1) The two subimages shown were extracted from the top, right corners of Figs. 5.7(c) and (d), respectively. Thus, the subimage on the left is the result of using an arithmetic mean filter of size 3×3; the other subimage is the result of using a geometric mean filter of the same size.

   a) Explain why the subimage obtained with geometric mean filtering is less blurred. (*Hint:* Start your analysis by examining a 1-D step edge profile.)
   Answer: The geometric mean can be defined as:

   $$G_{mean} = \sqrt[n]{a_1 \cdot a_2 \cdots a_n} \quad .$$

   Thus, we can see that if any pixel in the surrounding region is 0, then the color value of the pixel in question will change to 0, which is black and thus preserves edge features better.

   b) Explain why the black components in the right image are thicker.
   Answer: Whenever one of the pixels in the surrounding area is 0, the geometric mean of the area will be 0, which is black. Therefore the new value of the center pixel will also be 0. With the arithmetic mean, all of the values in the surrounding area have to be 0 in order for the new center pixel to be 0. Thus, the black components in the right image, which is the geometric mean filter, are thicker.

2) Download from the class web page the image Fig5.07(b).jpg (X-Ray image corrupted by Gaussian noise).

   a) Write a computer program to implement the arithmetic mean filter of size 3×3. Apply the program to the image 'Fig5.07(b).jpg' .
   Plese see attached pages for images.

```
original_image = imread('Fig5.07(b).jpg');
original_image_double = im2double(original_image);

%Create seperate matrix to duplicate and show image
B = zeros(size(original_image_double));

%Loop through interior pixels of image
for i = 2:size(original_image_double,1) - 2
    for j = 2:size(original_image_double,2) - 2

        %Create the filter with zeros and counter
        filter = [0 0 0 0 0 0 0 0 0];
        count = 1;

        %Loop through the filter
        for x = 1:3
            for y = 1:3
                %Replace values of filter with those of the image
                filter(count) = original_image_double(i + x - 1, j + y - 1);
                count = count + 1;
            end
        end

        value = 0;
        for a = 1:9
            value = value + filter(a);
        end
        value = value / 9;
        B(i,j) = value;
    end
end
```

b) Write a computer program to implement the geometric mean filter of size 3×3. Apply the program to the image 'Fig5.07(b).jpg' .

Plese see attached pages for images.

```
original_image = imread('Fig5.07(b).jpg');
original_image_double = im2double(original_image);

%Create seperate matrix to duplicate and show image
B = zeros(size(original_image_double));

%Loop through interior pixels of image
for i = 2:size(original_image_double,1) - 2
    for j = 2:size(original_image_double,2)-2

        %Create the filter with zeros and counter
        filter = [0 0 0 0 0 0 0 0 0];
        count = 1;

        %Loop through the filter
        for x = 1:3
            for y = 1:3
                %Replace values of filter with those of the image
                filter(count) = original_image_double(i + x - 1, j + y - 1);
                count = count + 1;
            end
        end

        value = 1;
        for a = 1:9
            value = value * filter(a);
        end

        value = (value)^(1/9);

        B(i,j) = value;
    end
end
```

c) Explain your results. Evaluate the SNR (signal-to-noise-ratio) for both results in (a) and (b) (before denoising and after denoising). Note, the higher the SNR, the better the image is denoised. Let $\hat{f}$ be the denoised image, and f the clean true image. Then

$$SNR = 10 \log_{10} \frac{\sum_{x,y} (\hat{f})^2}{\sum_{x,y} (f - \hat{f})^2} \quad .$$

To evaluate the SNR before denoising, substitute $\hat{f}$ by g in the above formula.

Answer: In the images, we can see that there was not much change from the original or that there are not that many differences from each other, except for the dark regions. This can be attributed to the fact that the geometric mean filter was designed to preserved edges better than the arithmetic mean filter.
To calculate the SNR, I wrote a MATLAB script to do all of the computations. For the arithmetic mean filter, we get $SNR = 14.5366$, while for the geometric mean filter, we get $SNR = 14.3878$. These SNR levels are not that high, which explains why there was not much difference between the original and the new images.

3) Refer to the contraharmonic filter given in Eq. (5.3-6).

a) Explain why the filter is effective in eliminating pepper noise when Q is positive.

Answer: We know that pepper noise is associated with dark intensity values, so let's consider the case where the intensity of the center pixel is 0. Assume that the other $mn-1$ pixels in the neighborhood are similar and let that intensity be A. We then can get the formula that is derived from the original contraharmonic formula:

$$\hat{f}(x,y) = \frac{(mn-1)A^{Q+1}}{(mn-1)A^{Q}} \quad .$$

Thus, we can see that the value of the center pixel is set to the same as its surroundings, where Q does not affect the results. However, it will affect the result when intensity of the pixel when it is non-zero. If we consider the case where the pepper noise is set to the intensity of 1, we will see that $\hat{f}(x,y)$ will be less than A when Q is positive and approaches A as the value of Q increases.

b) Explain why the filter is effective in eliminating salt noise when Q is negative.

Answer: We know that sal noise is associated with bright intenstiy values, so let's consider the case where the intensity of the center pixel is $(L-1)$. Assume that the other $mn-1$ pixels in the neighborhood are similar and let that intensity be A. We then get the formula derived from the contraharmonic formula:

$$\hat{f}(x,y) = \frac{(mn-1)A^{Q+1} + (L-1)^{Q+1}}{(mn-1)A^{Q} + (L-1)^{Q+1}} \quad .$$

Clearly, we can see that when Q is negative, the intensity values will be positive, but very low. We can see that $(mn-1)A^{Q}$ is much greater than $(L-1)^{Q}$. Thus, the denominator reduces to essentially $(mn-1)A^{Q}$. Considering the variety of values that Q and A can take, the result will be near to A as $(L-1)^{Q+1}$ will decrease as the values of Q is decreased.

c) Explain why the filter gives poor results (such as the results shown in Fig. 5.9) when the wrong polarity is chosen for Q.

Answer: The contraharmonic mean filter was designed to eliminate sudden occurrences of dark pixels when Q is positive or eliminate sudden occurrences of bright pixels when Q is negative. If the polarity is reversed, these occurrences would be even more emphasized. Thus, the pepper noise would be emphasized when Q is negative and the salt noise will be emphasized when Q is positive.

d) Discuss the behavior of the filter when $Q = -1$.

Answer: When $Q = -1$, we can see that the contraharmonic filter will reduce to the following:

$$\hat{f}(x,y) = \frac{\sum_{s,t \in S_{x,y}} g(s,t)^{0}}{\sum_{s,t \in S_{x,y}} g(s,t)^{-1}} \tag{1}$$

$$\Rightarrow \frac{mn}{\sum_{s,t \in S_{x,y}} g(s,t)^{-1}} \quad . \tag{2}$$

Thus, we can see that the contraharmonic filter is essentially reduced to the harmonic filter, which works well for salt noise and Gaussian noise, but not for pepper noise.

e) Discuss (for positive and negative Q) the behavior of the filter in areas of constant gray levels.

Answer: Consider an area of constant intensity level A. We then can write the contraharmonic filter formula as:

$$\hat{f}(x,y) = \frac{\sum_{s,t \in S_{x,y}} g(s,t)^{Q+1}}{\sum_{s,t \in S_{x,y}} g(s,t)^{Q}} \tag{3}$$

$$\Rightarrow \frac{mnA^{Q+1}}{mnA^{Q}} \tag{4}$$

$$\Rightarrow A$$

Thus, regardless of the sign of Q, the intensity level of the center pixel in an area of constant intensity level remains the same.

4a) Download from the class web page the images 'Fig5.08(a).jpg' (X-Ray image corrupted by pepper noise) and 'Fig5.08(b).jpg' (X-Ray image corrupted by salt noise).

a) Write a computer program that will filter this image with a 3×3 contraharmonic filter of order 1.5.

Plese see attached pages for images.

```
original_image = imread('Fig5.08(a).jpg');
original_image_double = im2double(original_image);

%Create seperate matrix to duplicate and show image
B = zeros(size(original_image_double));

%Loop through internior pixels of image
for i = 2:size(original_image_double,1) - 2
    for j = 2:size(original_image_double,2)-2

        %Create the filter with zeros and counter
        filter = [0 0 0 0 0 0 0 0 0];
        count = 1;

        %Loop through the filter
        for x = 1:3
            for y = 1:3
                %Replace values of filter with those of the image
                filter(count) = original_image_double(i + x - 1, j + y - 1);
                count = count + 1;
            end
        end

        value_1 = 0;
        value_2 = 0;
        for a = 1:9
            value_1 = (value_1 + filter(a)) ^ 2.5;
            value_2 = (value_2 + filter(a)) ^ 1.5;
        end

        final_value = value_1 / value_2;

        B(i,j) = final_value;

    end
end
```

b) Write a computer program that will filter this image with a 3×3 contraharmonic filter of order -1.5.

Plese see attached pages for images.

```
original_image = imread('Fig5.08(b).jpg');
original_image_double = im2double(original_image);

%Create seperate matrix to duplicate and show image
B = zeros(size(original_image_double));

%Loop through internior pixels of image
for i = 2:size(original_image_double,1) - 2
    for j = 2:size(original_image_double,2)-2

        %Create the filter with zeros and counter
        filter = [0 0 0 0 0 0 0 0 0];
        count = 1;

        %Loop through the filter
        for x = 1:3
            for y = 1:3
                %Replace values of filter with those of the image
                filter(count) = original_image_double(i + x - 1, j + y - 1);
                count = count + 1;
            end
        end

        value_1 = 0;
        value_2 = 0;
        for a = 1:9
            value_1 = (value_1 + filter(a)) ^ 0.5;
            value_2 = (value_2 + filter(a)) ^ (-1.5);
        end

        final_value = value_1 / value_2;

        B(i,j) = final_value;

    end
end
```

5a) Recall the definition of the convolution $f \star g(x, y)$ in continuous variables and two dimensions.

Answer: The definition of the convolution in continuous variables and two dimensions is:

$$f \star g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(m, n) g(x - m, y - n) dx dy \quad .$$

5b) Show that $\nabla^2(f * g) = f * (\nabla^2 g)$, where ∇^2 denotes the Laplace operator in (x, y).

Answer: From the definition of convolution, we can write the above formula as:

$$f \star g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(m,n)f(x-m,y-n)dxdy \quad .$$

We then can write the following:

$$\nabla^2(f \star g) = \frac{\partial}{\partial x^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(m,n)f(x-m,y-n)dxdy + \frac{\partial}{\partial y^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(m,n)f(x-m,y-n)dxdy \tag{5}$$

$$\Rightarrow \frac{\partial}{\partial x^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(m,n)g(x-m,y-n)dxdy + \frac{\partial}{\partial y^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(m,n)g(x-m,y-n)dxdy \tag{6}$$

$$\Rightarrow \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left(g\left(\frac{\partial}{\partial x^2}\right) + g\left(\frac{\partial}{\partial x^2}\right)\right)(x-m,y-n)f(m,n) \tag{7}$$

$$\Rightarrow \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left(g\left(\frac{\partial}{\partial x^2}\right) + g\left(\frac{\partial}{\partial x^2}\right)\right)(m,n)f(x-m,y-n) \tag{8}$$

$$\Rightarrow f \star \left(g\left(\frac{\partial}{\partial x^2}\right) + g\left(\frac{\partial}{\partial x^2}\right)\right) \tag{9}$$

$$\Rightarrow f \star (\nabla^2 g) \tag{10}$$