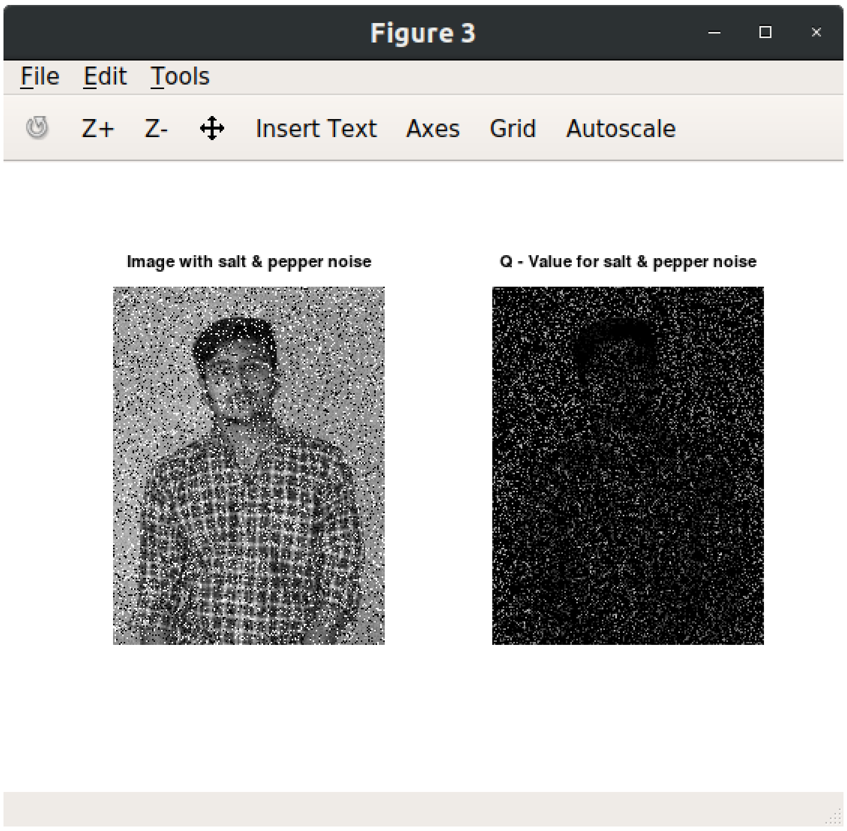


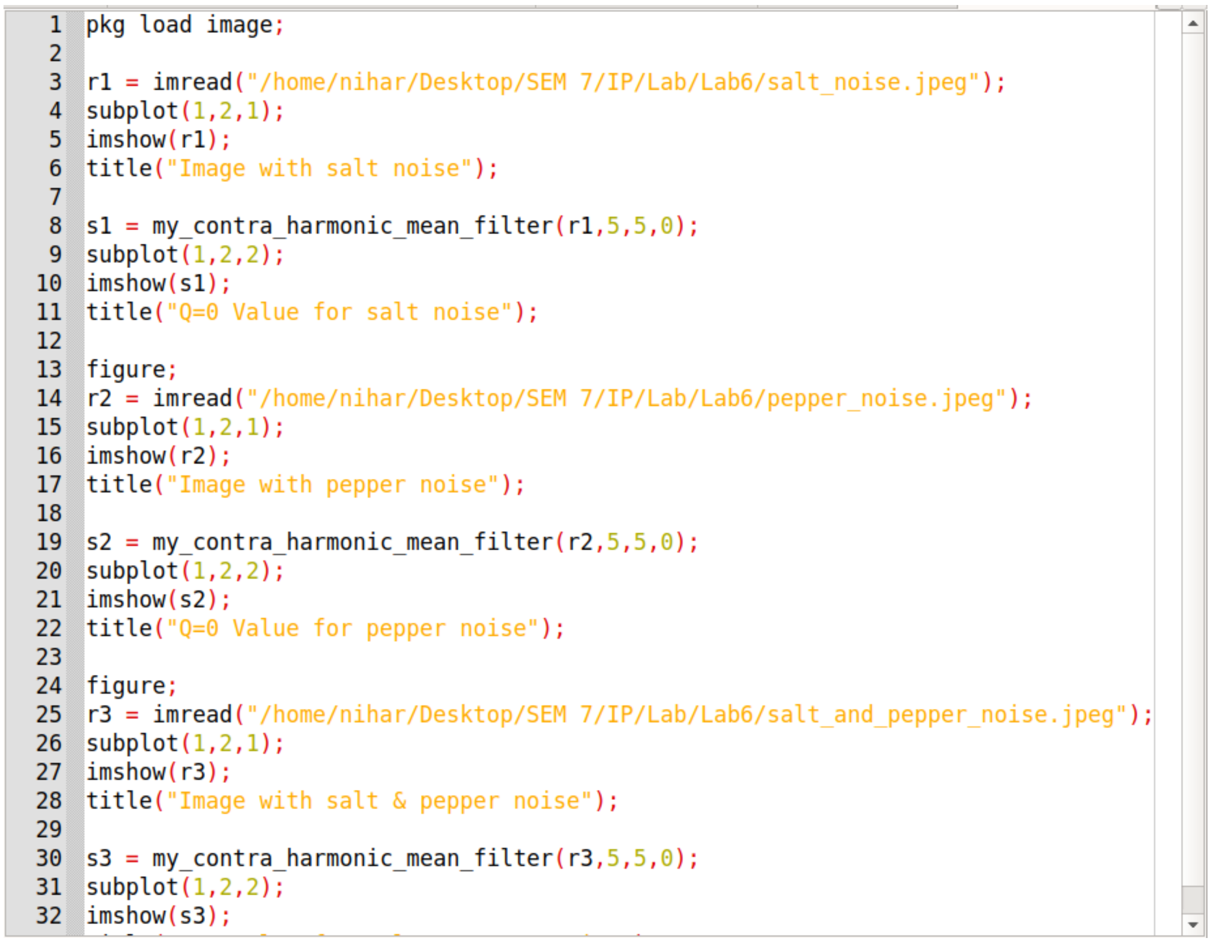
Figure 3
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Image with salt & pepper noise
Q - Value for salt & pepper noise

4. **If Q value is taken zero in contra harmonic filter it will be suitable for which kind of noise? Why? Justify your answer by implementation and results.**

→ **Solution :-**

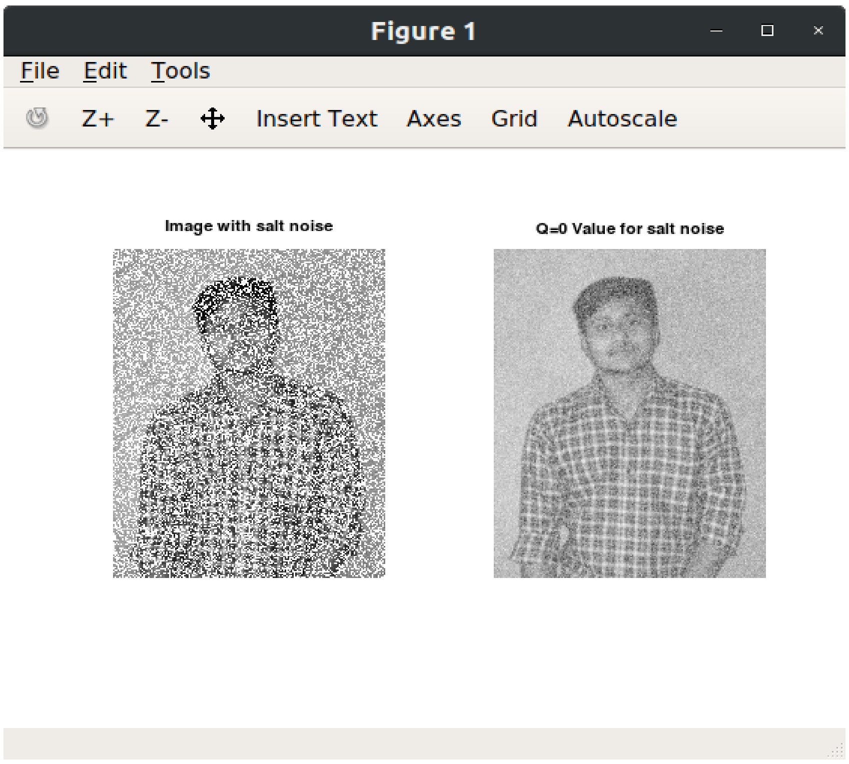
- For Q = 0 contra harmonic filter behaves like arithmetic mean filter(it's equation becomes like arithmetic mean filter).
- so it is suitable for local variation in image(I used image with gaussian noise) so via blur noise is reduced.

Code :



```
pkg load image;

r1 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/salt_noise.jpeg");
subplot(1,2,1);
imshow(r1);
title("Image with salt noise");

s1 = my_contra_harmonic_mean_filter(r1,5,5,0);
subplot(1,2,2);
imshow(s1);
title("Q=0 Value for salt noise");

figure;
r2 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/pepper_noise.jpeg");
subplot(1,2,1);
imshow(r2);
title("Image with pepper noise");

s2 = my_contra_harmonic_mean_filter(r2,5,5,0);
subplot(1,2,2);
imshow(s2);
title("Q=0 Value for pepper noise");

figure;
r3 = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/salt_and_pepper_noise.jpeg");
subplot(1,2,1);
imshow(r3);
title("Image with salt & pepper noise");

s3 = my_contra_harmonic_mean_filter(r3,5,5,0);
subplot(1,2,2);
imshow(s3);
```

Output :



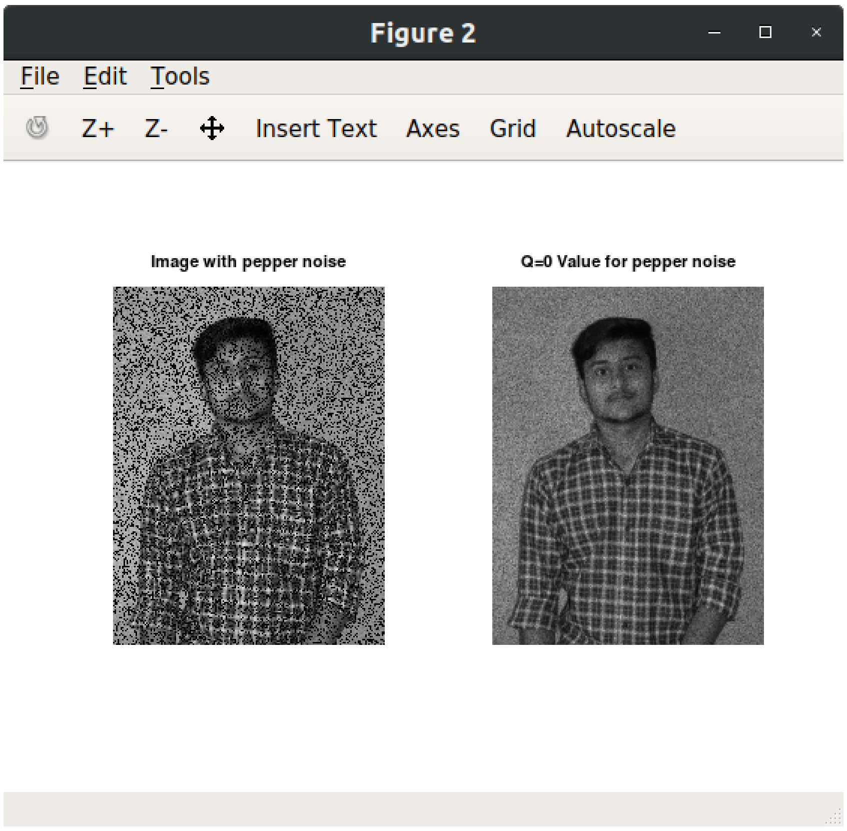


Figure 2
File
Edit
Tools
Z+
Z-
Insert Text
Axes
Grid
Autoscale
Image with pepper noise
Q=0 Value for pepper noise

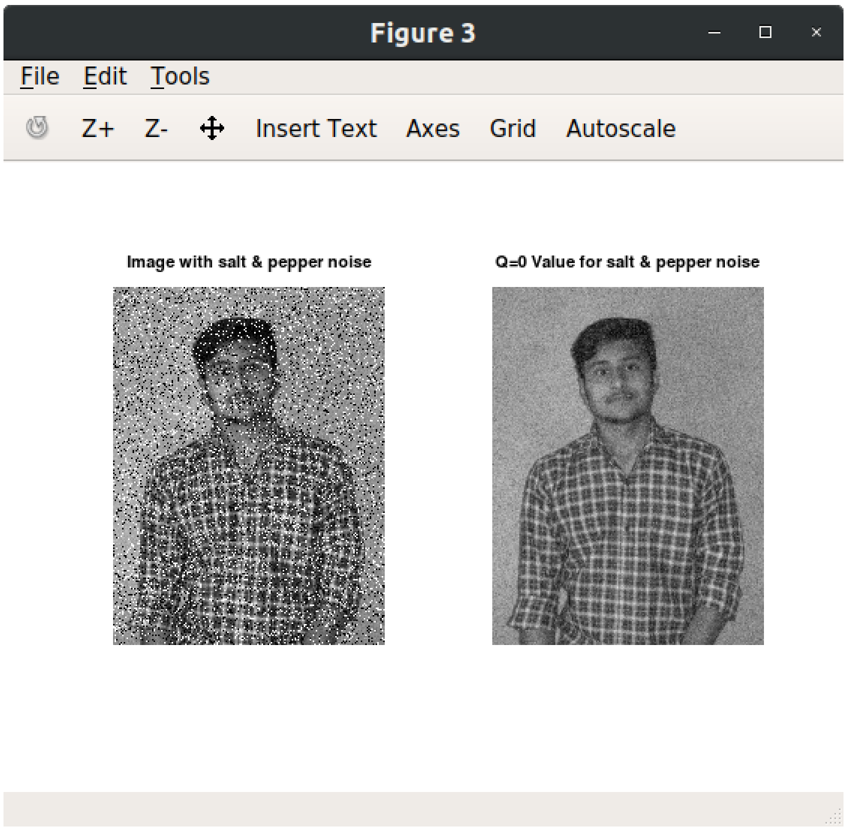


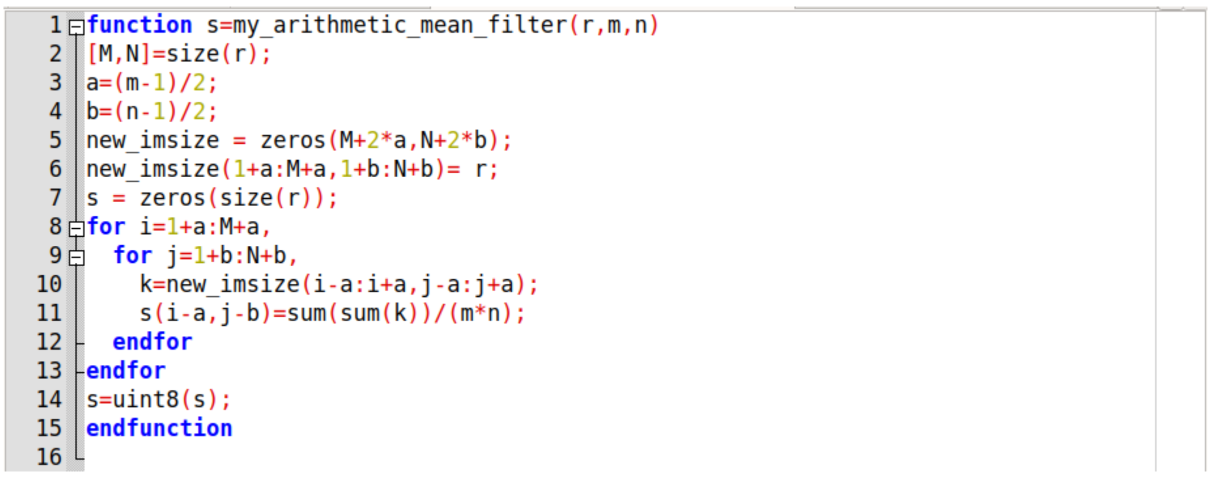
Figure 3
File
Edit
Tools
Z+
Z-
Insert Text
Axes
Grid
Autoscale
Image with salt & pepper noise
Q=0 Value for salt & pepper noise

5. **Apply arithmetic mean filter and geometric mean filter of various size to noisy image. Compare geometric mean filter and arithmetic mean filter in terms of blurring.**
   Original image contains gaussian noise.
   For filter size = 3(left image applied applied arithmetic filter and right image geometric filter).

→ **Solution :-**

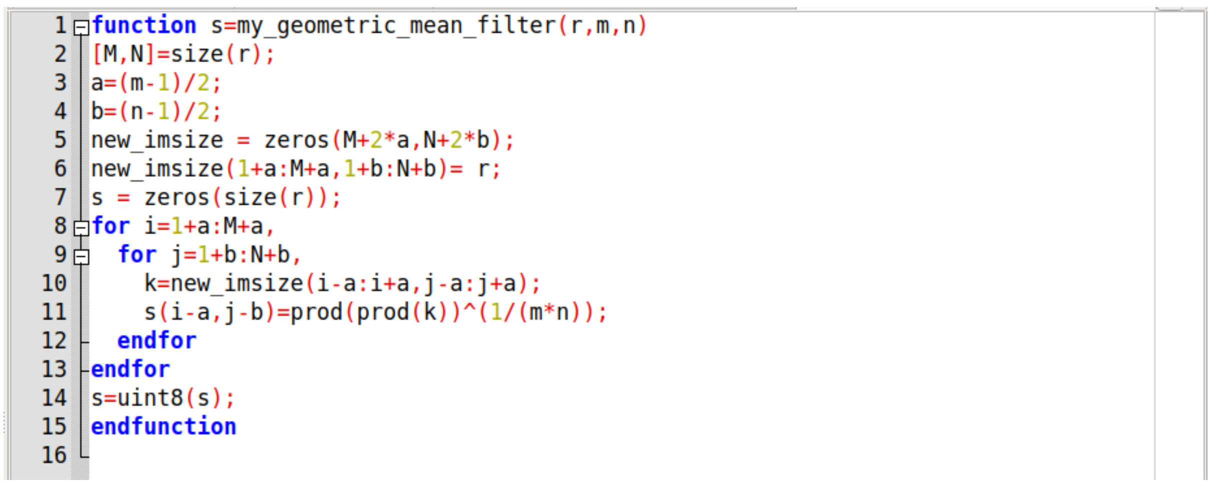
- Geometric filter achieves smoothing comparable to arithmetic filter and but it tends to loss less image detail in process.

Function for Arithmetic Mean Filter :



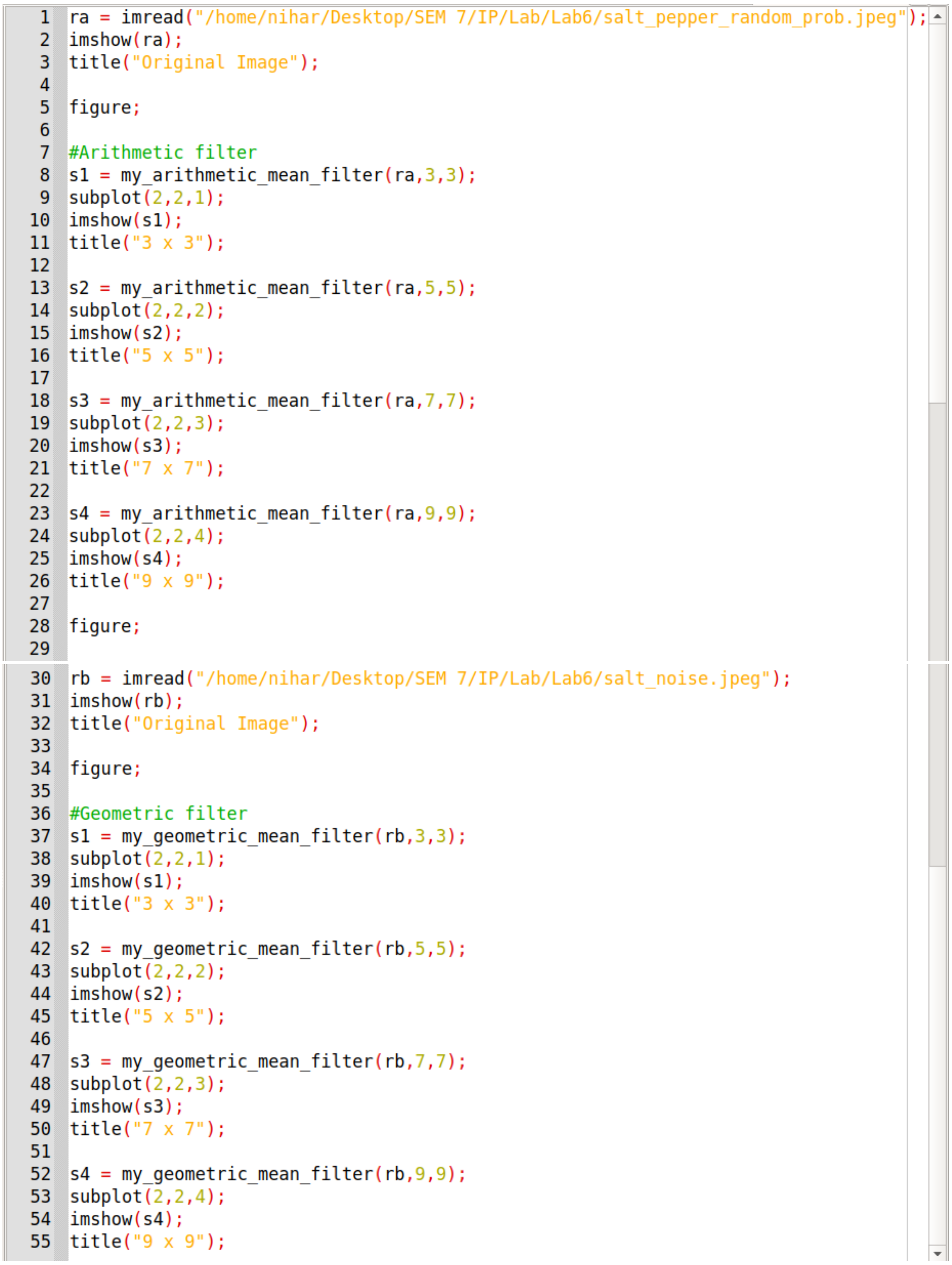
```
function s=my_arithmetic_mean_filter(r,m,n)
[M,N]=size(r);
a=(m-1)/2;
b=(n-1)/2;
new_imsize = zeros(M+2*a,N+2*b);
new_imsize(1+a:M+a,1+b:N+b)= r;
s = zeros(size(r));
for i=1+a:M+a,
  for j=1+b:N+b,
    k=new_imsize(i-a:i+a,j-a:j+a);
    s(i-a,j-b)=sum(sum(k))/(m*n);
  endfor
endfor
s=uint8(s);
endfunction
```

Function for Geometric Mean Filter :



```
function s=my_geometric_mean_filter(r,m,n)
[M,N]=size(r);
a=(m-1)/2;
b=(n-1)/2;
new_imsize = zeros(M+2*a,N+2*b);
new_imsize(1+a:M+a,1+b:N+b)= r;
s = zeros(size(r));
for i=1+a:M+a,
  for j=1+b:N+b,
    k=new_imsize(i-a:i+a,j-a:j+a);
    s(i-a,j-b)=prod(prod(k))^(1/(m*n));
  endfor
endfor
s=uint8(s);
endfunction
```

Code :



```
ra = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/salt_pepper_random_prob.jpeg");
imshow(ra);
title("Original Image");

figure;

#Arithmetic filter
s1 = my_arithmetic_mean_filter(ra,3,3);
subplot(2,2,1);
imshow(s1);
title("3 x 3");

s2 = my_arithmetic_mean_filter(ra,5,5);
subplot(2,2,2);
imshow(s2);
title("5 x 5");

s3 = my_arithmetic_mean_filter(ra,7,7);
subplot(2,2,3);
imshow(s3);
title("7 x 7");

s4 = my_arithmetic_mean_filter(ra,9,9);
subplot(2,2,4);
imshow(s4);
title("9 x 9");

figure;

rb = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/salt_noise.jpeg");
imshow(rb);
title("Original Image");

figure;

#Geometric filter
s1 = my_geometric_mean_filter(rb,3,3);
subplot(2,2,1);
imshow(s1);
title("3 x 3");

s2 = my_geometric_mean_filter(rb,5,5);
subplot(2,2,2);
imshow(s2);
title("5 x 5");

s3 = my_geometric_mean_filter(rb,7,7);
subplot(2,2,3);
imshow(s3);
title("7 x 7");

s4 = my_geometric_mean_filter(rb,9,9);
subplot(2,2,4);
imshow(s4);
title("9 x 9");
```

Output :

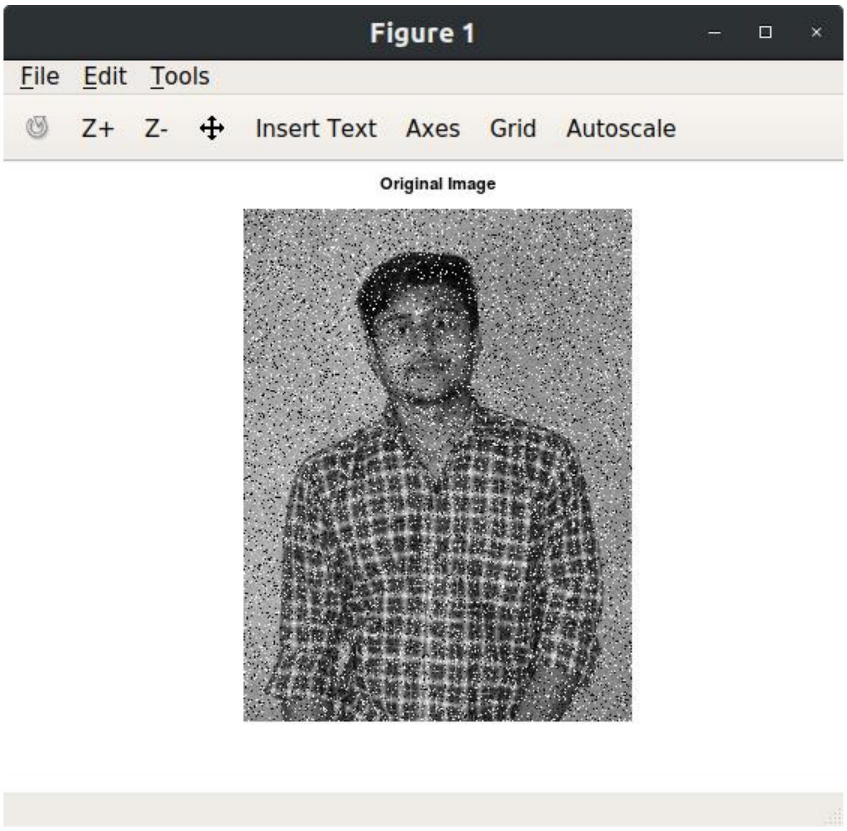


Figure 1
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Original Image

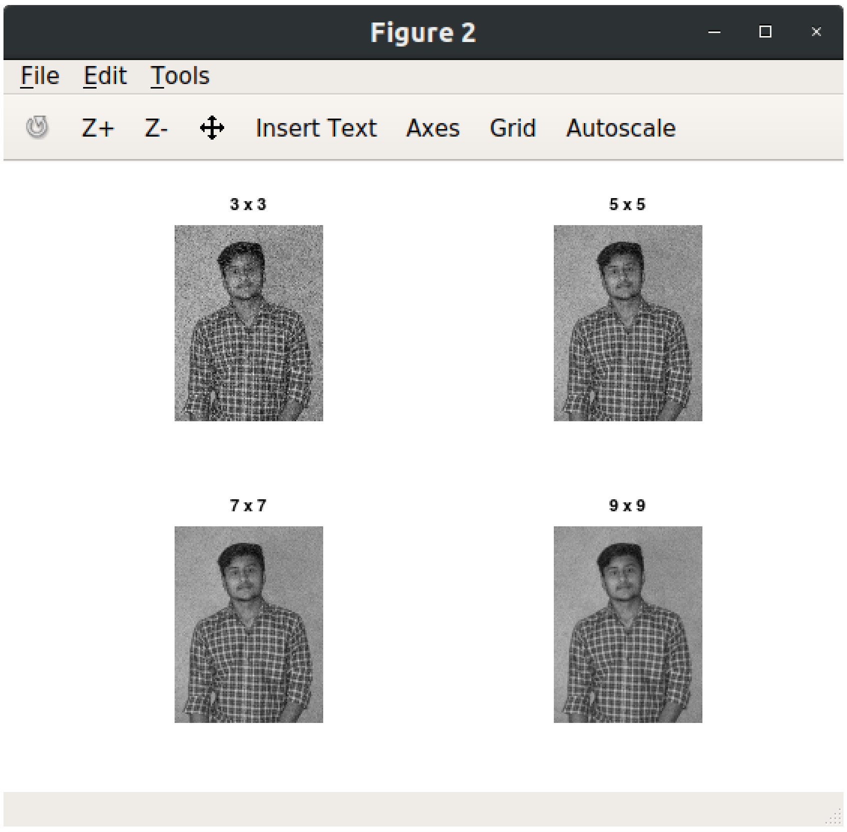


Figure 2
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
3 x 3
5 x 5
7 x 7
9 x 9

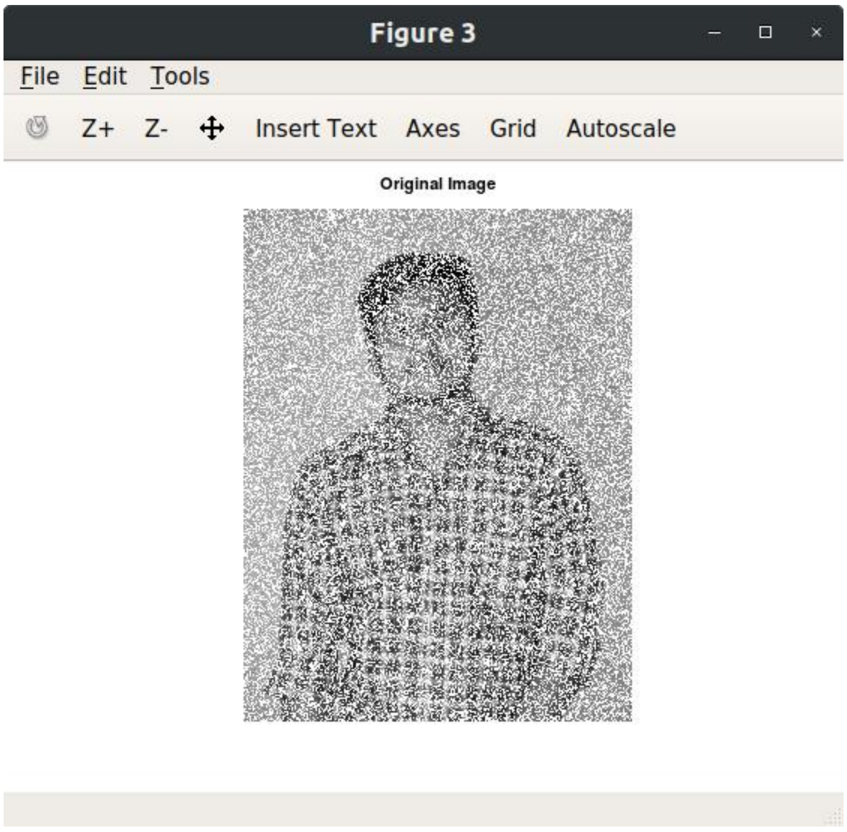


Figure 3
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Original Image

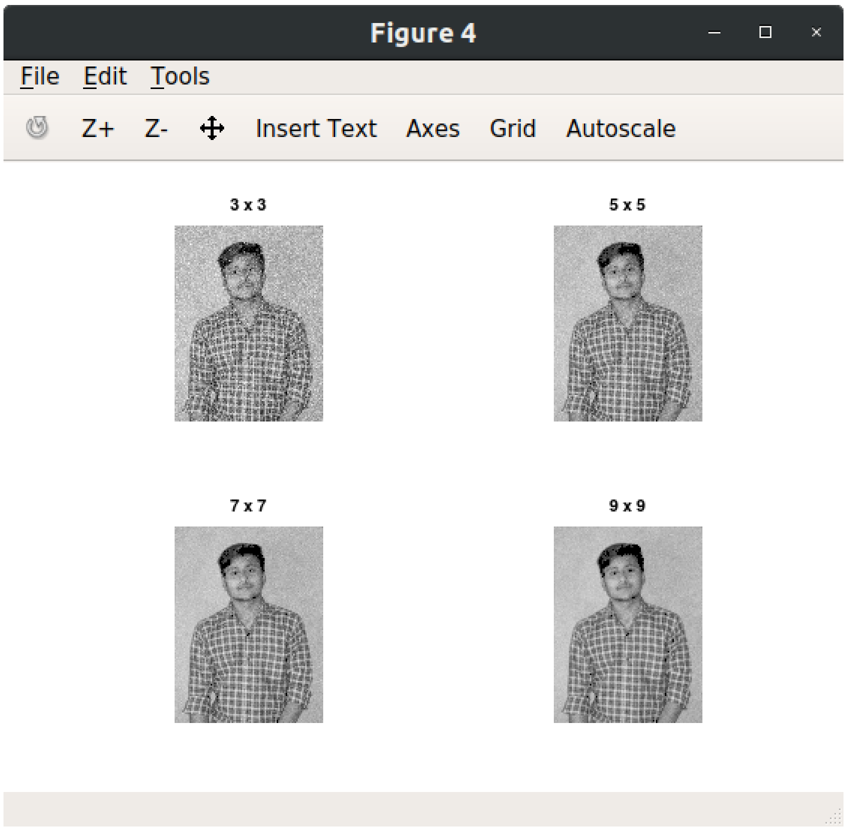


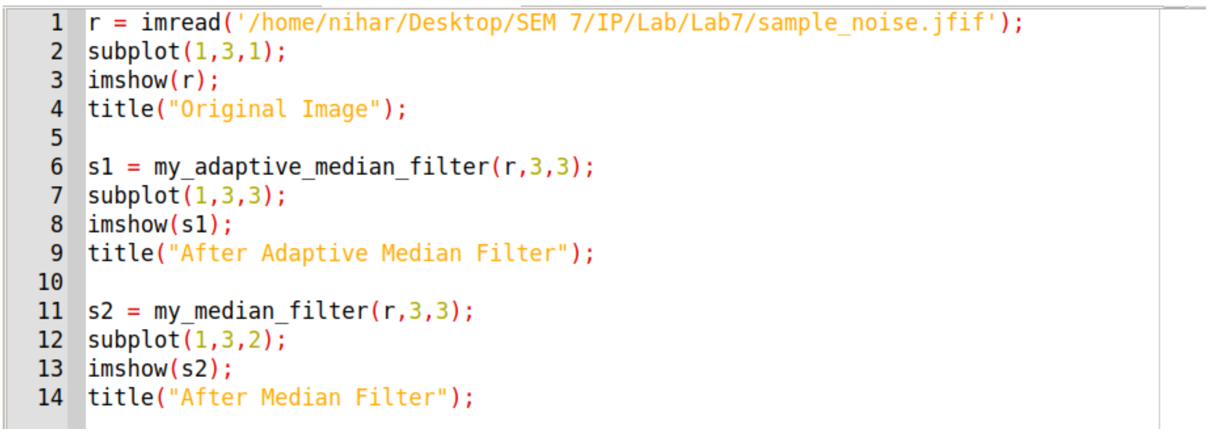
Figure 4
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
3 x 3
5 x 5
7 x 7
9 x 9

6. **Implement Adaptive median filter. compare the results with median filter and comment on the outcome. (original image(salt and pepper noise) – median filter ---- adaptive median filter).**

→ **Solution :-**

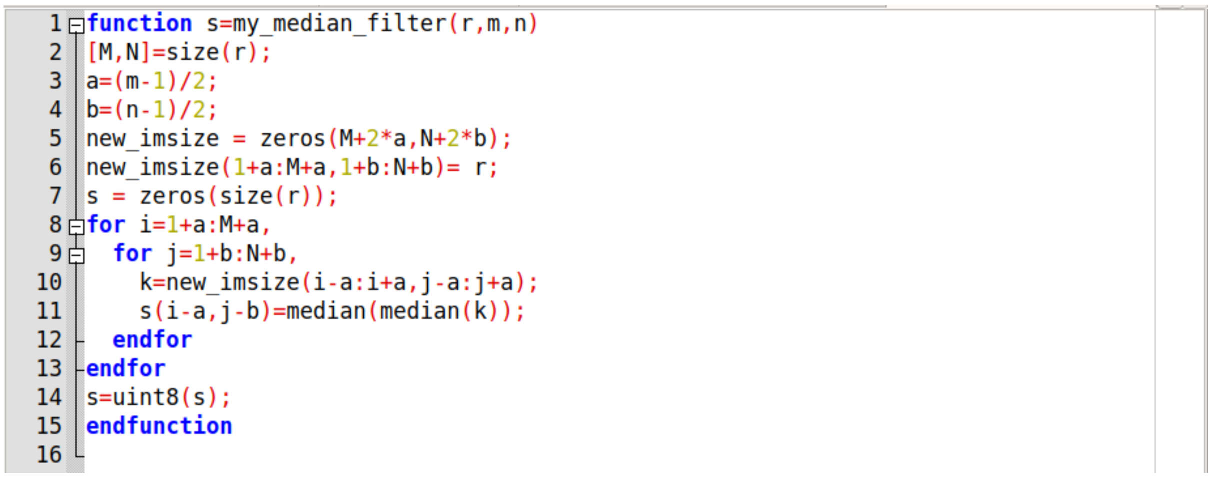
- Median filter works good for salt and pepper noise but looses details while adaptive median filter maintains details.

Code :



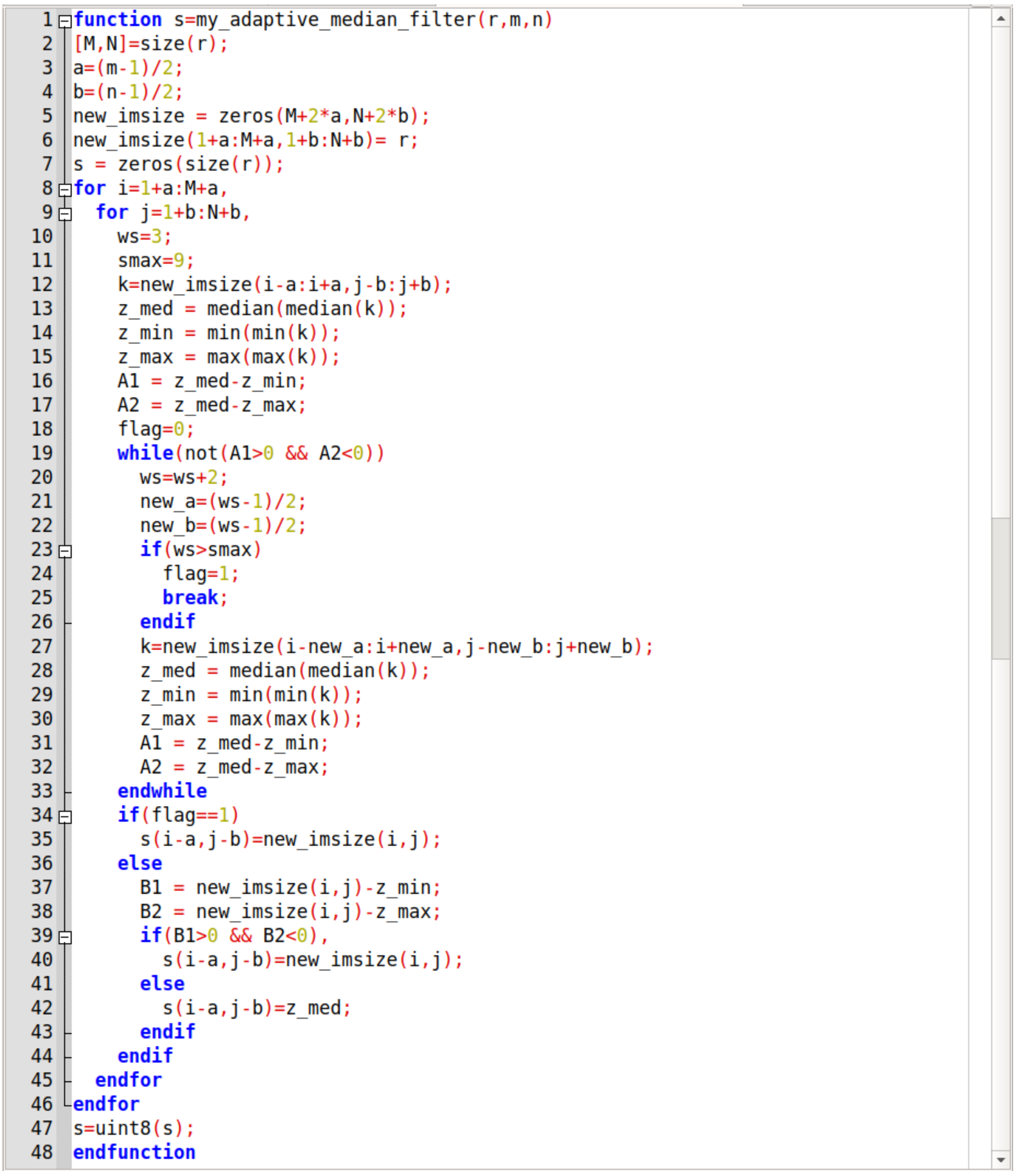
```
r = imread('/home/nihar/Desktop/SEM 7/IP/Lab/Lab7/sample_noise.jfif');
subplot(1,3,1);
imshow(r);
title("Original Image");

s1 = my_adaptive_median_filter(r,3,3);
subplot(1,3,3);
imshow(s1);
title("After Adaptive Median Filter");

s2 = my_median_filter(r,3,3);
subplot(1,3,2);
imshow(s2);
title("After Median Filter");
```

Function for Median Filter :



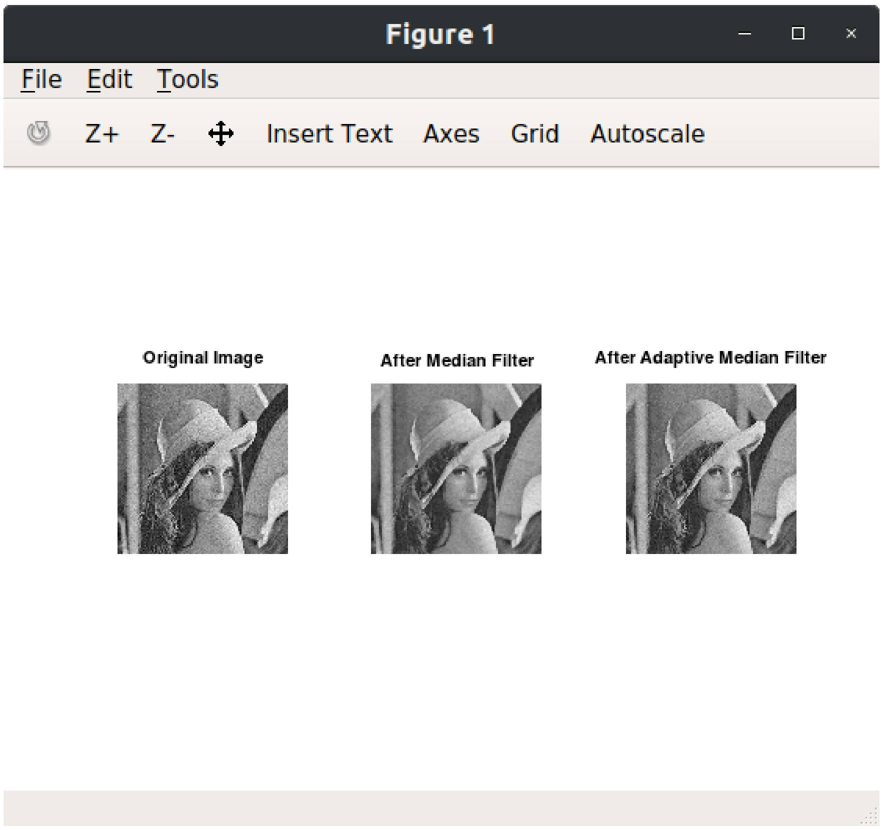
```
function s=my_median_filter(r,m,n)
[M,N]=size(r);
a=(m-1)/2;
b=(n-1)/2;
new_imsize = zeros(M+2*a,N+2*b);
new_imsize(1+a:M+a,1+b:N+b)= r;
s = zeros(size(r));
for i=1+a:M+a,
  for j=1+b:N+b,
    k=new_imsize(i-a:i+a,j-a:j+a);
    s(i-a,j-b)=median(median(k));
  endfor
endfor
s=uint8(s);
endfunction
```

Function for Adaptive Median Filter :



```
function s=my_adaptive_median_filter(r,m,n)
[M,N]=size(r);
a=(m-1)/2;
b=(n-1)/2;
new_imsize = zeros(M+2*a,N+2*b);
new_imsize(1+a:M+a,1+b:N+b)= r;
s = zeros(size(r));
for i=1+a:M+a,
 for j=1+b:N+b,
  ws=3;
  smax=9;
  k=new_imsize(i-a:i+a,j-b:j+b);
  z_med = median(median(k));
  z_min = min(min(k));
  z_max = max(max(k));
  A1 = z_med-z_min;
  A2 = z_med-z_max;
  flag=0;
  while(not(A1>0 && A2<0))
   ws=ws+2;
   new_a=(ws-1)/2;
   new_b=(ws-1)/2;
   if(ws>smax)
    flag=1;
    break;
   endif
   k=new_imsize(i-new_a:i+new_a,j-new_b:j+new_b);
   z_med = median(median(k));
   z_min = min(min(k));
   z_max = max(max(k));
   A1 = z_med-z_min;
   A2 = z_med-z_max;
  endwhile
  if(flag==1)
   s(i-a,j-b)=new_imsize(i,j);
  else
   B1 = new_imsize(i,j)-z_min;
   B2 = new_imsize(i,j)-z_max;
   if(B1>0 && B2<0),
    s(i-a,j-b)=new_imsize(i,j);
   else
    s(i-a,j-b)=z_med;
   endif
  endif
 endfor
endfor
s=uint8(s);
endfunction
```

Output :



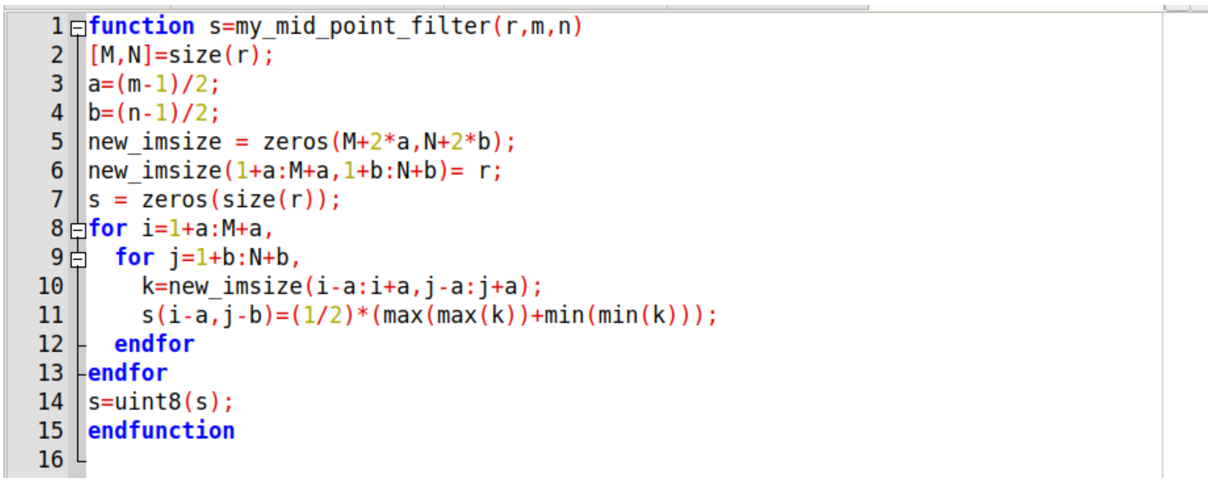
7. Implement mid point filter and apply it on your photo with Gaussian noise.

→ Solution :-

Code :

```
r = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/gaussian_noise.jpeg");
subplot(1,2,1);
imshow(r);
title("Original Image");

s = my_mid_point_filter(r,5,5);
subplot(1,2,2);
imshow(s);
title("After applying mid point Image");
```

Function for Mid Point Filter :



```
function s=my_mid_point_filter(r,m,n)
[M,N]=size(r);
a=(m-1)/2;
b=(n-1)/2;
new_imsize = zeros(M+2*a,N+2*b);
new_imsize(1+a:M+a,1+b:N+b)= r;
s = zeros(size(r));
for i=1+a:M+a,
  for j=1+b:N+b,
    k=new_imsize(i-a:i+a,j-a:j+a);
    s(i-a,j-b)=(1/2)*(max(max(k))+min(min(k)));
  endfor
endfor
s=uint8(s);
endfunction

```

Output :

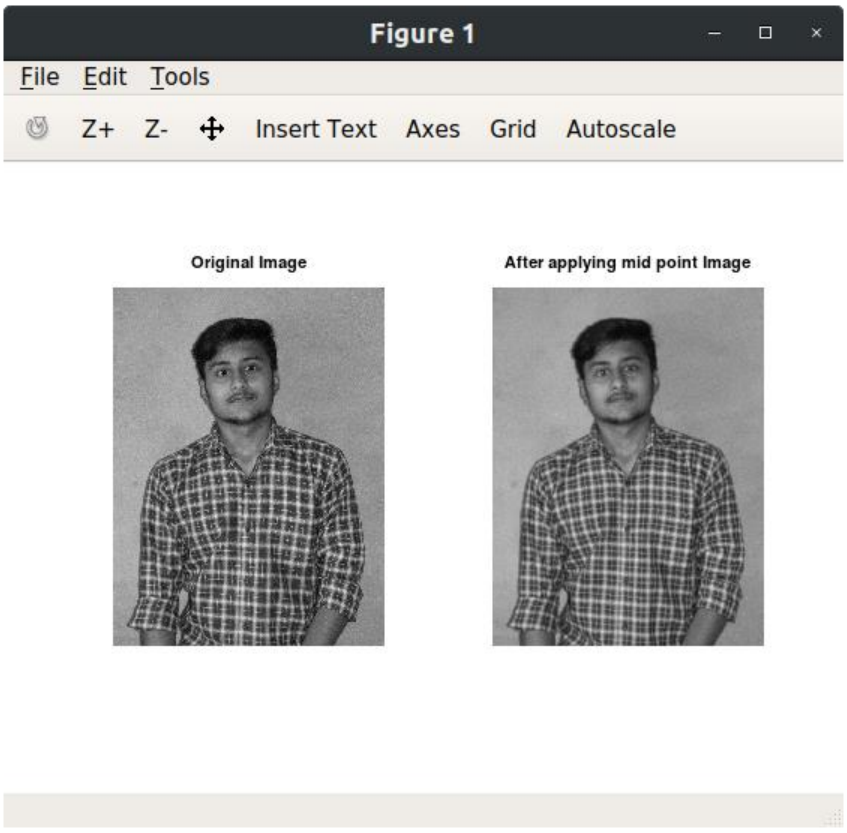


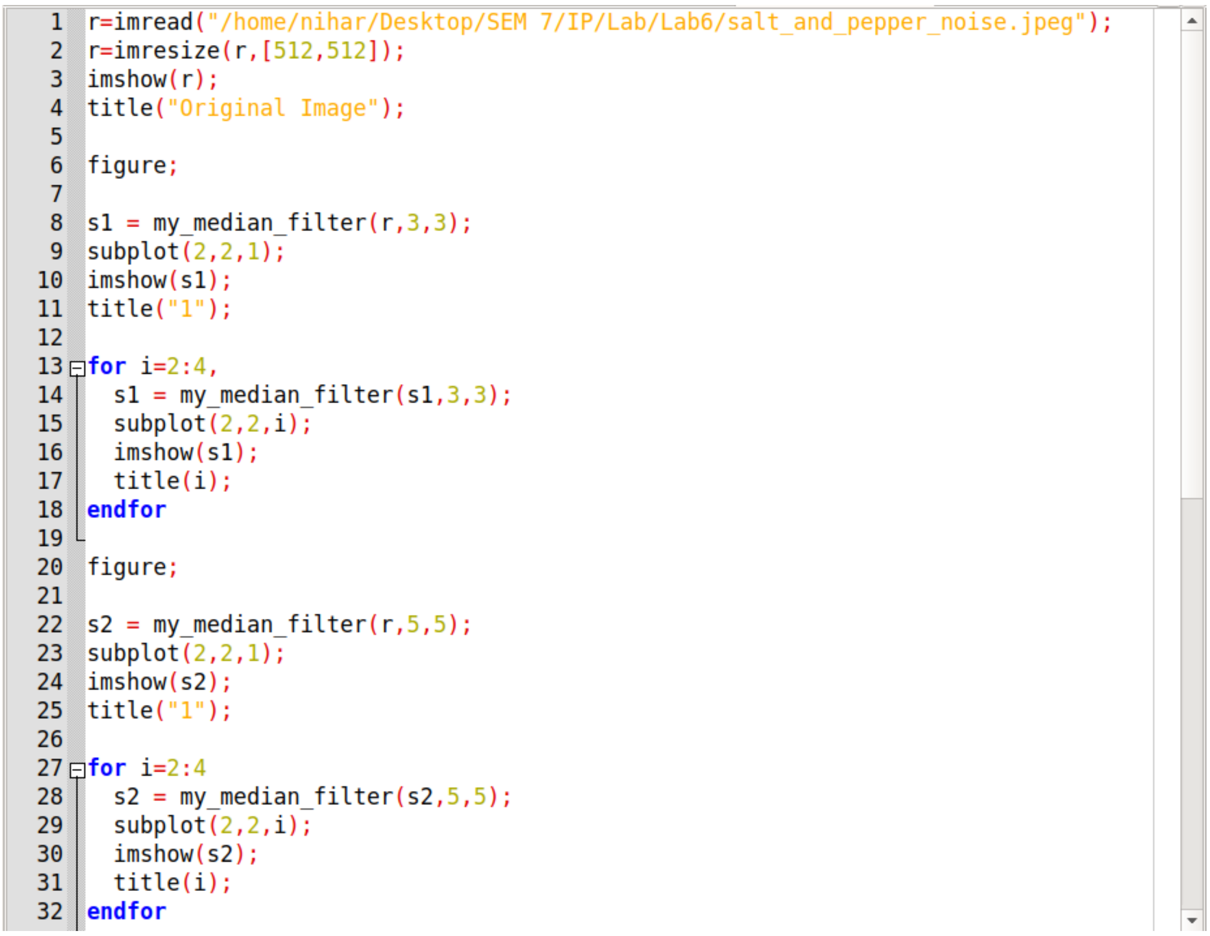
Figure 1
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Original Image
After applying mid point Image

8. **Take your photo with salt and pepper noise (probability 0.4) and Apply multiple passes of median filter and conclude about the outcome and blurring. Also apply median filter of different size (3x3, 5x5,7x7) comment on the outcome.**

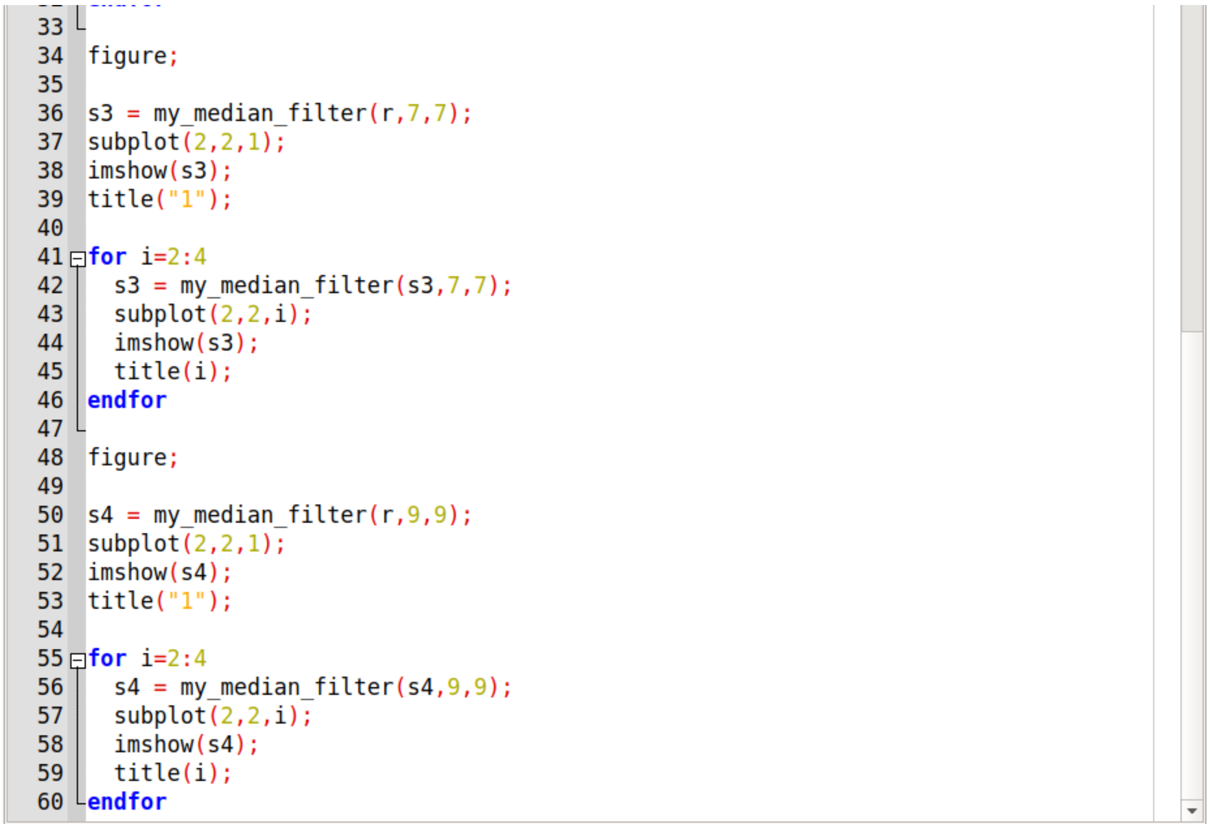
→ **Solution :-**

- Multiple passes of median filter helps for reducing salt and pepper noise. In each iteration it will reduce noise.
- More the size of median filter more the salt and pepper noise is reduced.

Code :

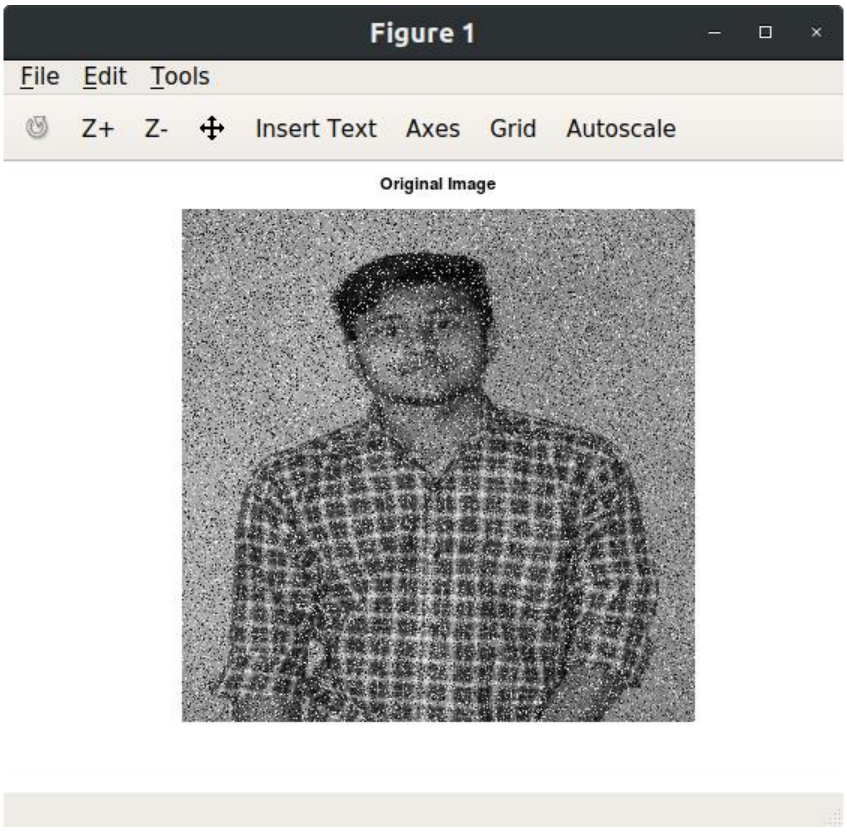


```
r=imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/salt_and_pepper_noise.jpeg");
r=imresize(r,[512,512]);
imshow(r);
title("Original Image");

figure;

s1 = my_median_filter(r,3,3);
subplot(2,2,1);
imshow(s1);
title("1");

for i=2:4,
  s1 = my_median_filter(s1,3,3);
  subplot(2,2,i);
  imshow(s1);
  title(i);
endfor

figure;

s2 = my_median_filter(r,5,5);
subplot(2,2,1);
imshow(s2);
title("1");

for i=2:4
  s2 = my_median_filter(s2,5,5);
  subplot(2,2,i);
  imshow(s2);
  title(i);
endfor
```



```
figure;

s3 = my_median_filter(r,7,7);
subplot(2,2,1);
imshow(s3);
title("1");

for i=2:4
  s3 = my_median_filter(s3,7,7);
  subplot(2,2,i);
  imshow(s3);
  title(i);
endfor

figure;

s4 = my_median_filter(r,9,9);
subplot(2,2,1);
imshow(s4);
title("1");

for i=2:4
  s4 = my_median_filter(s4,9,9);
  subplot(2,2,i);
  imshow(s4);
  title(i);
endfor
```

Output :



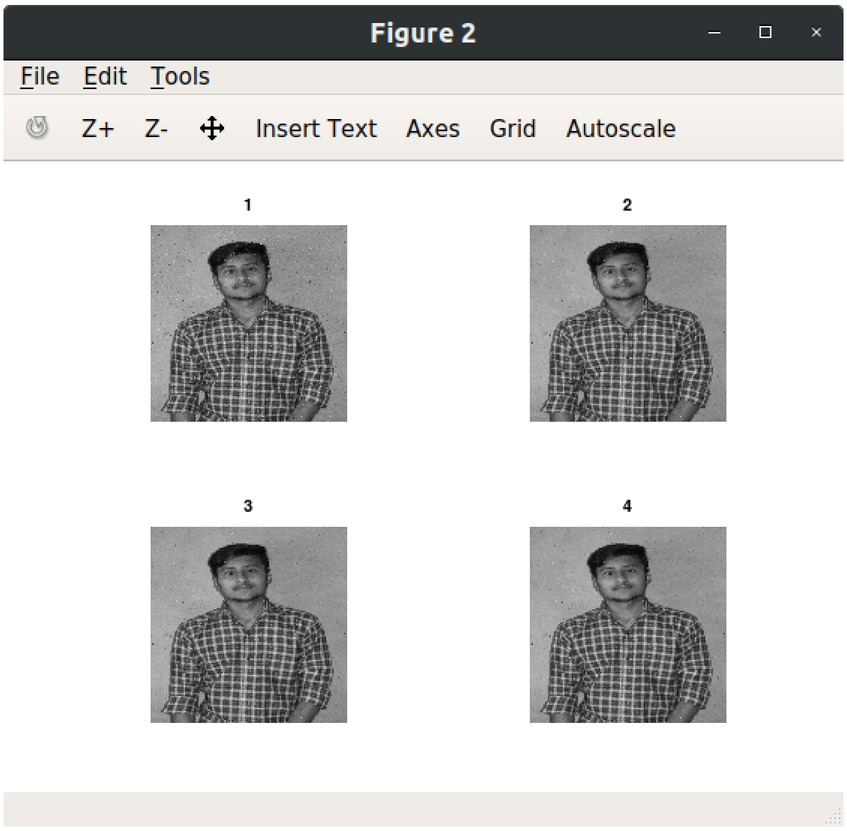


Figure 2
File
Edit
Tools
Z+
Z-
Insert Text
Axes
Grid
Autoscale
1
2
3
4

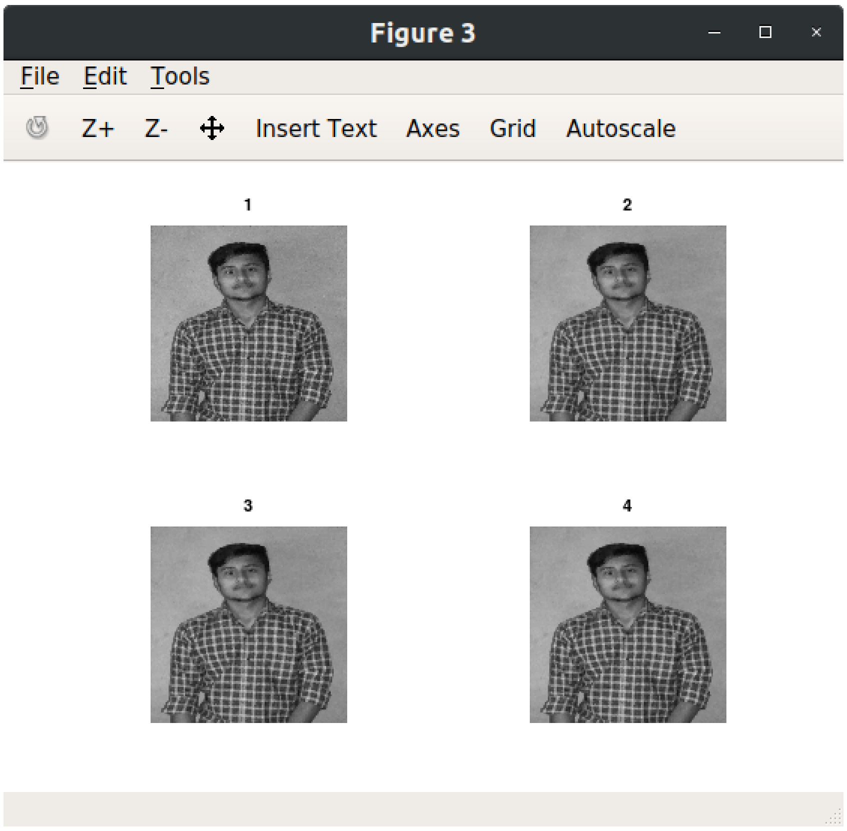


Figure 3
File
Edit
Tools
Z+
Z-
Insert Text
Axes
Grid
Autoscale
1
2
3
4

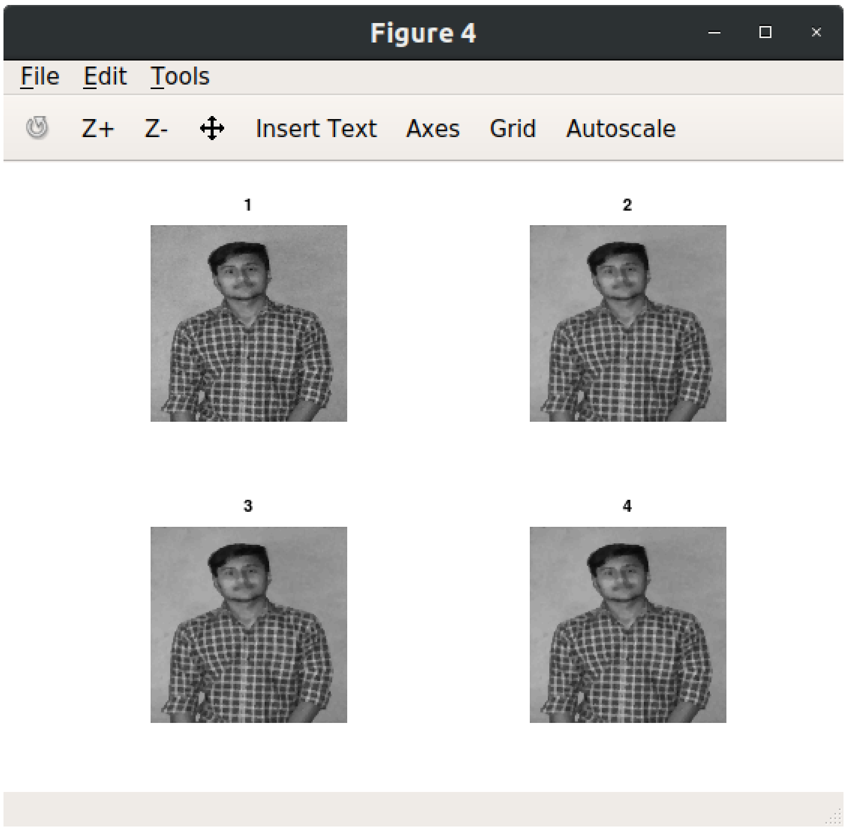


Figure 4
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
1
2
3
4

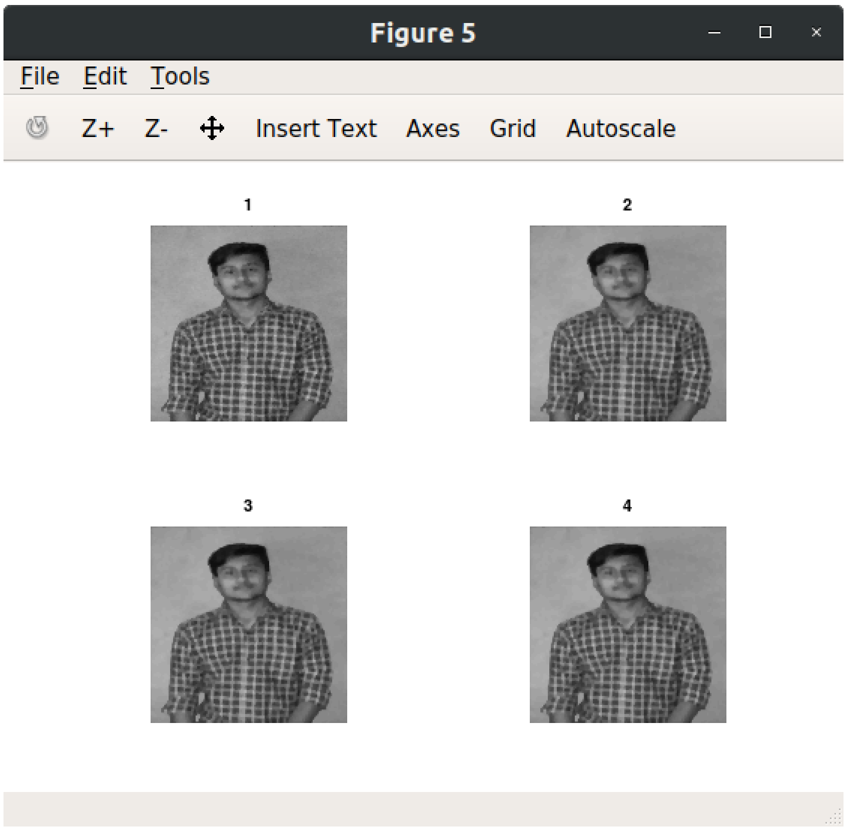


Figure 5
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
1
2
3
4

9. Take your photo with Gaussian noise and apply median filter over it. Comment on the outcome.

→ Solution :-

- Median filter is effective for bipolar and unipolar noise.
- it has good effect on gaussian noise image too.

Code :

```
r = imread("/home/nihar/Desktop/SEM 7/IP/Lab/Lab6/gaussian_noise.jpeg");
subplot(1,2,1);
imshow(r);
title("Original Image");

s = my_median_filter(r,5,5);
subplot(1,2,2);
imshow(s);
title("After Applying Median Filter");
```

Output :

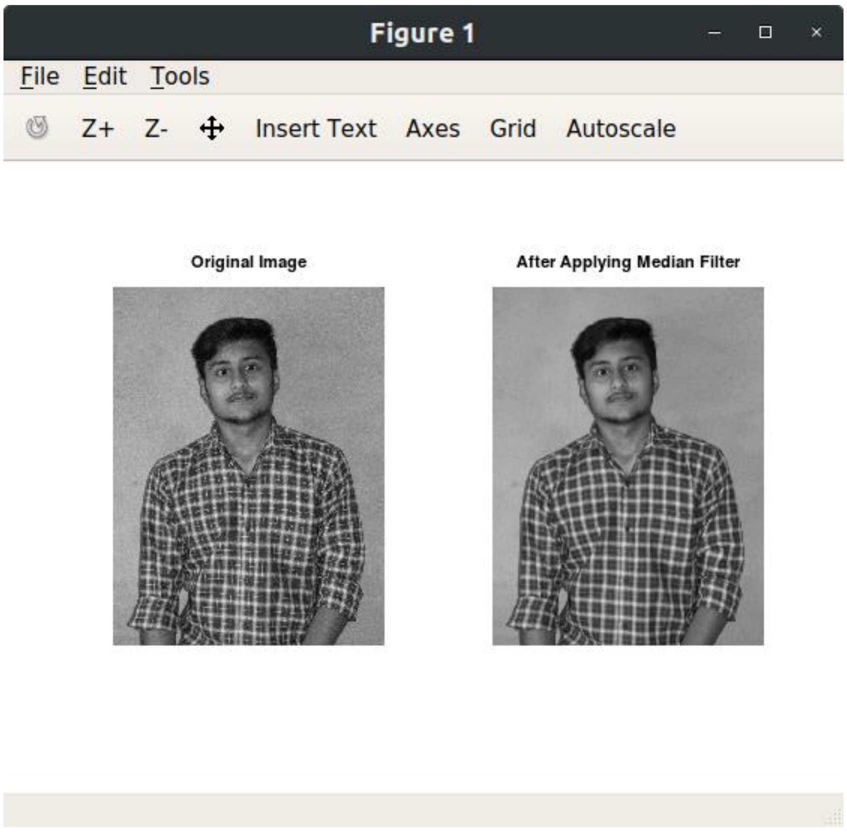
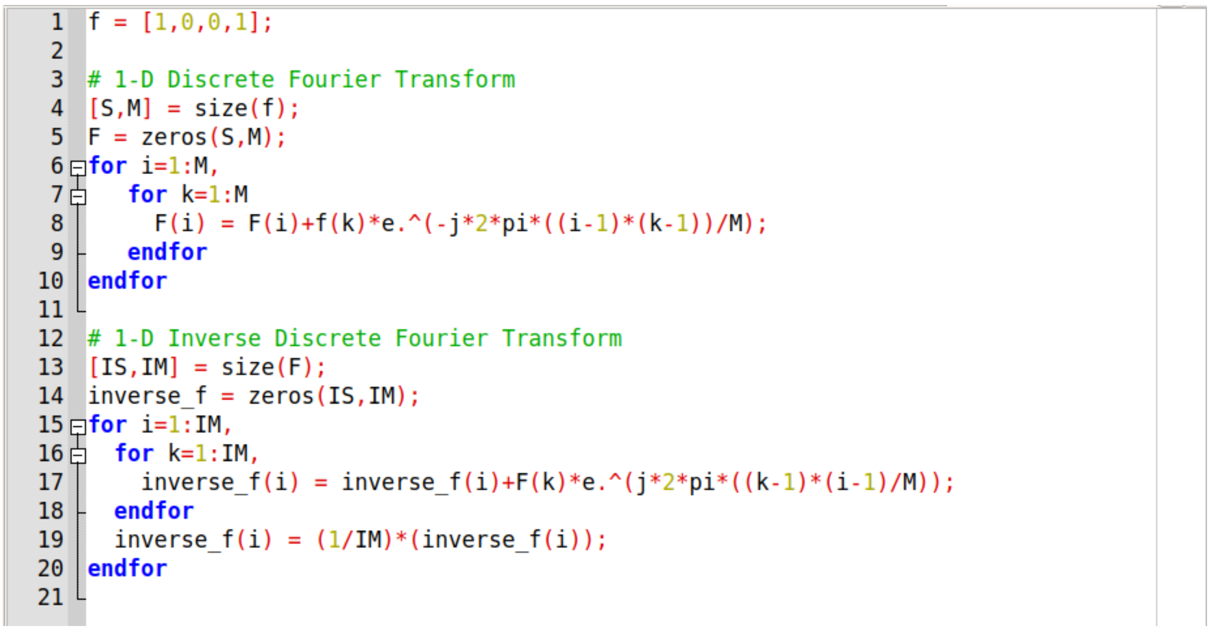


Figure 1
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
Original Image
After Applying Median Filter

10.Implement 1-D Discrete Fourier Transfom and Inverse Discrete Fourier Transform.

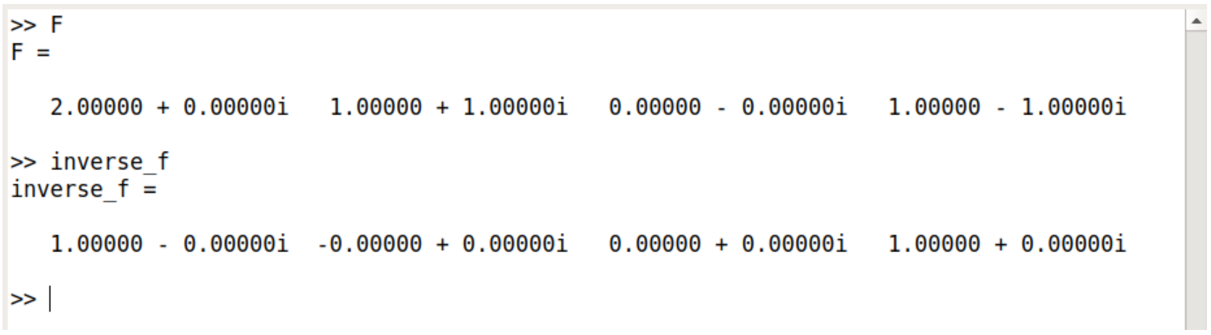
→ Solution :-

Code :



```
f = [1,0,0,1];

# 1-D Discrete Fourier Transform
[S,M] = size(f);
F = zeros(S,M);
for i=1:M,
   for k=1:M
     F(i) = F(i)+f(k)*e.^(-j*2*pi*((i-1)*(k-1))/M);
   endfor
endfor

# 1-D Inverse Discrete Fourier Transform
[IS,IM] = size(F);
inverse_f = zeros(IS,IM);
for i=1:IM,
  for k=1:IM,
    inverse_f(i) = inverse_f(i)+F(k)*e.^(j*2*pi*((k-1)*(i-1)/M));
  endfor
  inverse_f(i) = (1/IM)*(inverse_f(i));
endfor
```

Output :



```
>> F
F =

   2.00000 + 0.00000i   1.00000 + 1.00000i   0.00000 - 0.00000i   1.00000 - 1.00000i

>> inverse_f
inverse_f =

   1.00000 - 0.00000i  -0.00000 + 0.00000i   0.00000 + 0.00000i   1.00000 + 0.00000i

>> |
```