## CSE585/EE555:  Digital Image Processing II

## Computer Project # 3

## Nonlinear Filtering and Anisotropic Diffusion

#### Mandar Parab, Amogh S. Adishesha, Lyuzhou Zhuang

#### Date: 3/26/2017

* + 1. Objectives

Implement non-linear filtering and anisotropic diffusion in MATLAB. Observe effects of changing parameter values and number of iterations have on output images.

* + 1. Methods

Problem 1 Nonlinear Filtering

We implement 4 nonlinear filters in this problem: 7x7 median, 7x7 alpha-trimmed mean (α=0.25), 7x7 sigma filter (σ=20), 7x7 symmetric nearest-neighbor mean, and apply them successively to image ‘cwheelnoise’ and ‘disk’. Give filter results after 1st and 5th iterations.

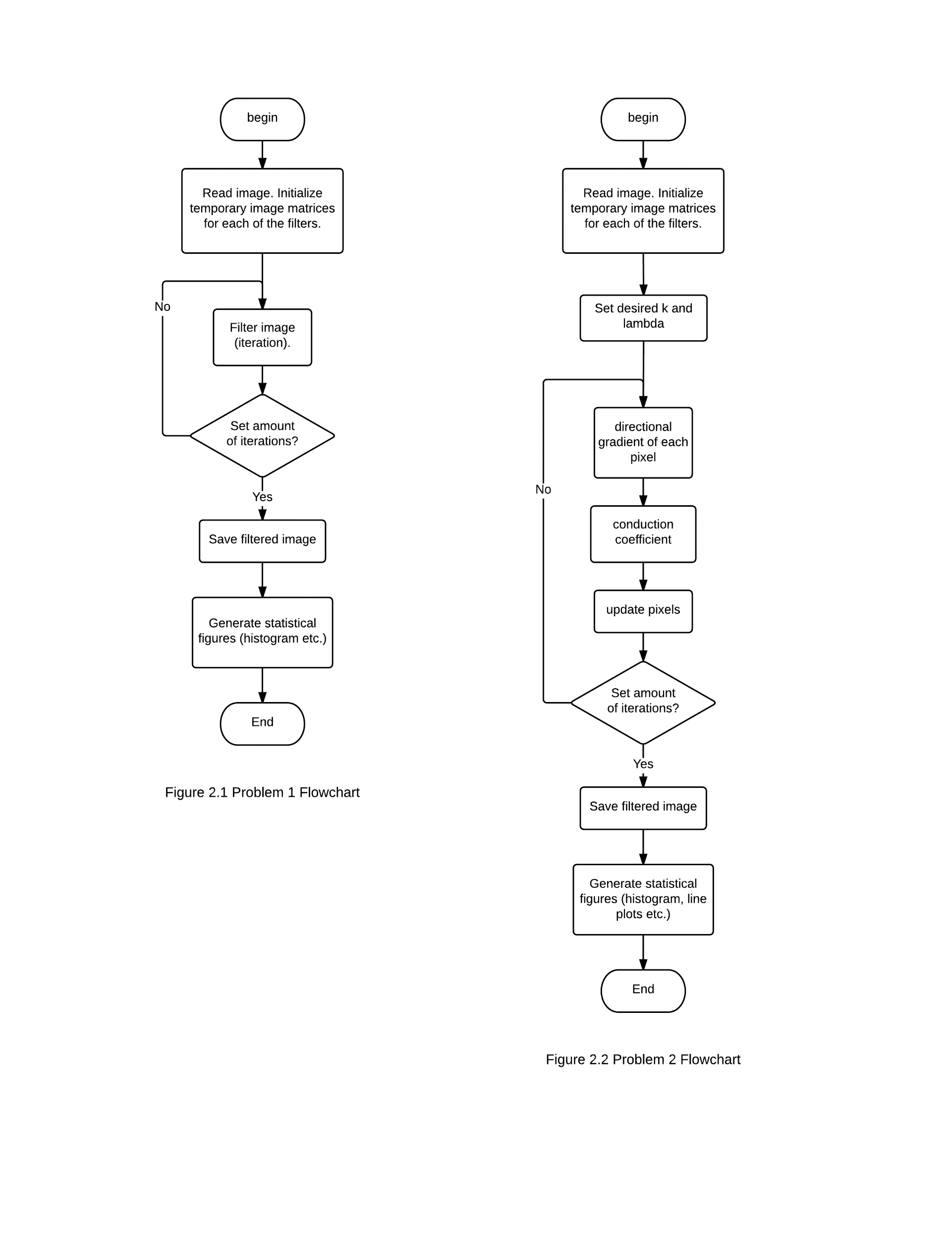
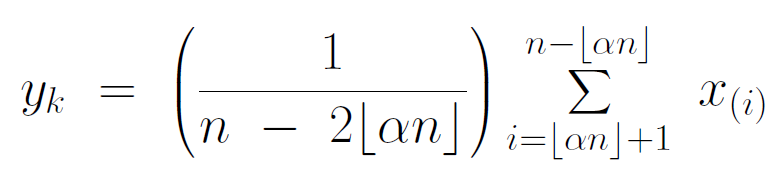
Figure 2.1 is the MATLAB processing flow for problem 1. (Note: Run main.m and it will automatically call subfuntions to perform any operations required.)

Load image(s)

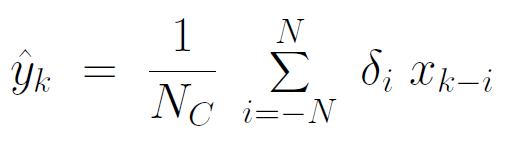
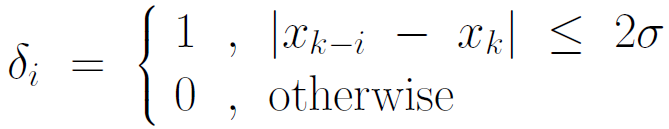
7x7 median filtering

We use build-in matlab function medfilt2 to perform median filtering on the image. It’s returning to the pixel of interest (center of filter) the 25th greatest pixel value of all 49 pixels covered by the filter.

7x7 alpha-trimmed mean filtering (α=0.25)

We wrote function alpha() to perform alpha-trimmed mean filtering based on the formula given on L12-4 where we returns to pixel of interest mean of middle half of pixels this filter coveres.

7x7 sigma filtering (σ = 20)

We wrote function sigmamain() to perform sigma filtering on the image based on the formula given in L12-7  

7x7 symmetric nearest-neighbor mean filtering

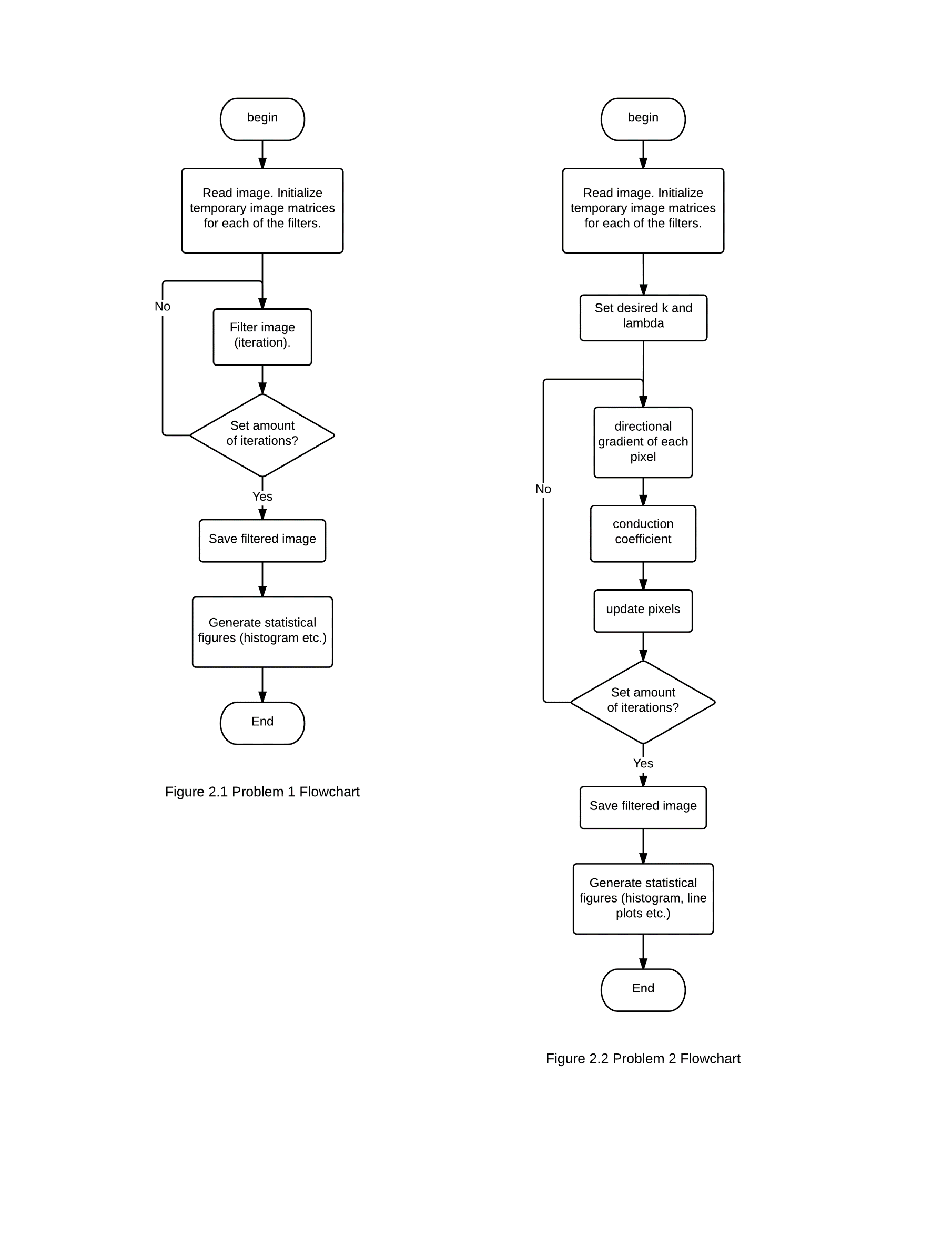
We wrote snnmain() to perform snn filtering on image based on course note L12-18. For each pair of points symmetrically opposite each other, select the point most similar to xk,l (point of interest), then average together selected points and assign the value to point of interest.

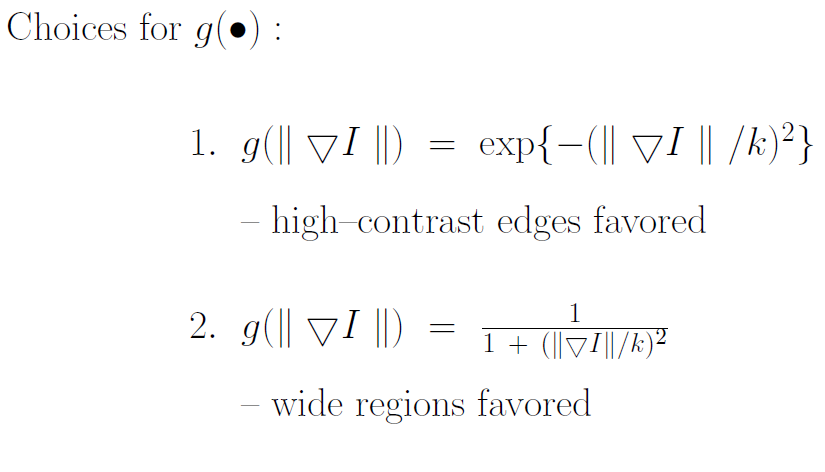
Apply the above 4 filters to the image for the amount of times required (number of iterations). Output images results and histograms if needed

Problem 2 Anisotropic Diffusion for image filtering

In this problem, we apply anisotropic diffusion to blurred images with 2 different diffusion algorithms.

Figure 2.2 is the MATLAB processing flow for problem 2. (Note: Run mainpart2.m and it will automatically call subfunctions to perform any operations required.)

1. Load Image(s)
2. Apply exponential form of g(•) and inverse quadratic form of g(•) successively, filter the original image for certain amount of iterations, finally show the results. Formulas are given in L14-12, L14-13, L14-14.

Choices for g() :

1. Segmenting out one of the wheel components using grayvalue threshold.
   * 1. Results

Problem 1

The results for problem 1 are shown below.

Figure 3.1.1 to Figure 3.1.8 shows cwheelnoise image fitered by the 4 filters mentioned above after 1 or 5 iterations.

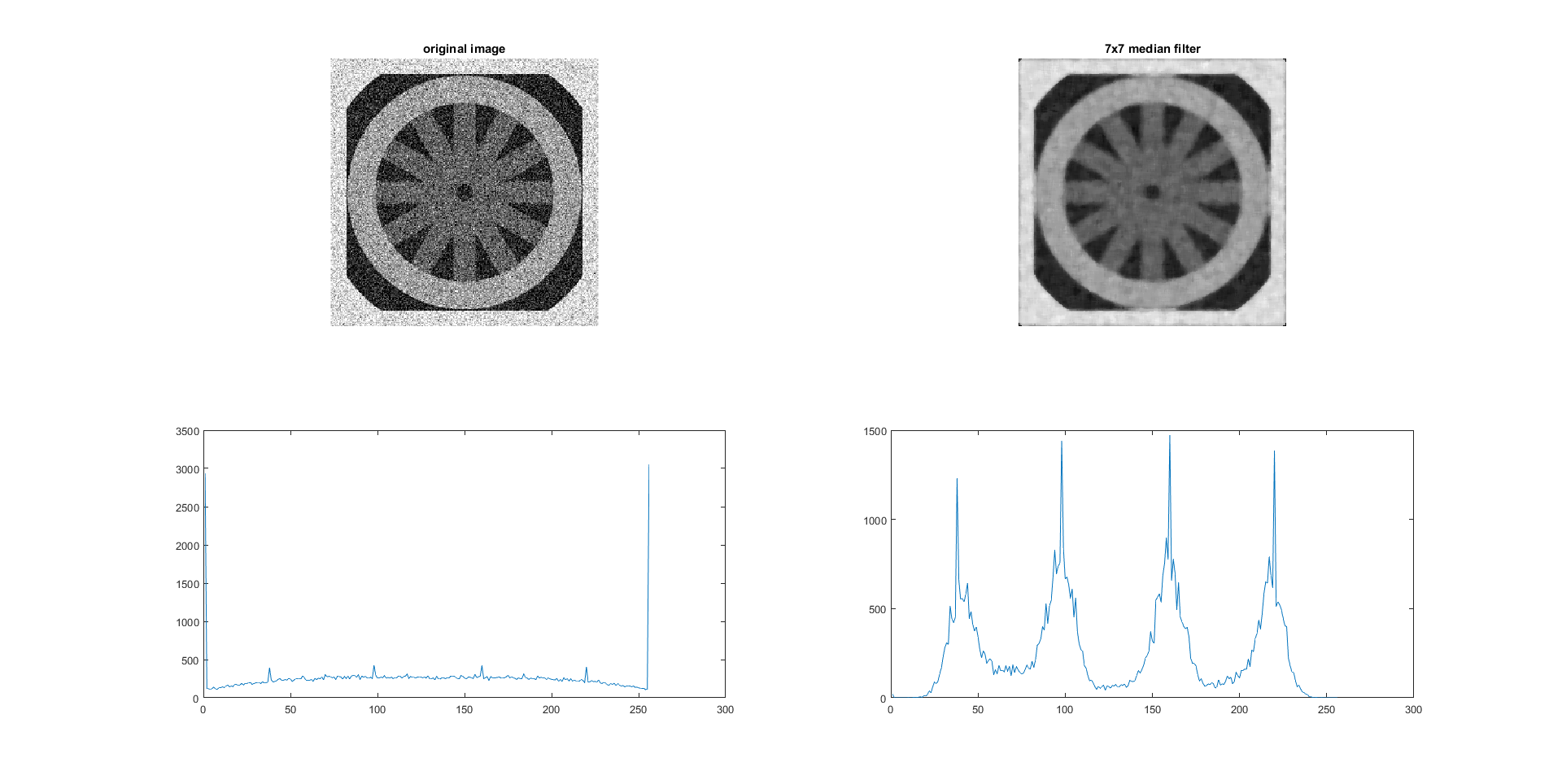


Figure 3.1.1 Original cwheelnoise image and result of it filtered by median filter after 1st iteration

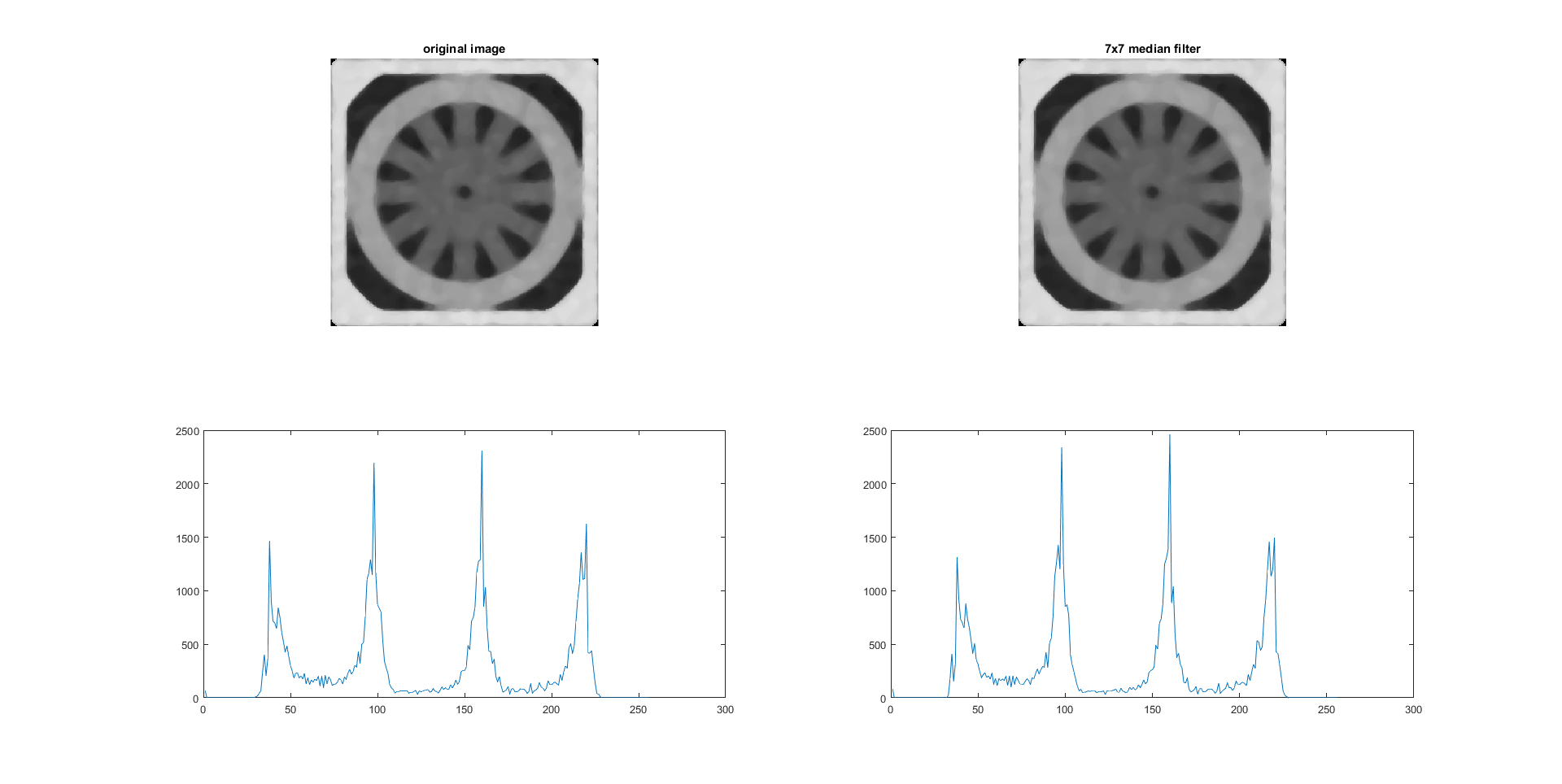


Figure 3.1.2 Original cwheelnoise image and result of it filtered by median filter after 5 iterations

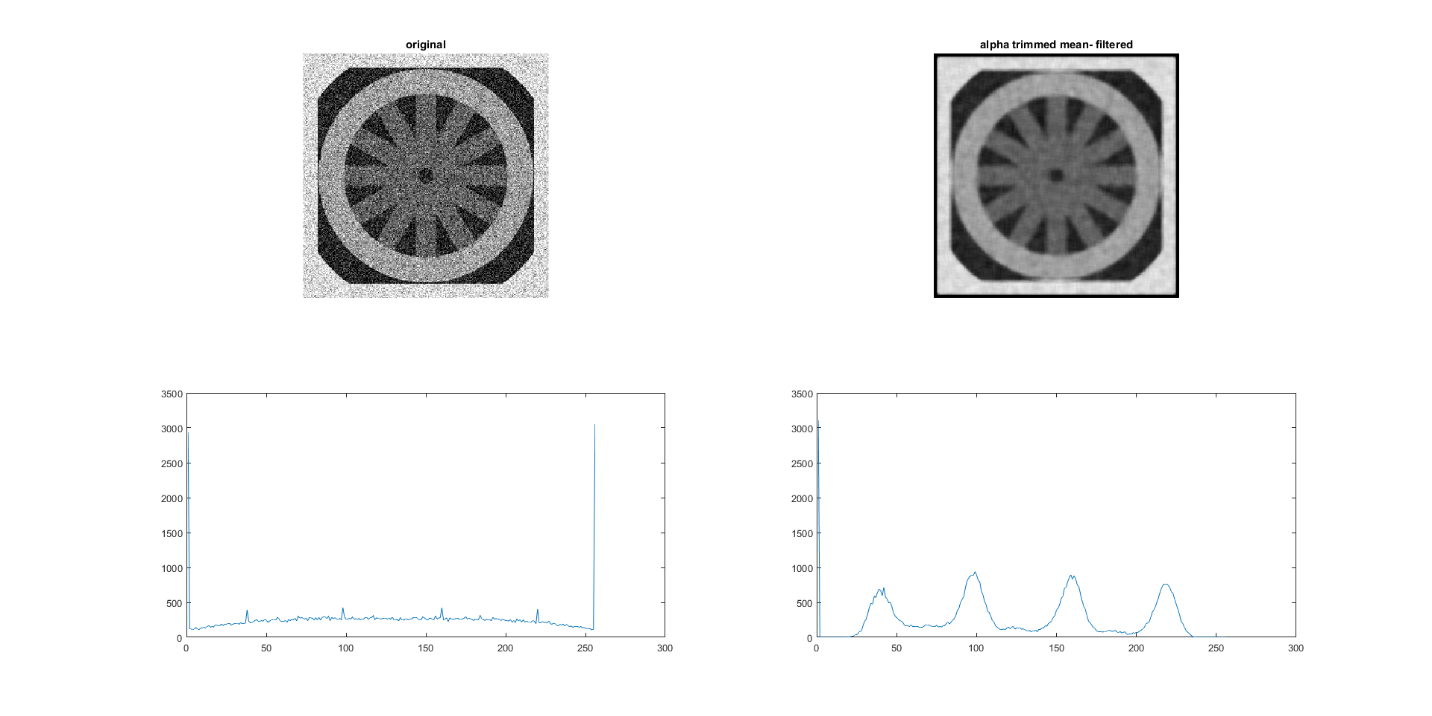


Figure 3.1.3 Original cwheelnoise image and result of it filtered by alpha-trim mean filter (α=0.25) after 1st iteration

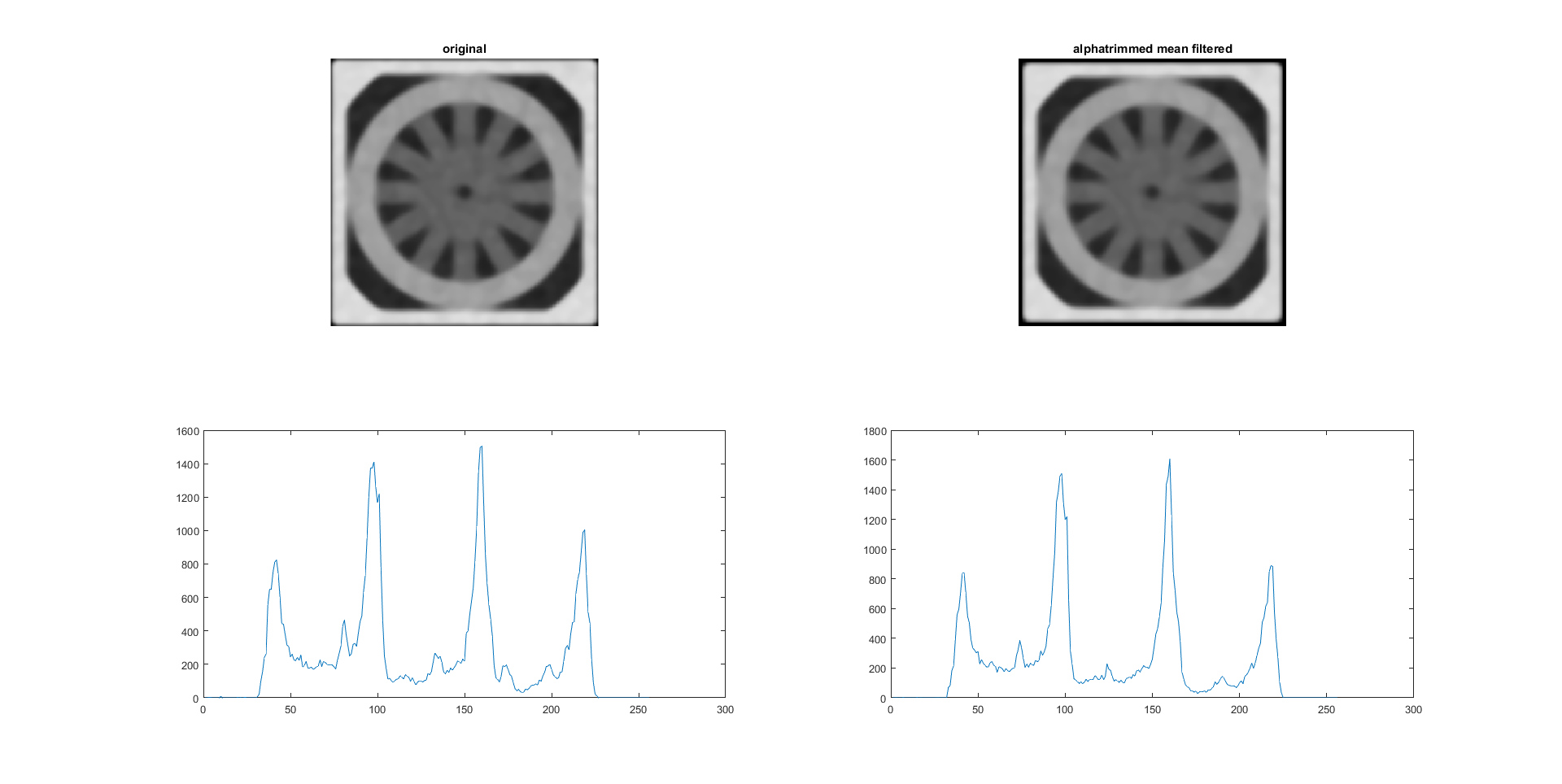


Figure 3.1.4 Original cwheelnoise image and result of it filtered by alpha-trim mean filter (α=0.25) after 5 iterations

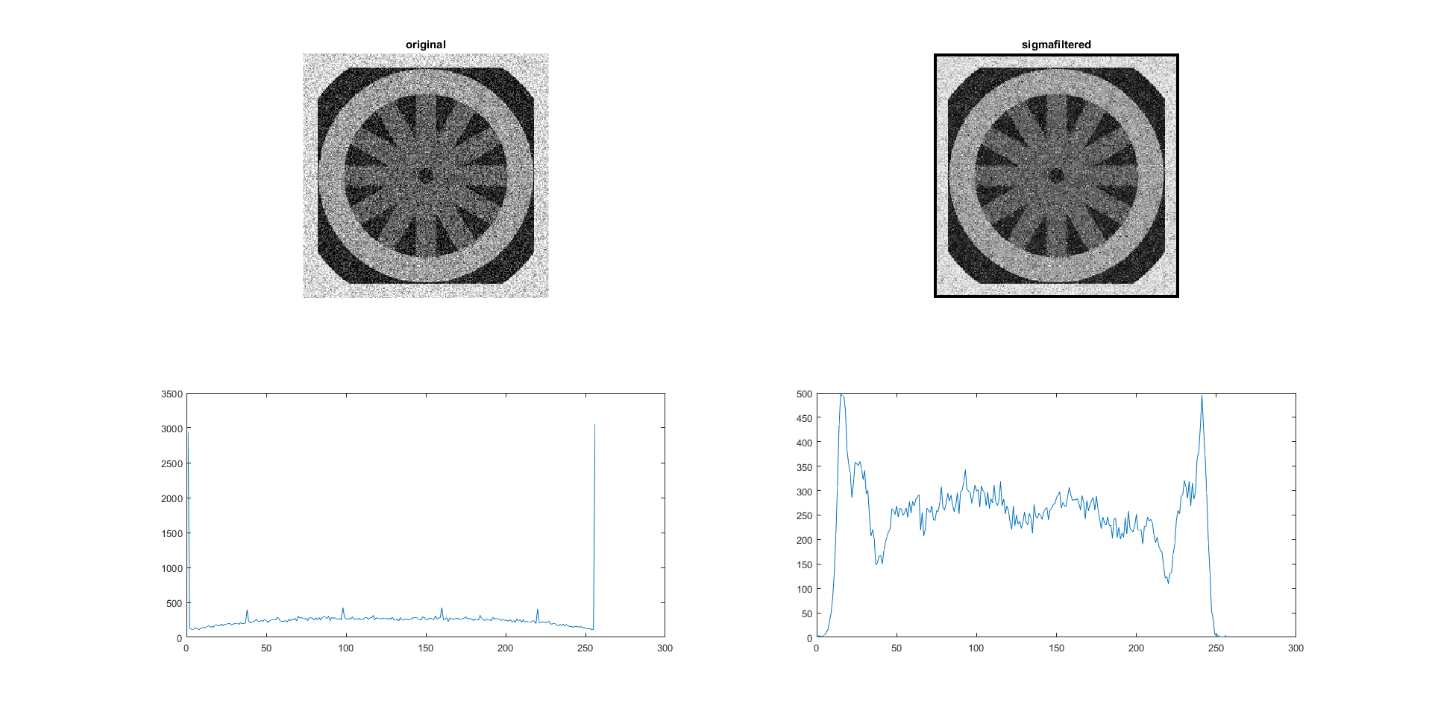


Figure 3.1.5 Original cwheelnoise image and result of it filtered by sigma filter (σ = 20)

after 1st iteration

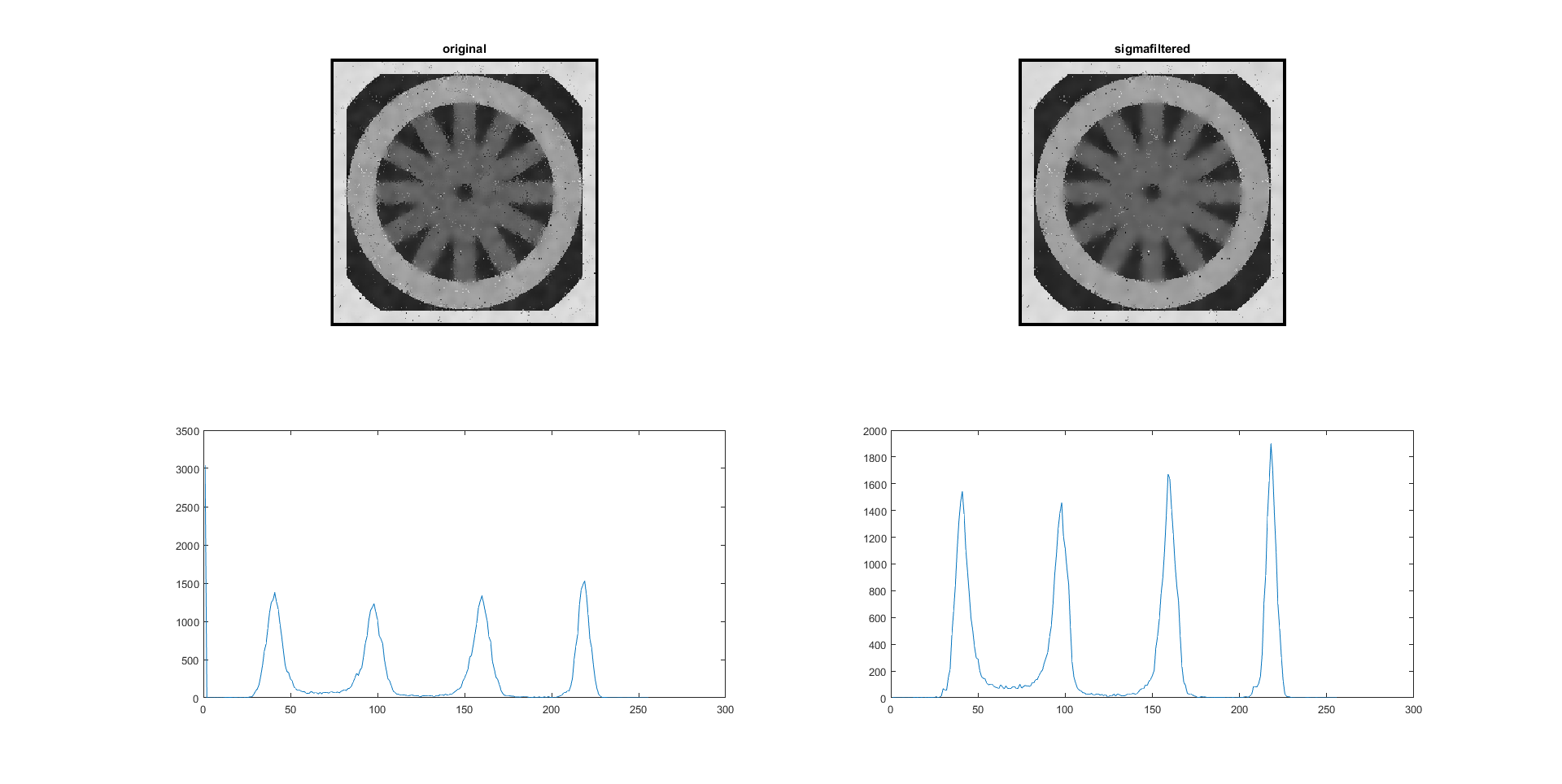


Figure 3.1.6 Original cwheelnoise image and result of it filtered by sigma filter (σ = 20)

after 5 iterations

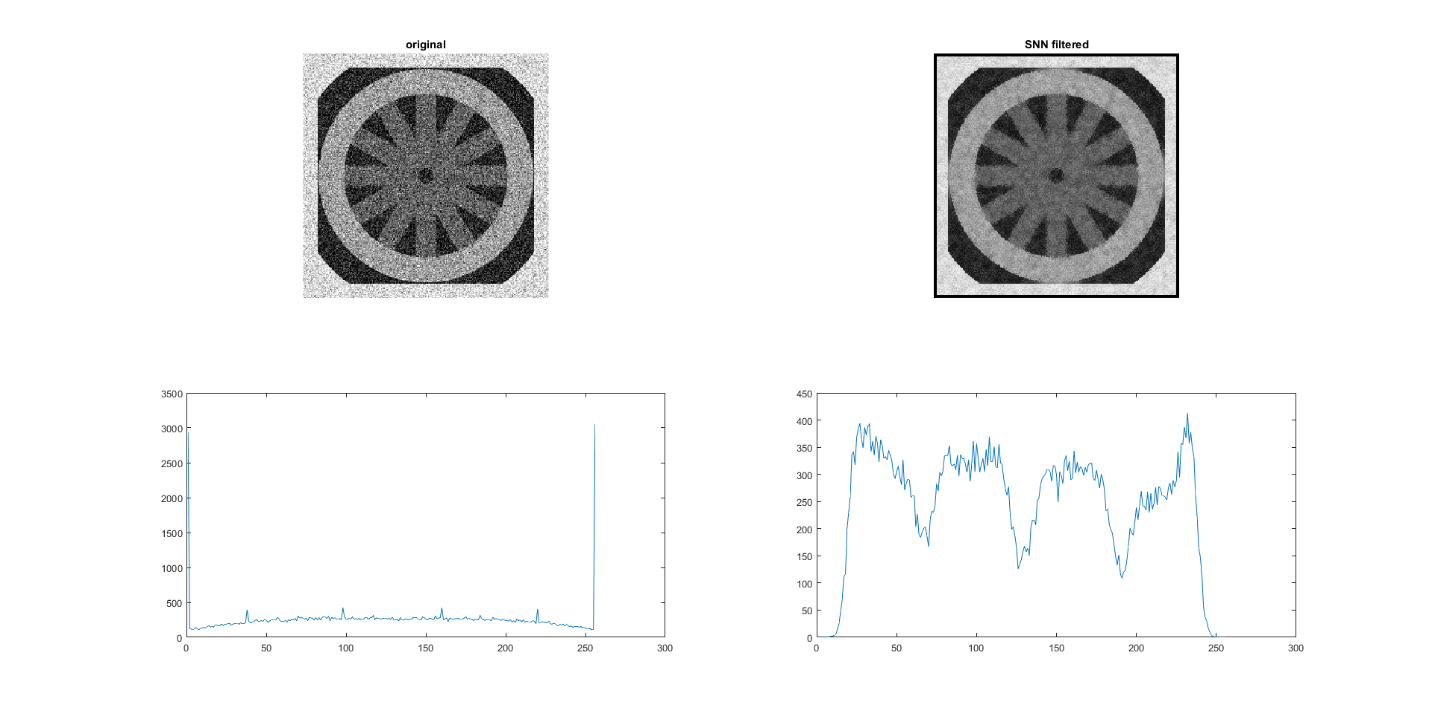


Figure 3.1.7 Original cwheelnoise image and result of it filtered by snn mean filter after 1st iteration

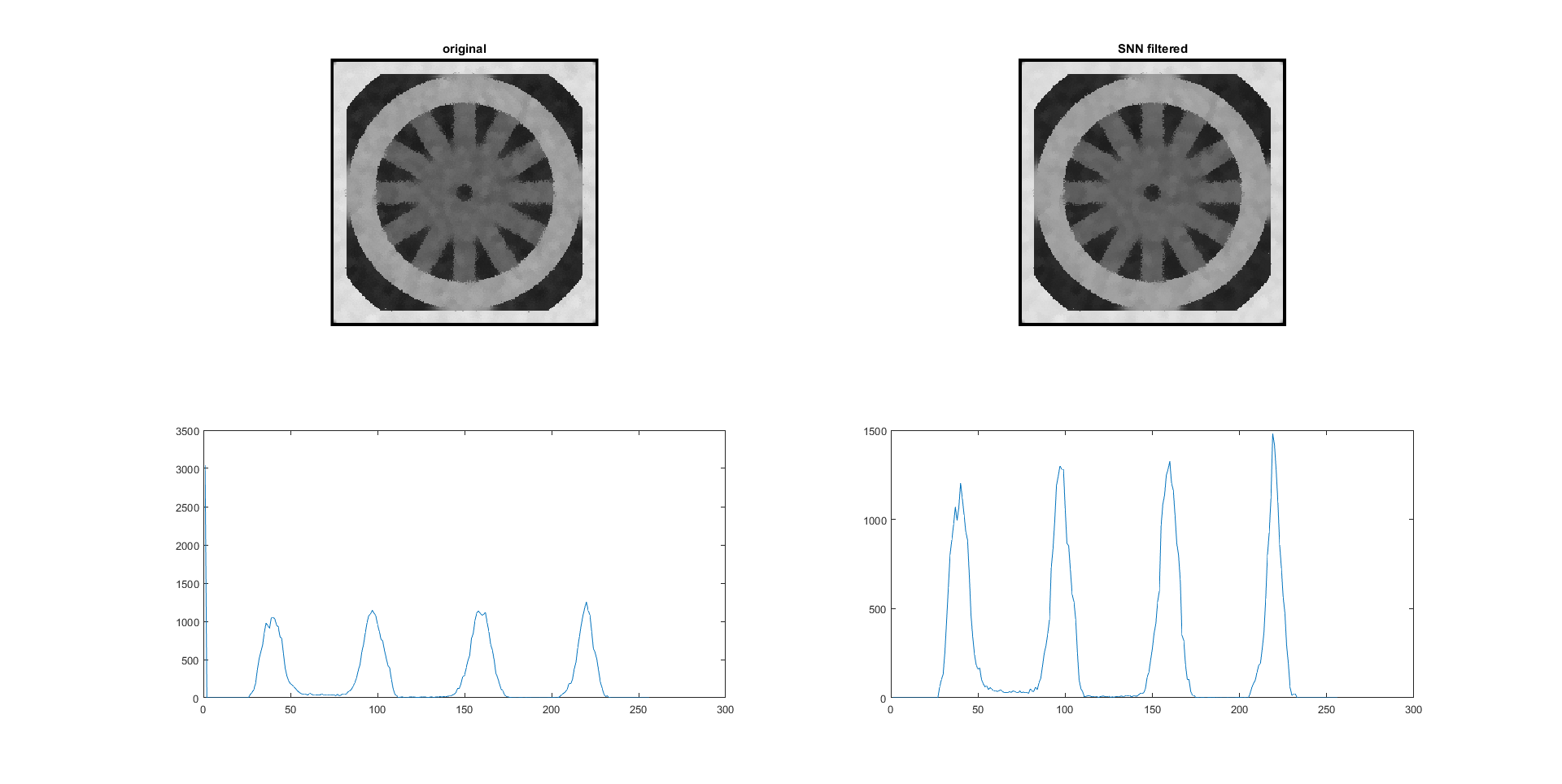


Figure 3.1.8 Original cwheelnoise image and result of it filtered by snn mean filter after 5 iterations

Figure 3.1.1 to Figure 3.1.8 shows cwheelnoise image fitered by the 4 filters mentioned above after 1 or 5 iterations

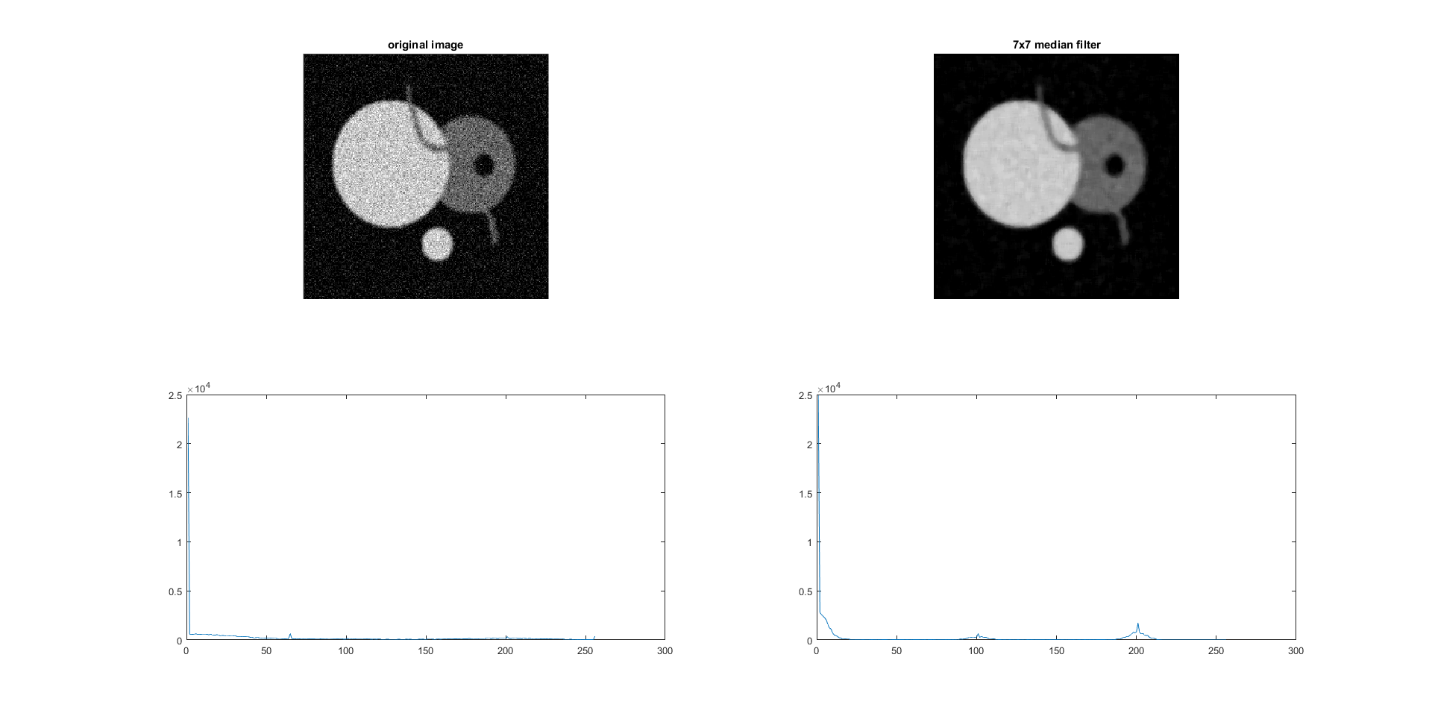


Figure 3.1.9 Original disk image and result of it filtered by median filter after 1st iteration

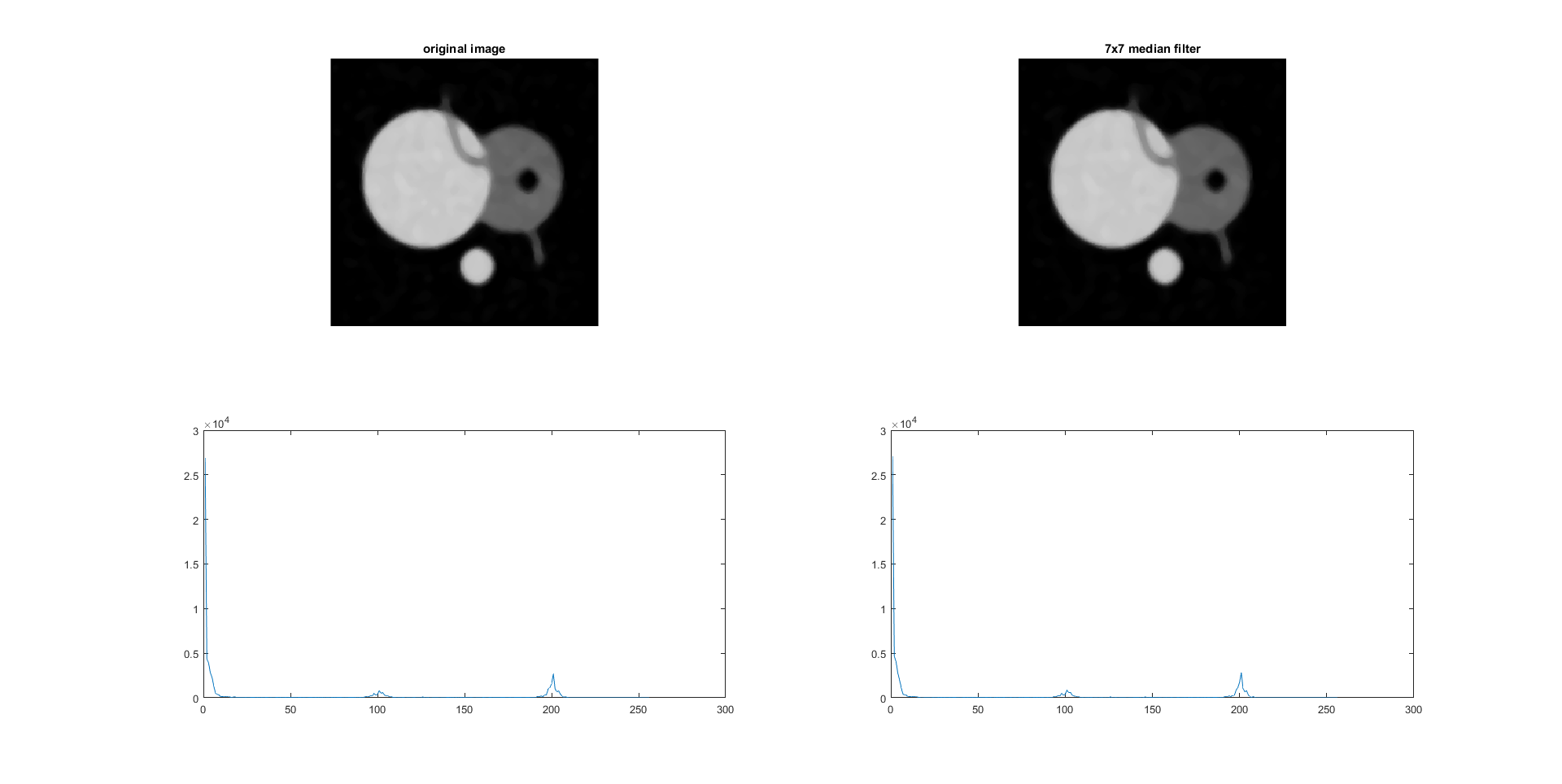


Figure 3.1.9 Original disk image and result of it filtered by median filter after 5 iterations

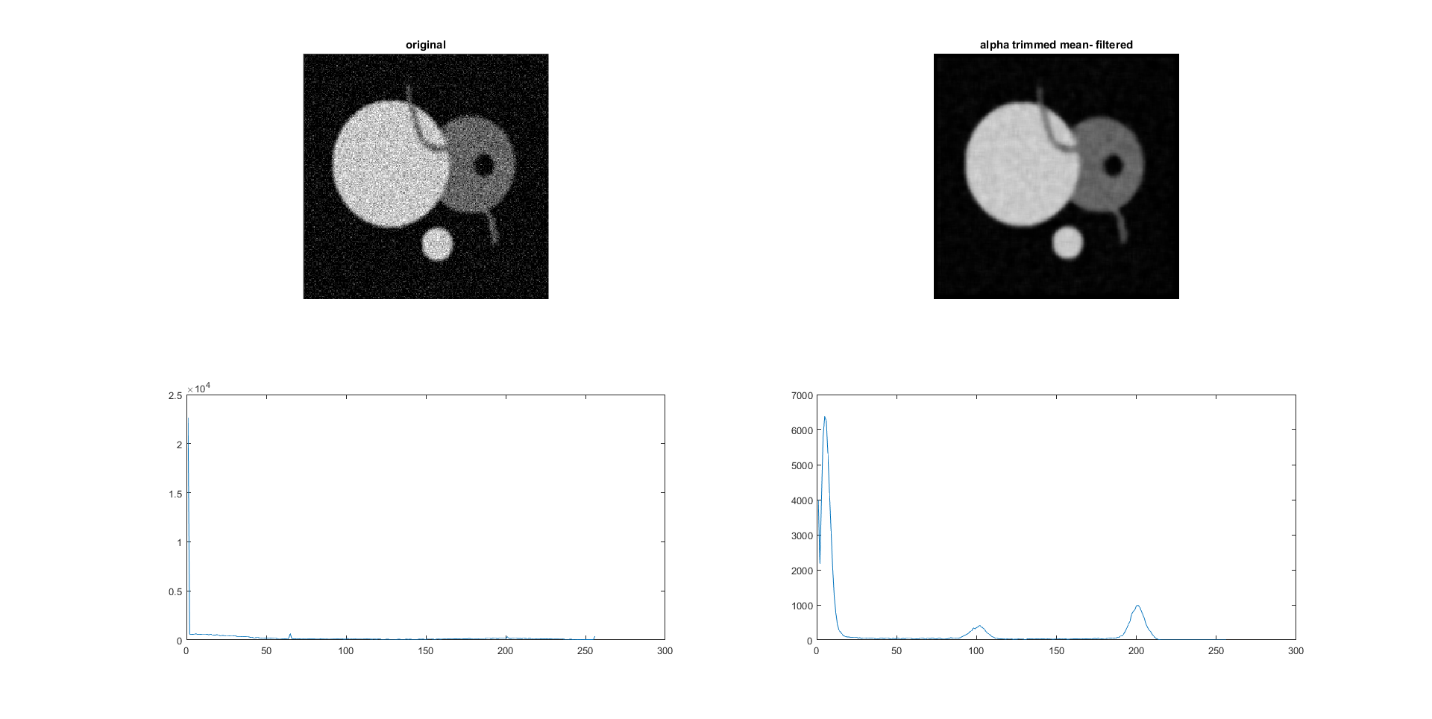


Figure 3.1.10 Original disk image and result of it filtered by alpha-trimmed mean filter (α=0.25) after 1st iteration

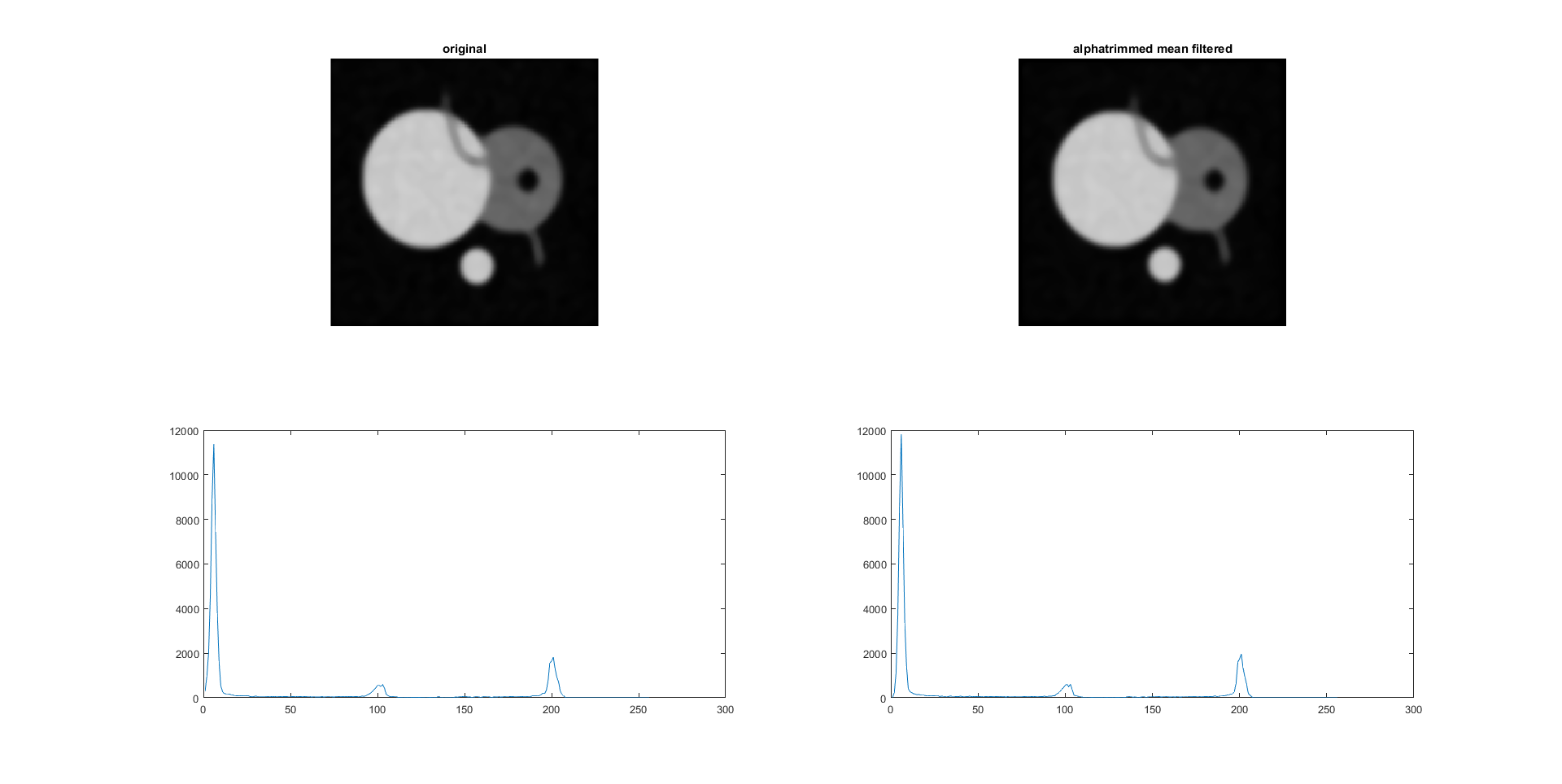


Figure 3.1.11 Original disk image and result of it filtered by alpha-trimmed mean filter (α=0.25) after 5 iterations

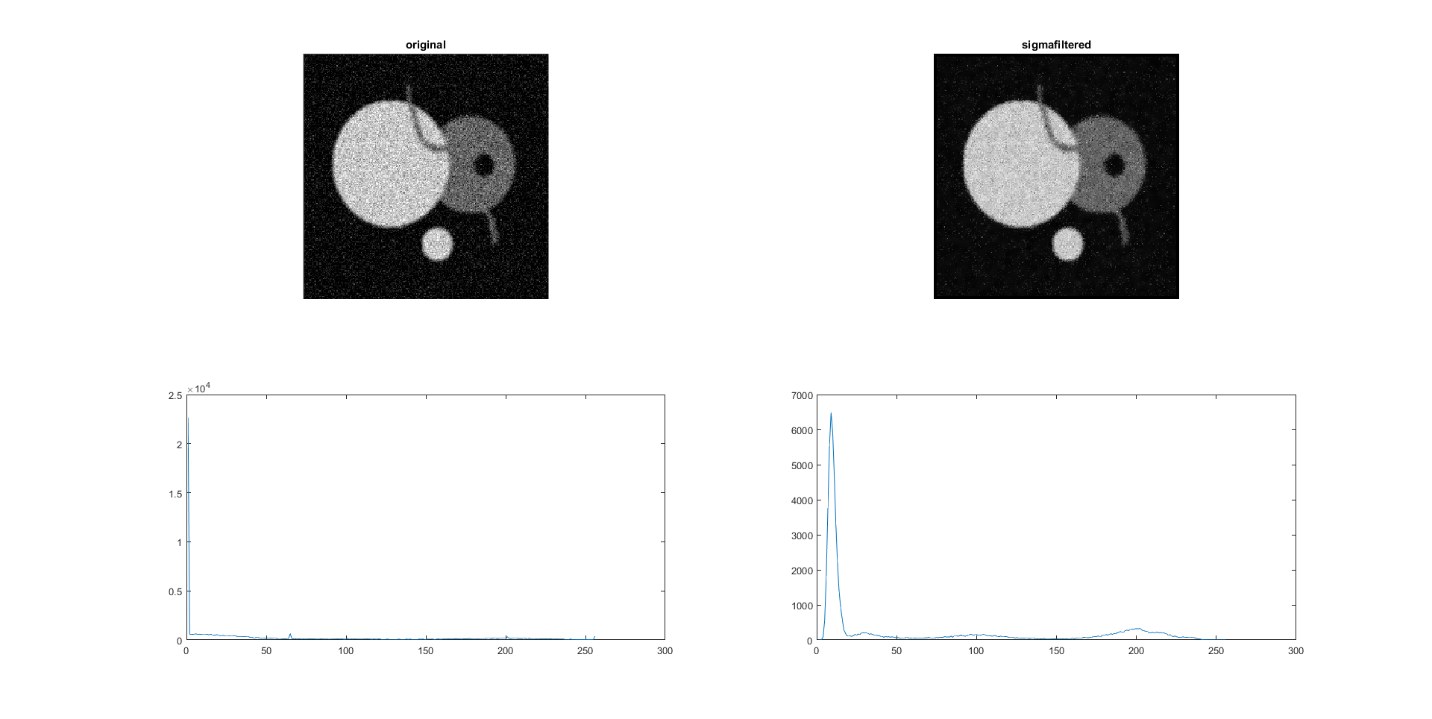


Figure 3.1.12 Original disk image and result of it filtered by sigma filter (σ = 20)

after 1st iteration

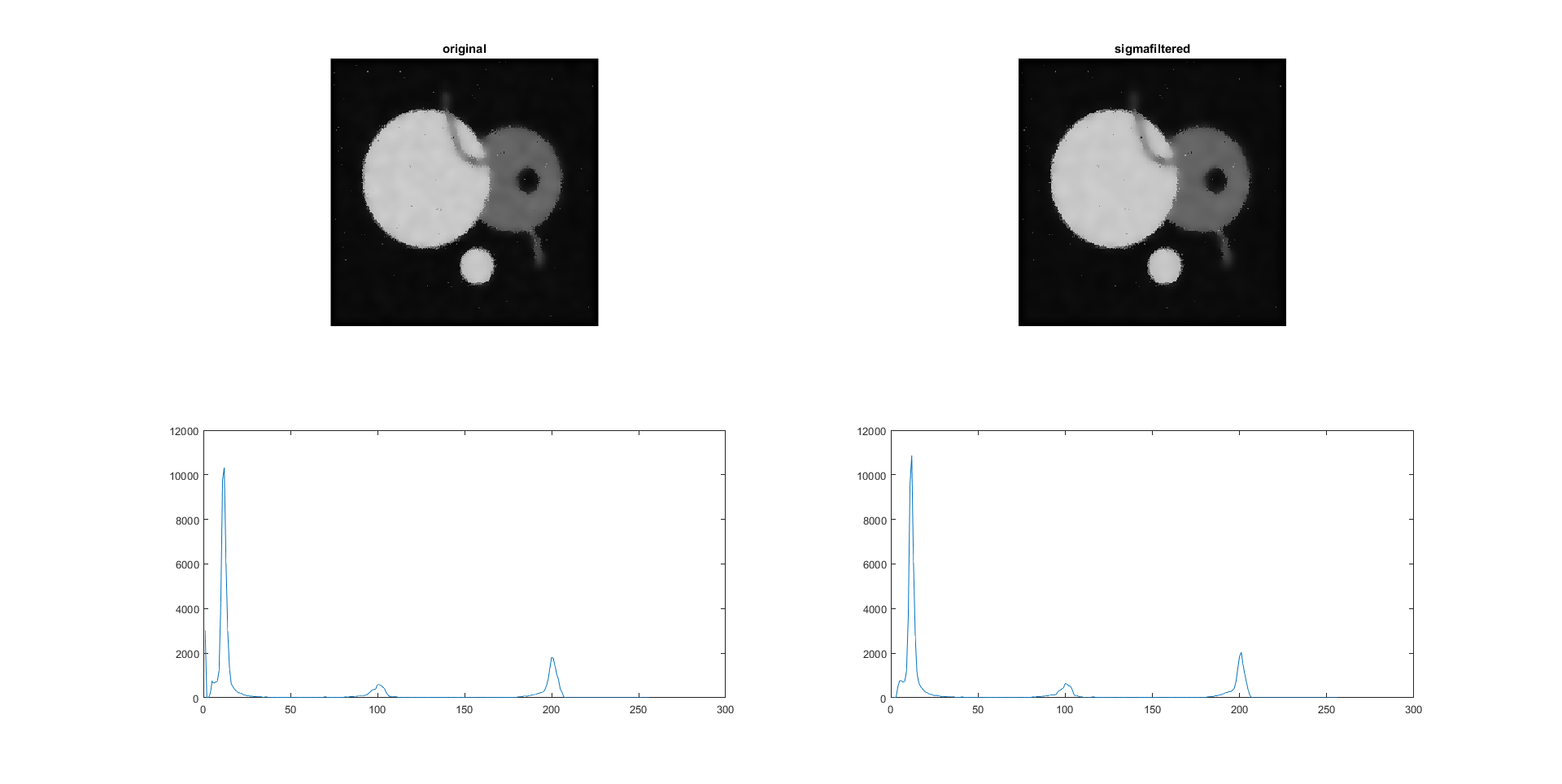


Figure 3.1.13 Original disk image and result of it filtered by sigma filter (σ = 20)

after 5 iterations

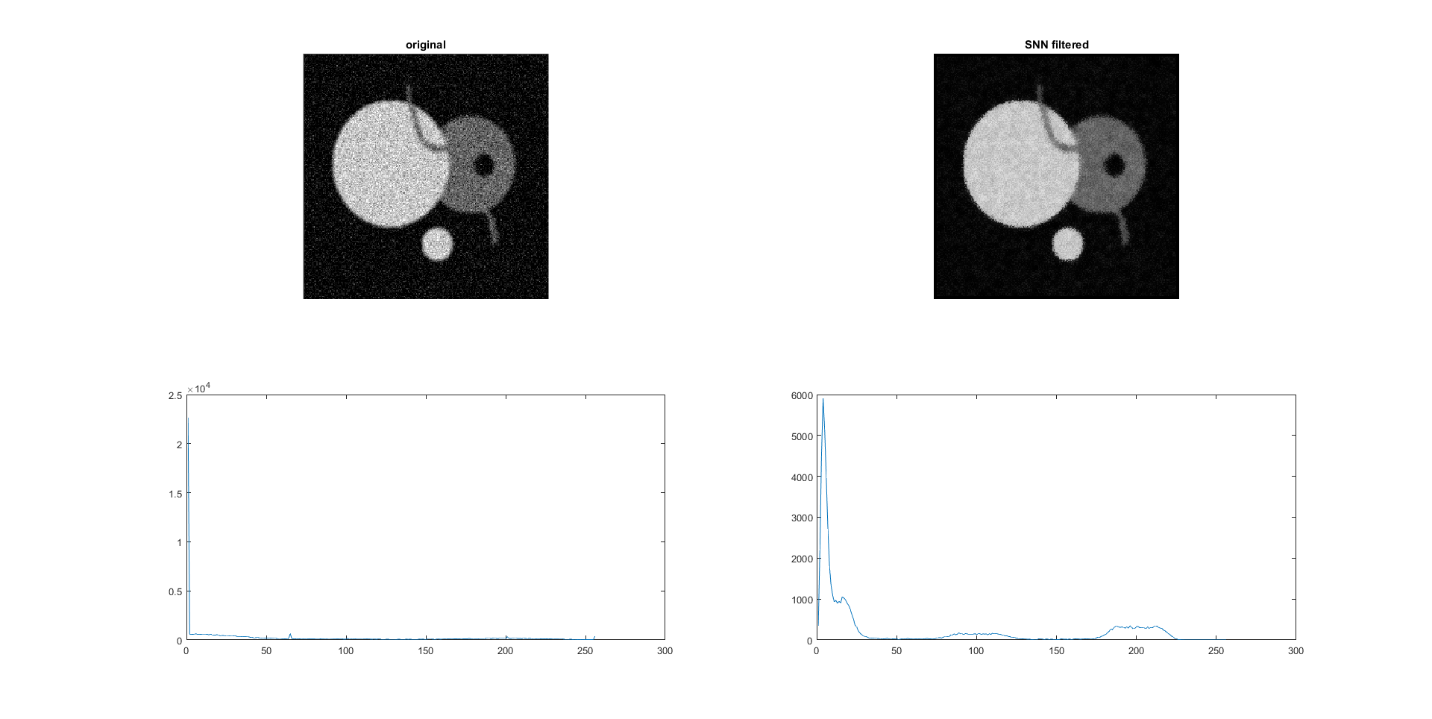


Figure 3.1.14 Original disk image and result of it filtered by snn mean filter after 1st iteration

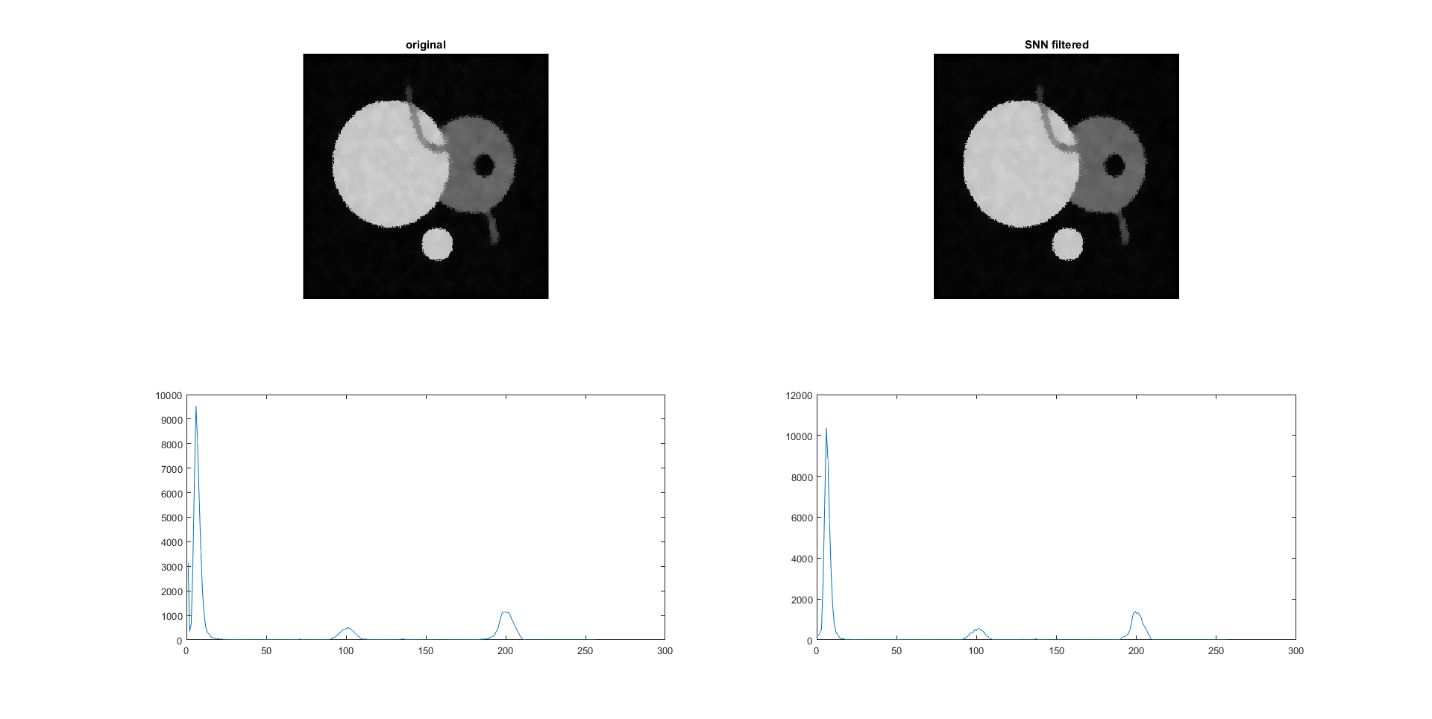


Figure 3.1.15 Original disk image and result of it filtered by snn mean filter after 5 iterations

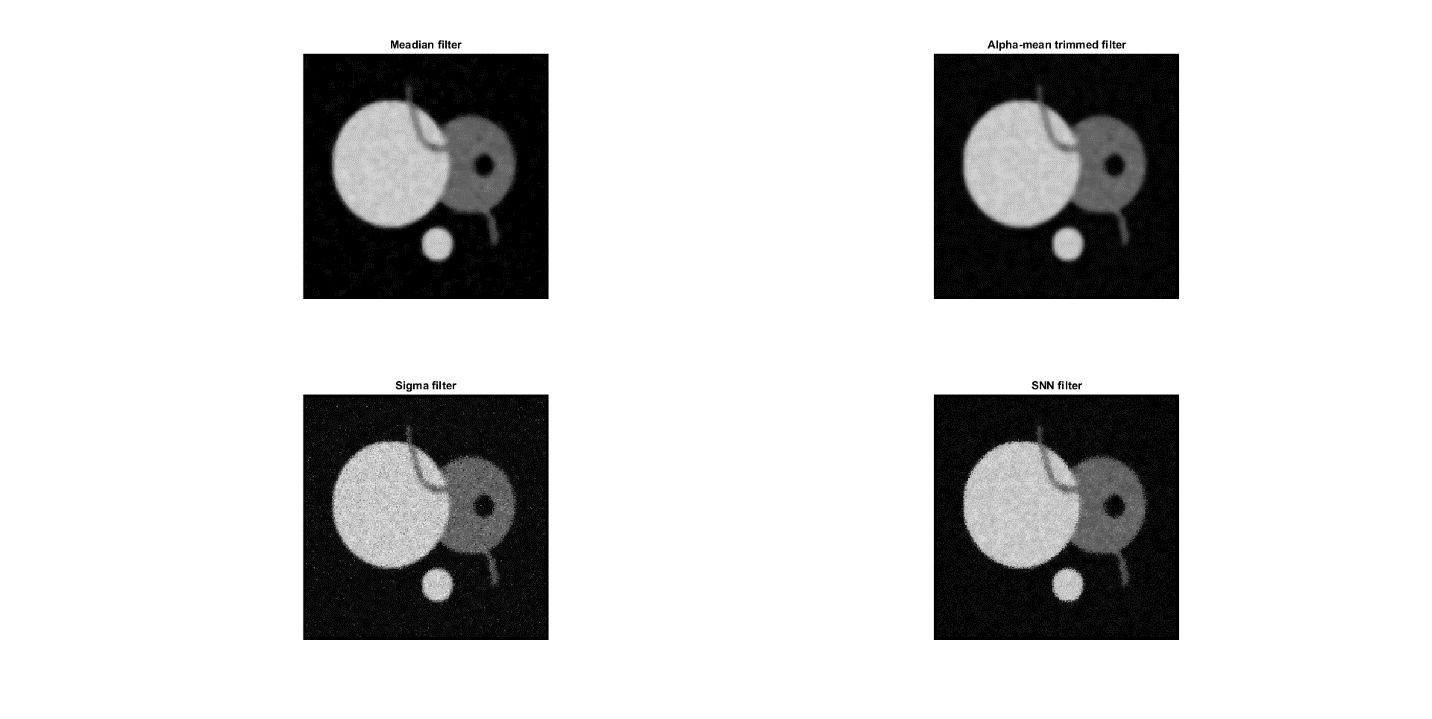


Figure 3.1.16 results of disk filtered by 4 different filters after 1 iterations

The ‘disk’ result consisting the mean and standard deviation of the interior of the large disk region is provided in the below table.

|  |  |  |  |
| --- | --- | --- | --- |
| Filter | Iterations | Mean | Standard Deviation |
| Median | 1 | 158.6697 | 71.6799 |
| Median | 5 | 158.1134 | 71.1877 |
| Alpha trimmed Mean | 1 | 158.9811 | 70.4617 |
| Alpha trimmed Mean | 5 | 157.8540 | 68.6299 |
| Sigma | 1 | 160.2391 | 72.1377 |
| Sigma | 5 | 160.2756 | 69.904 |
| SNN | 1 | 159.8117 | 72.4557 |
| SNN | 5 | 159.8652 | 74.3959 |

Table 1 results of mean and standard deviation of the interior of the large disk

From the above results, we can observe that by using the median filter the edges of the image are not disturbed. The mean and standard deviation of the interior disk of the image shows that median filter did not disturb the image as it used existing values of the image.

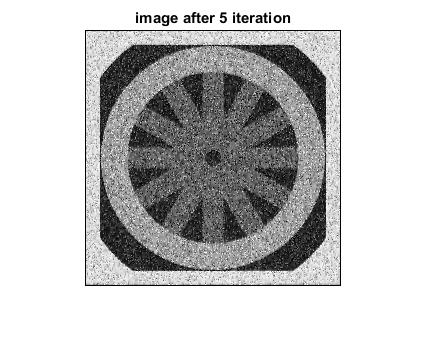
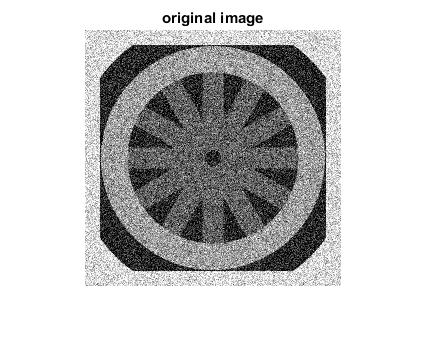
By using alpha trimmed mean filter the image gets blurred with each increasing iteration. The standard deviation and mean decreases with increase in iteration.

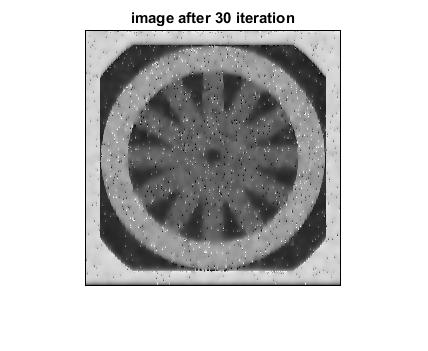
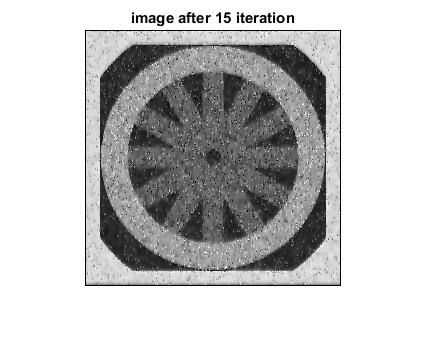
Sigma filter preserves the structure. With increase in iteration the standard deviation decreases with each increase in iteration. Pixel population sharpens and noise is better cleaned.

With the help of SNN filter the structure gets sharper. Noise gets removed but standard deviation of the internal disk shows that it increases with increase in iteration.

Problem 2

For this problem, we look into the effect of k and number of iterations when applying anisotropic diffusion to image cwheelnoise





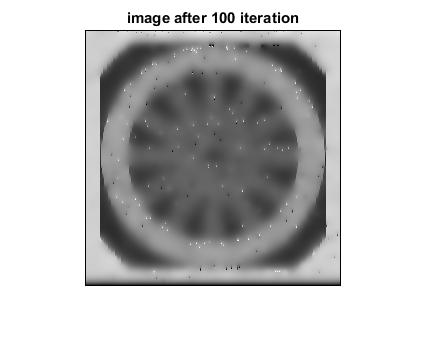
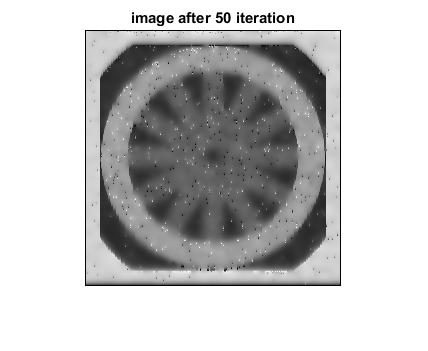


Figure 3.2.1 results after applying anisotropic diffusion filter to cwheelnoise K=25, g(.)=exponential, Lambda=0.25

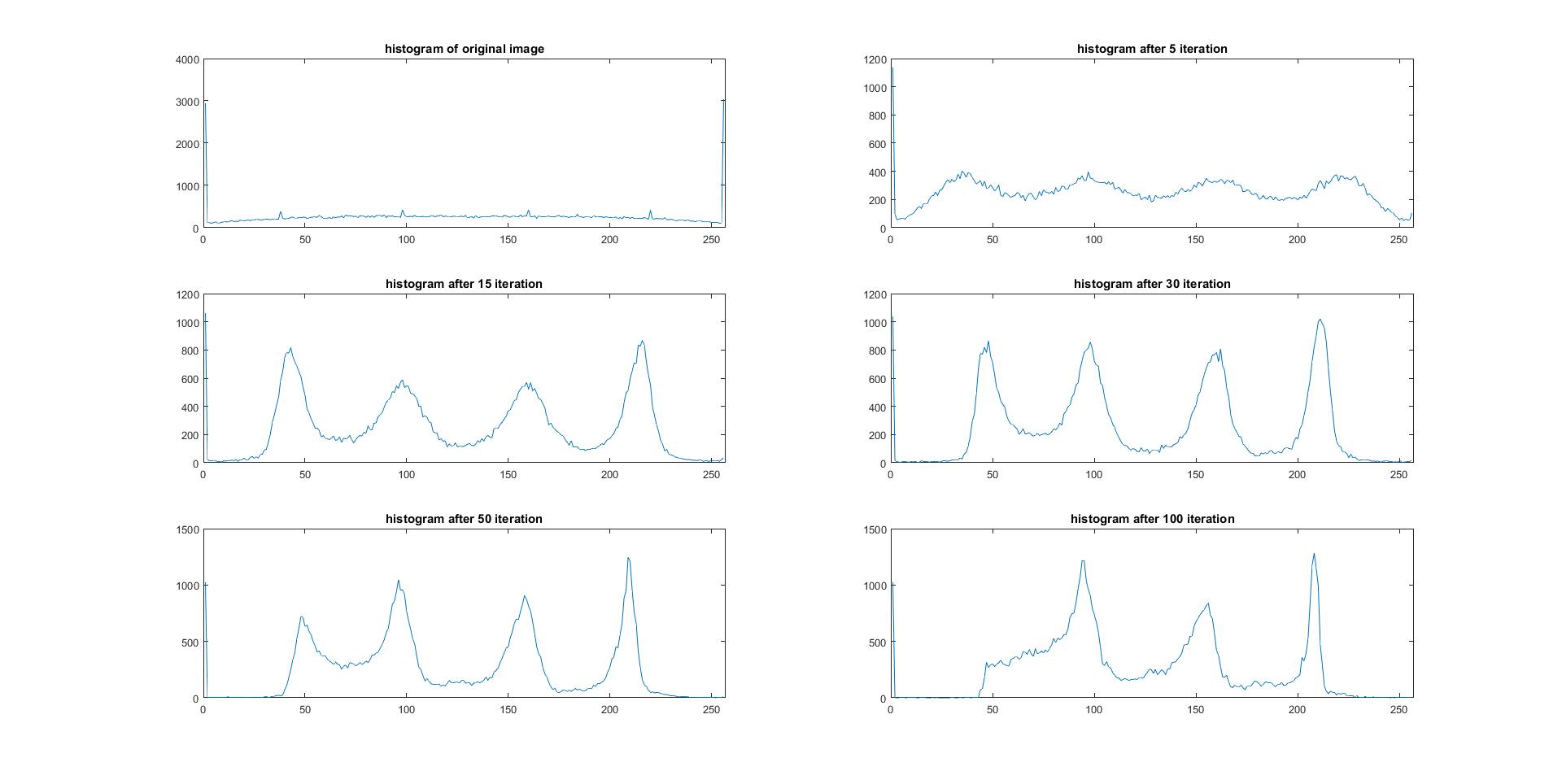


Figure 3.2.2 Corresponding histograms for images in Figure 3.2.1

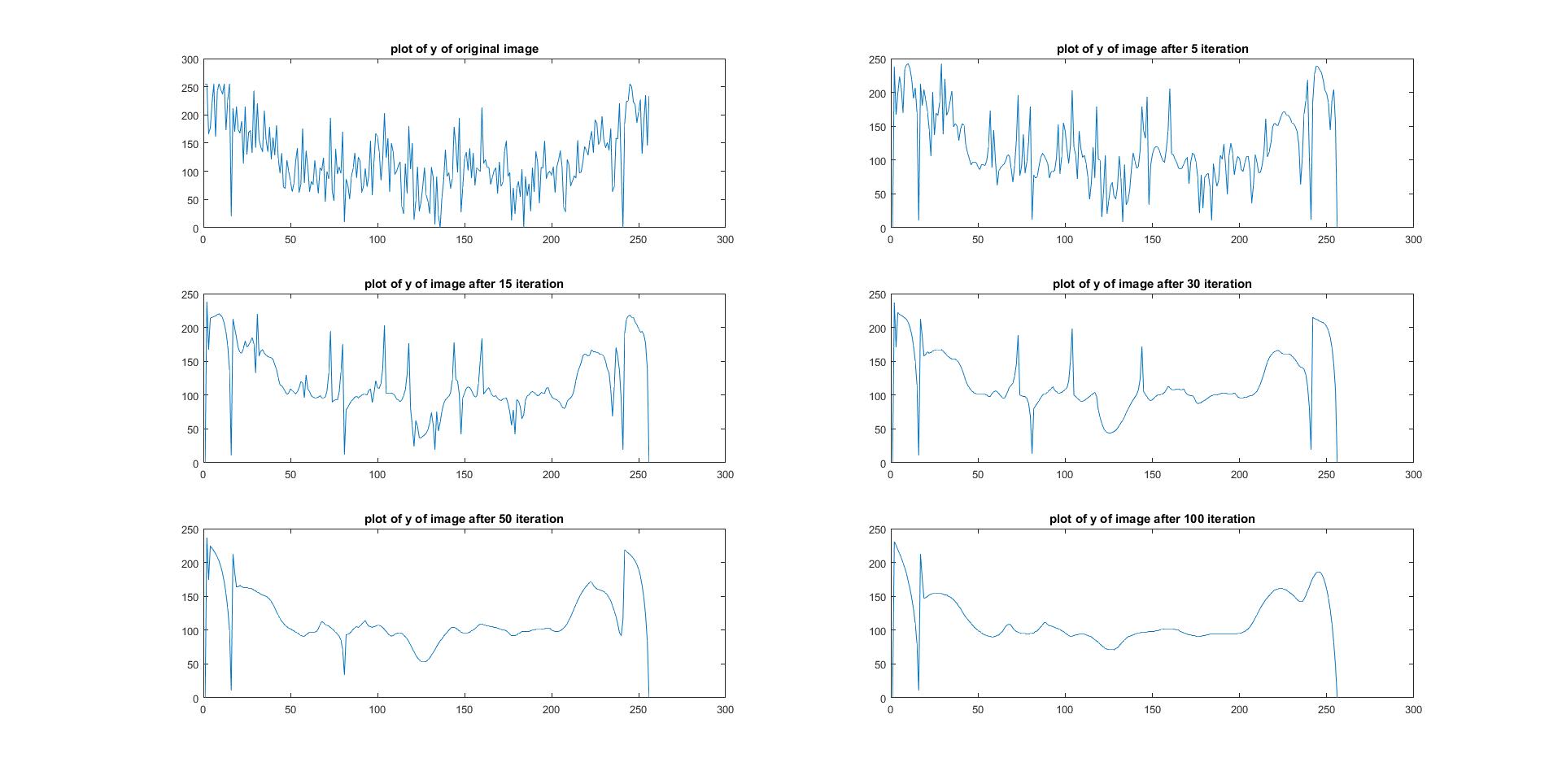
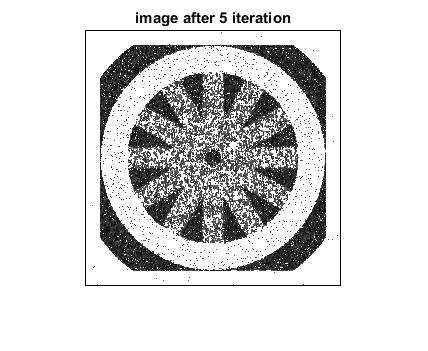
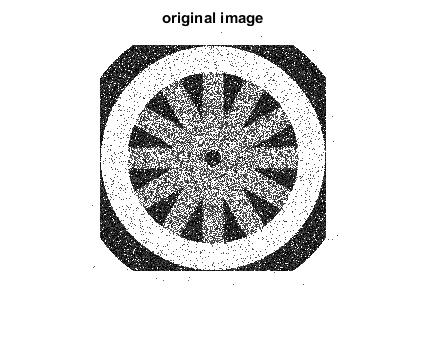
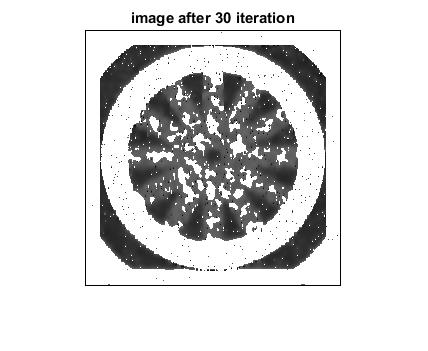
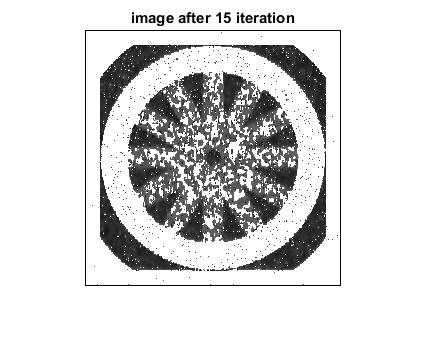


Figure 3.2.3 A plot of the line y = 128 through the image cwheelnoise





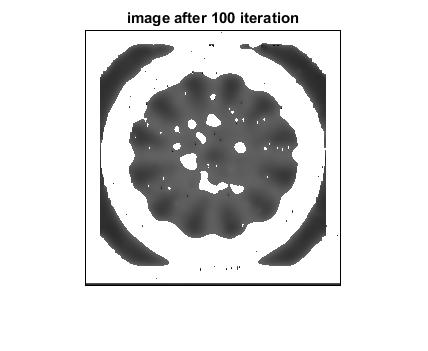
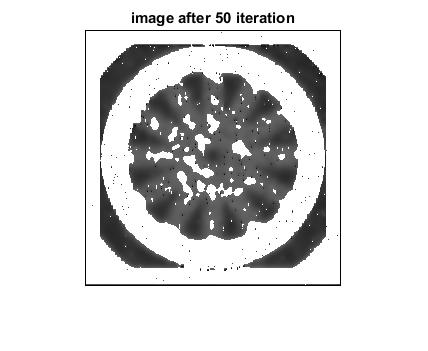
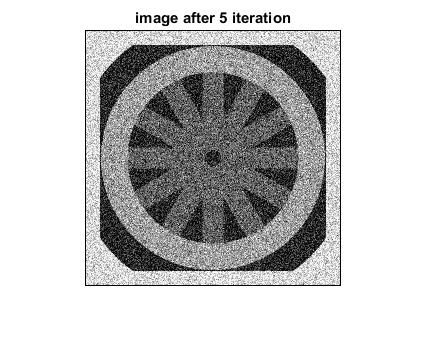
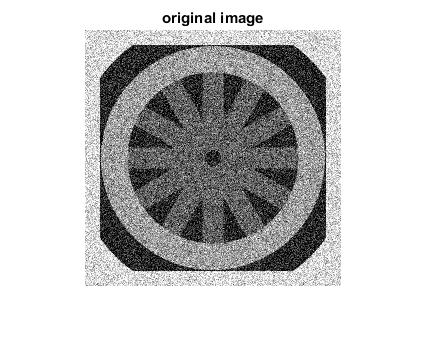
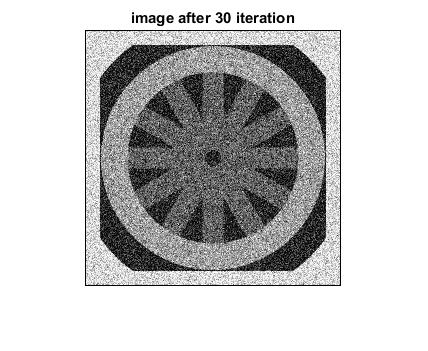


Figure 3.2.4 results after applying anisotropic diffusion filter to the segmented versions of image cwheelnoise K=25, g(.)=exponential, Lambda=0.25





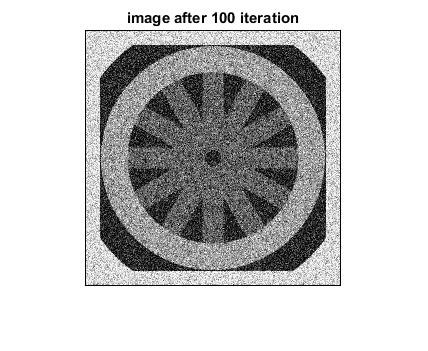
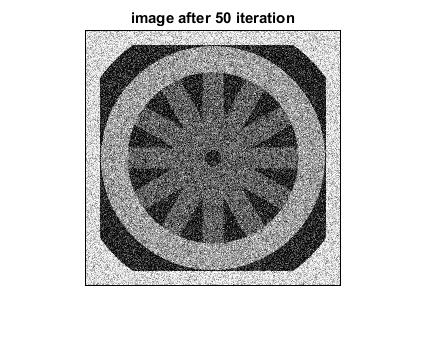


Figure 3.2.5 results after applying anisotropic diffusion filter to cwheelnoise K=5, g(.)=exponential, Lambda=0.25

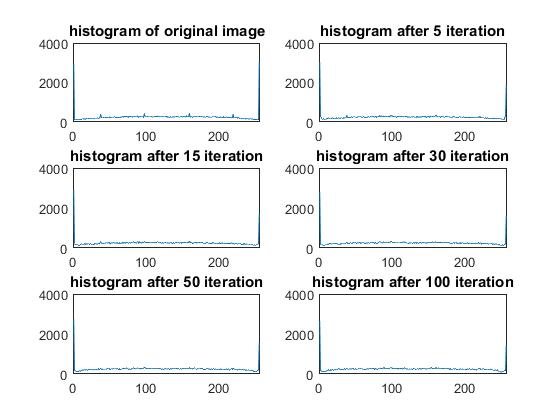


Figure 3.26 Corresponding histograms for images in Figure 3.2.5

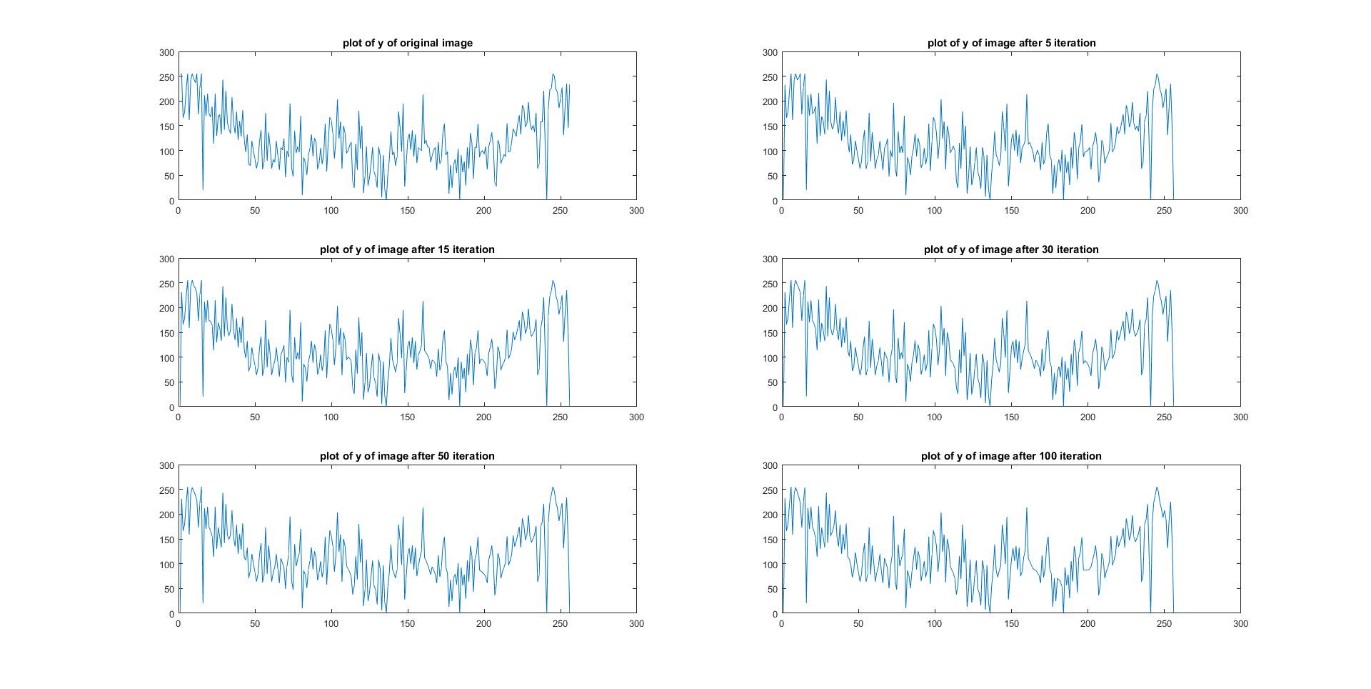
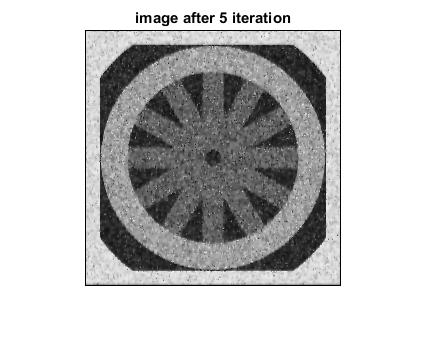
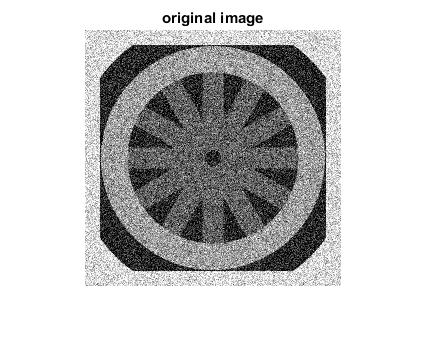
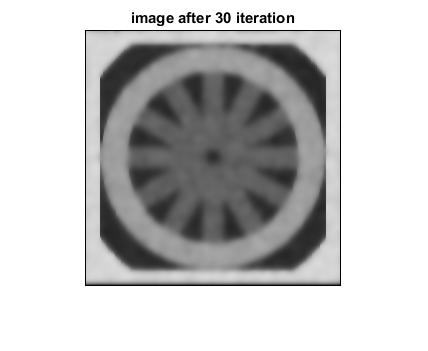
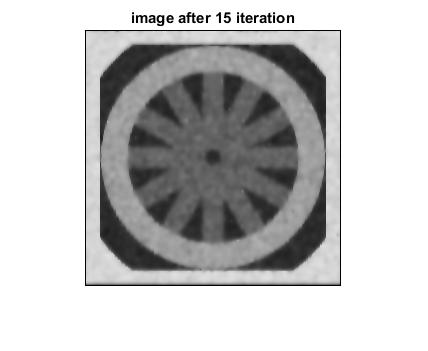


Figure 3.2.7 A plot of the line *y* = 128 through the image cwheelnoise

With K the diffusion changes. For exponential function a good diffusion is produced when K>10 else the image will not be diffused.





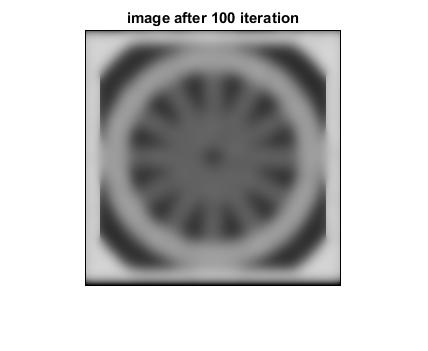
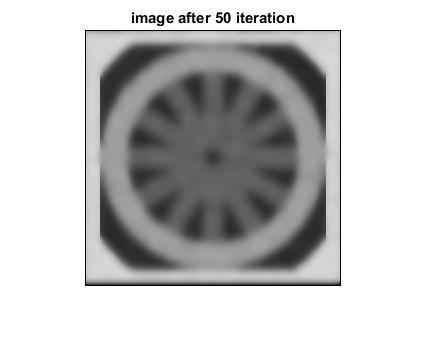


Figure 3.2.8 results after applying anisotropic diffusion filter to cwheelnoise K=25, g(.)=quadratic, Lambda=0.25

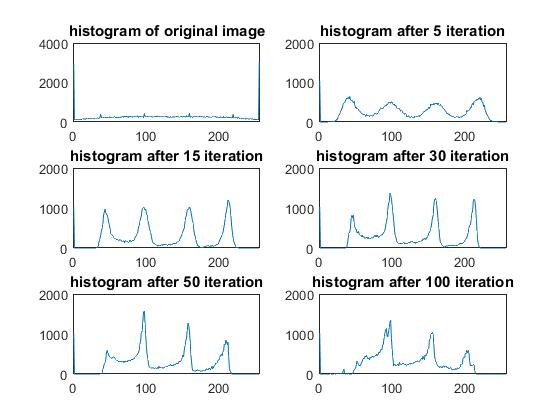


Figure 3.2.9 Corresponding histograms for images in Figure 3.2.8

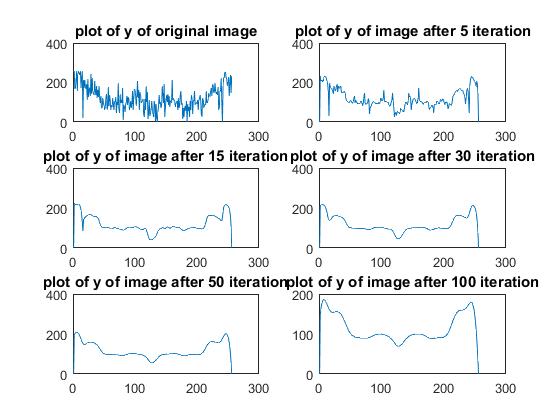
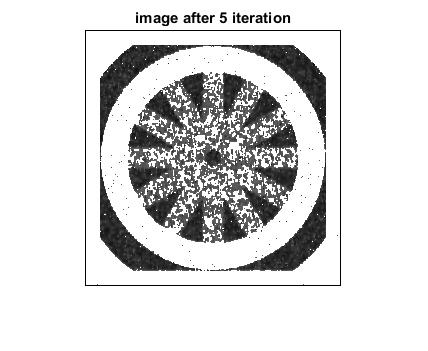
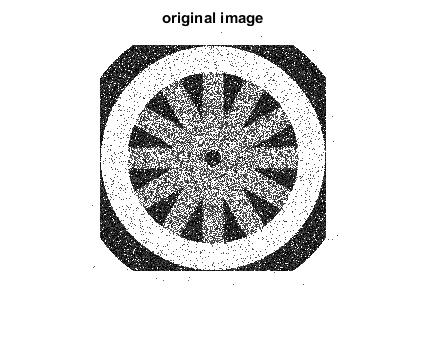
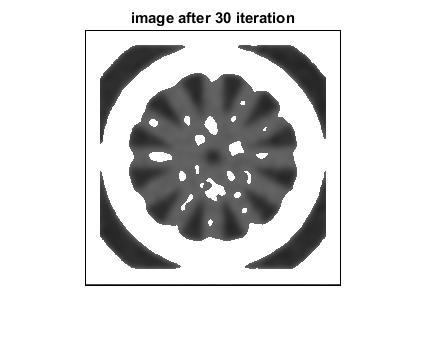
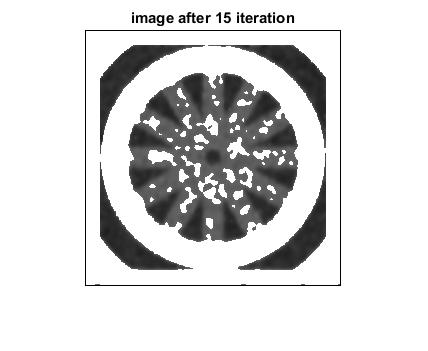


Figure 3.2.10 A plot of the line *y* = 128 through the image cwheelnoise





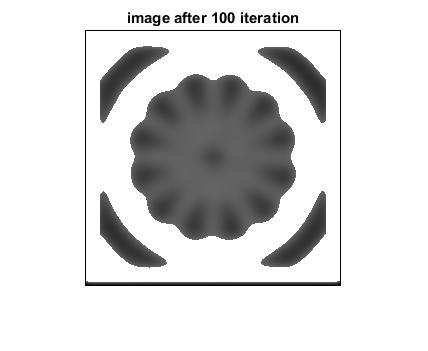
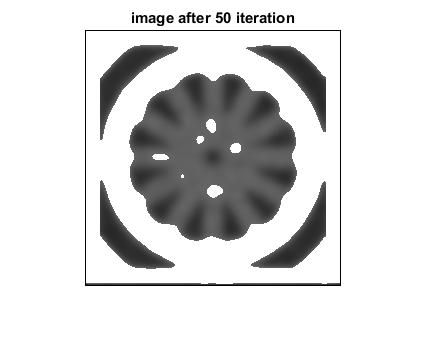
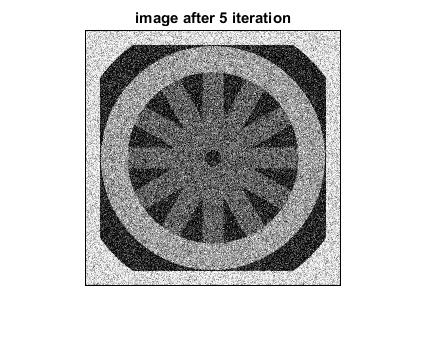
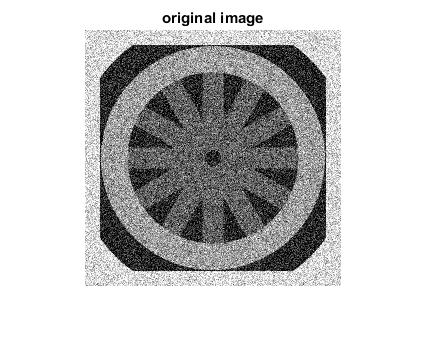
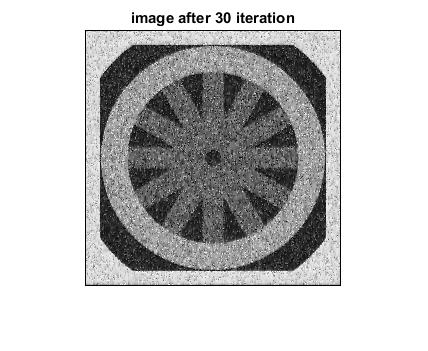
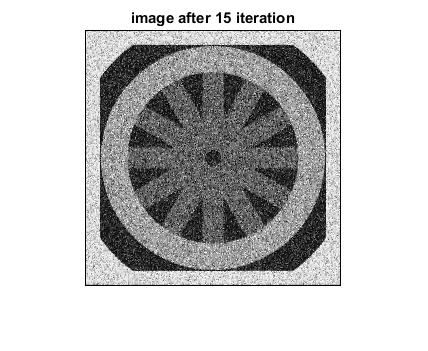


Figure 3.2.11 results after applying anisotropic diffusion filter to segmented versions of cwheelnoise K=25, g(.)=quadratic, Lambda=0.25





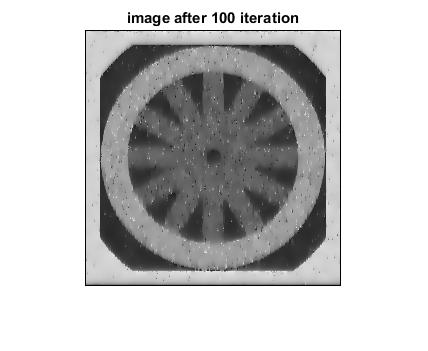
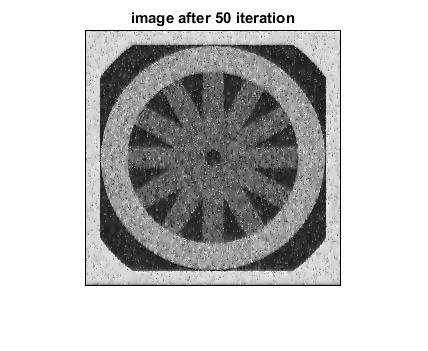


Figure 3.2.12 results after applying anisotropic diffusion filter to cwheelnoise K=5, g(.)=quadratic, Lambda=0.25

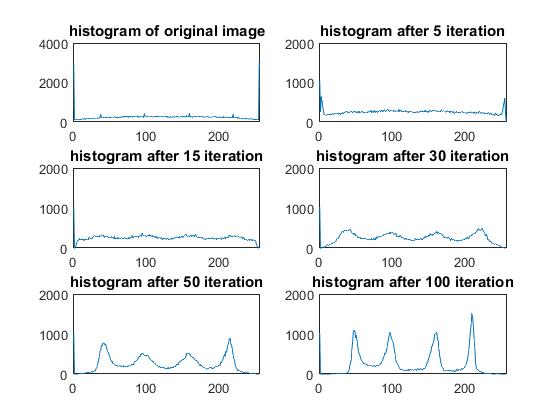


Figure 3.2.13 Corresponding histograms for images in Figure 3.2.12



Figure 3.2.14 A plot of the line *y* = 128 through the image cwheelnoise

Similar to the result of exponential function with K the diffusion changes. With decreasing K, the diffusion is less. However, in case of quadratic function there is still some diffusion compared to exponential for same value of K=5.

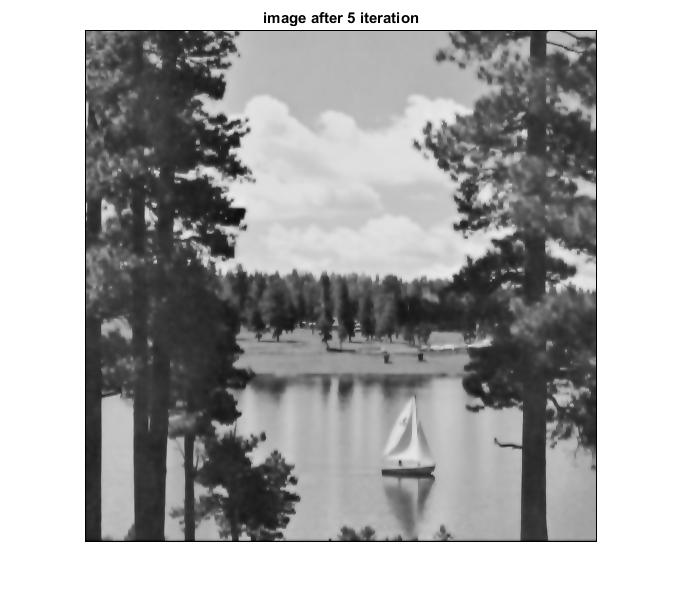






Figure 3.2.15 results after applying anisotropic diffusion filter to lake K=25, g(.)=exponential, Lambda=0.25

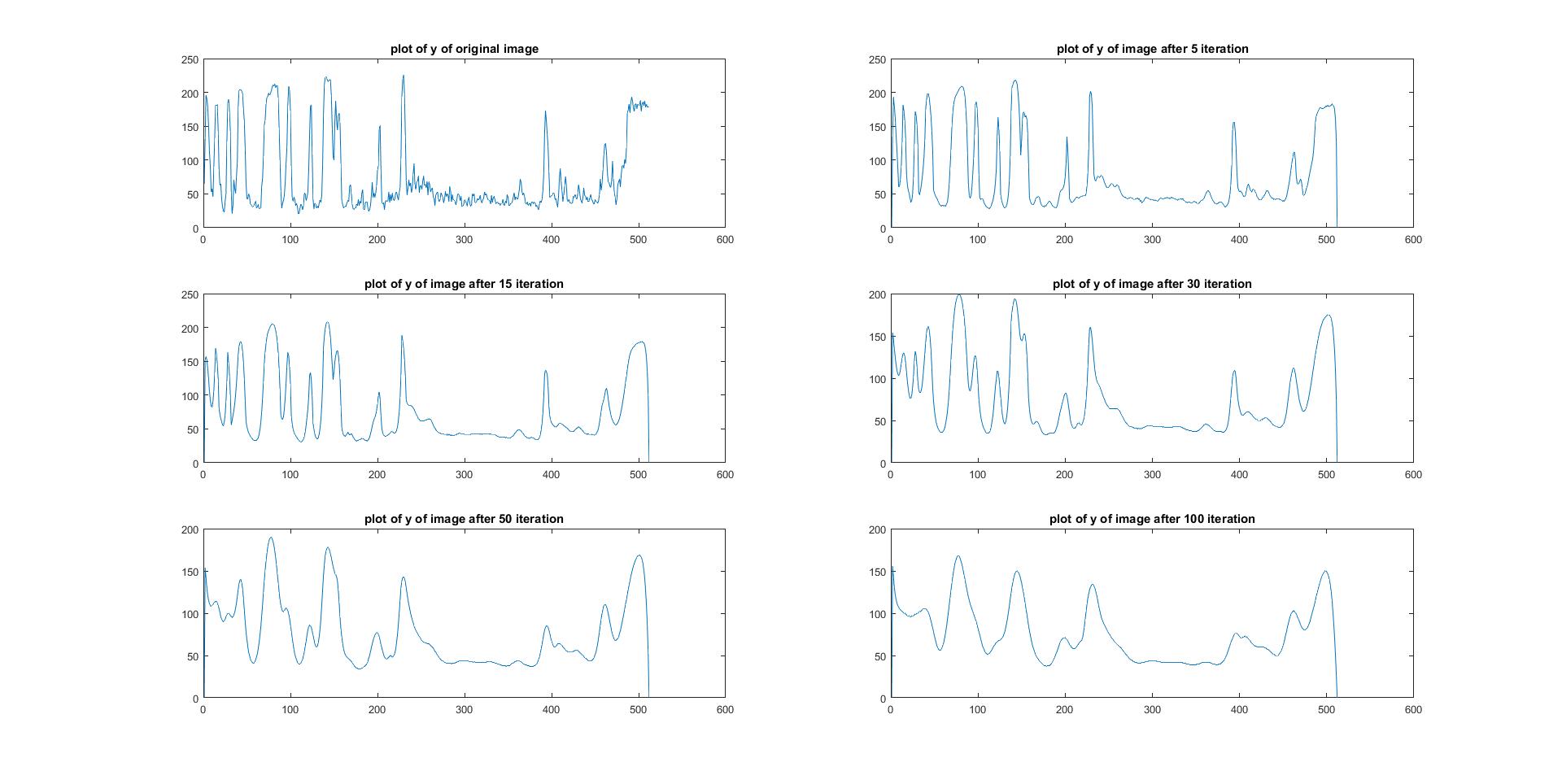


Figure 3.2.16 A plot of the line *y* = 128 through the image lake







Figure 3.2.17 results after applying anisotropic diffusion filter to lake K=25, g(.)=quadratic, Lambda=0.25

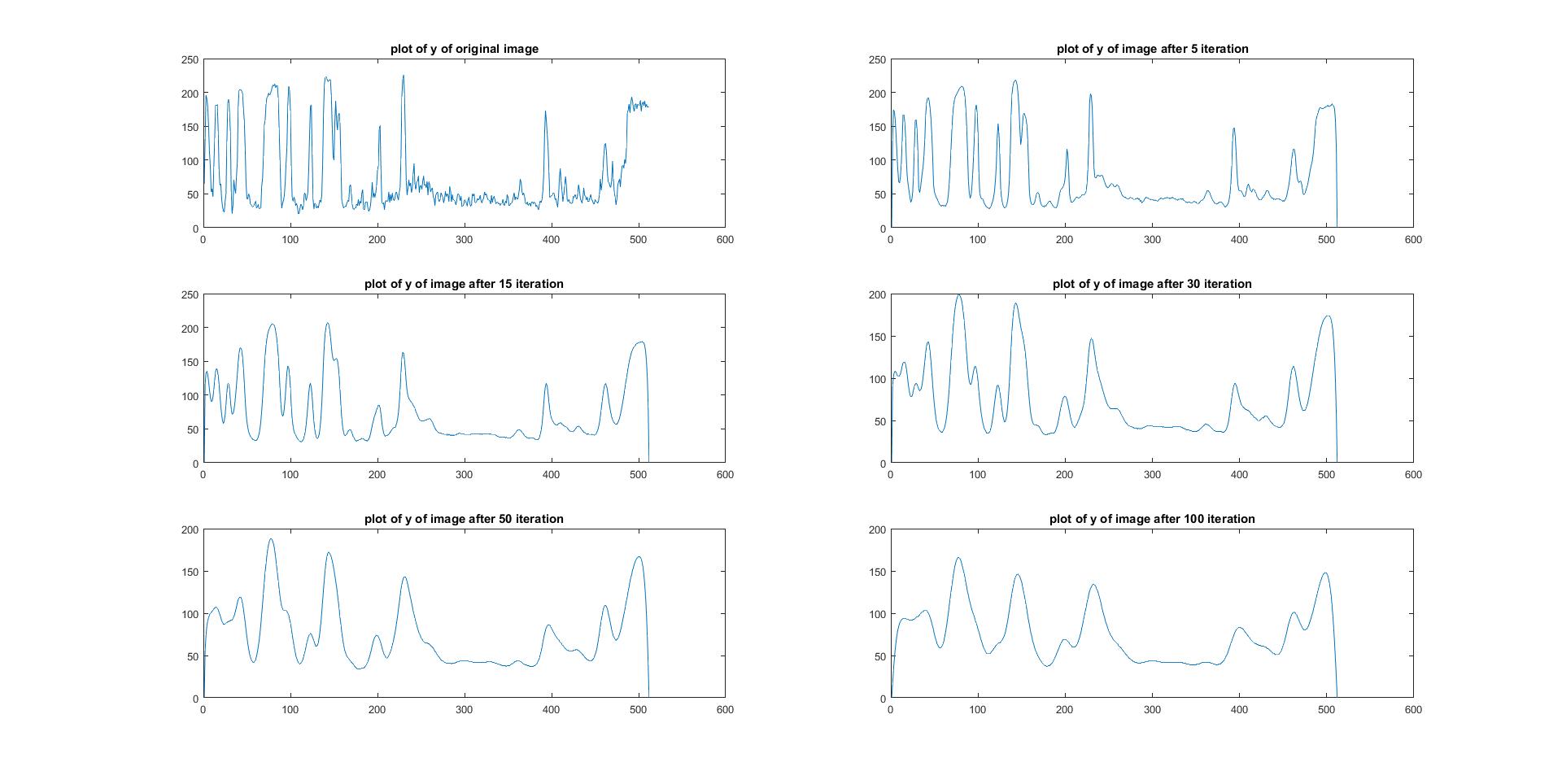
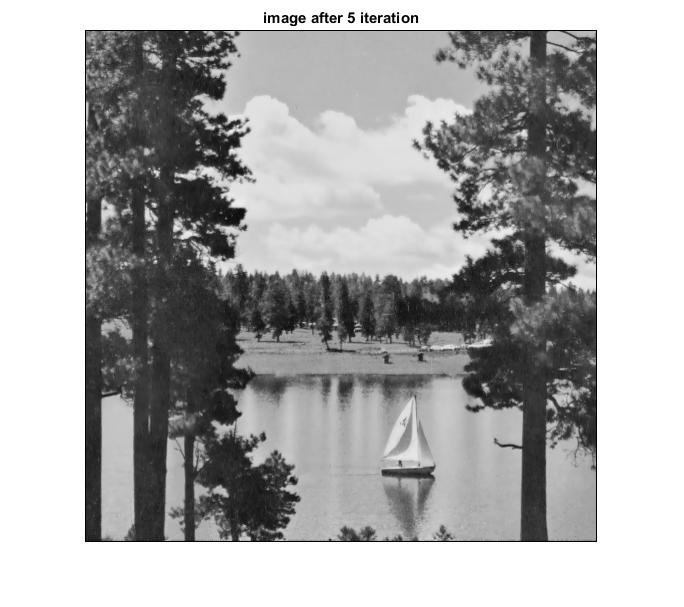
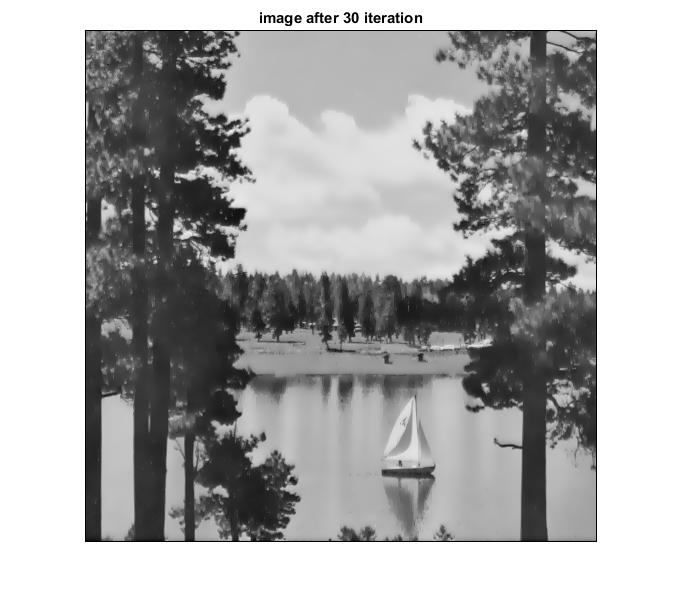
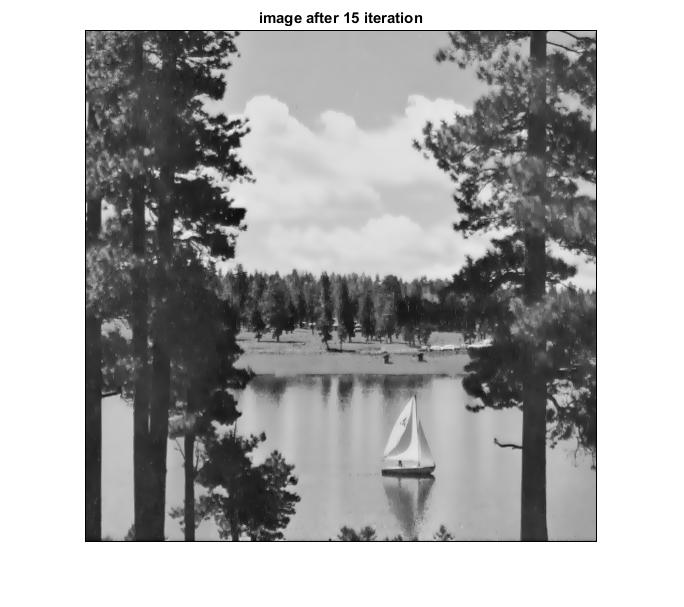


Figure 3.2.18 A plot of the line *y* = 128 through the image lake





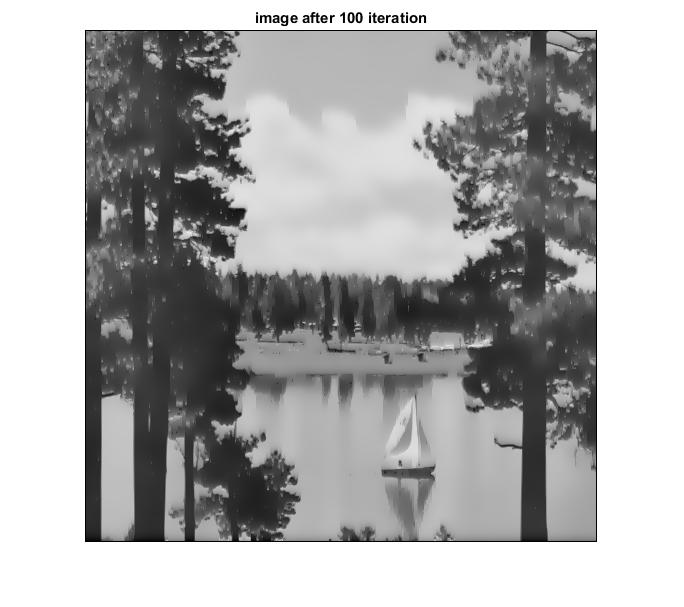
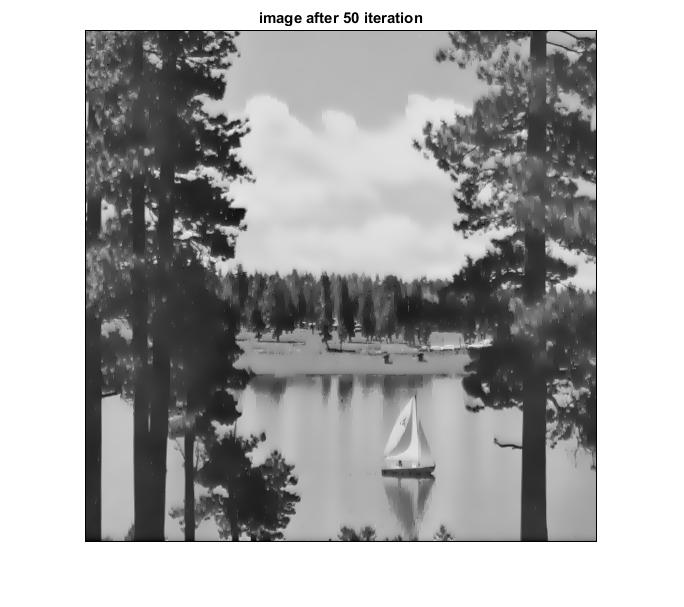


Figure 3.2.19 results after applying anisotropic diffusion filter to lake K=5, g(.)=exponential, Lambda=0.25

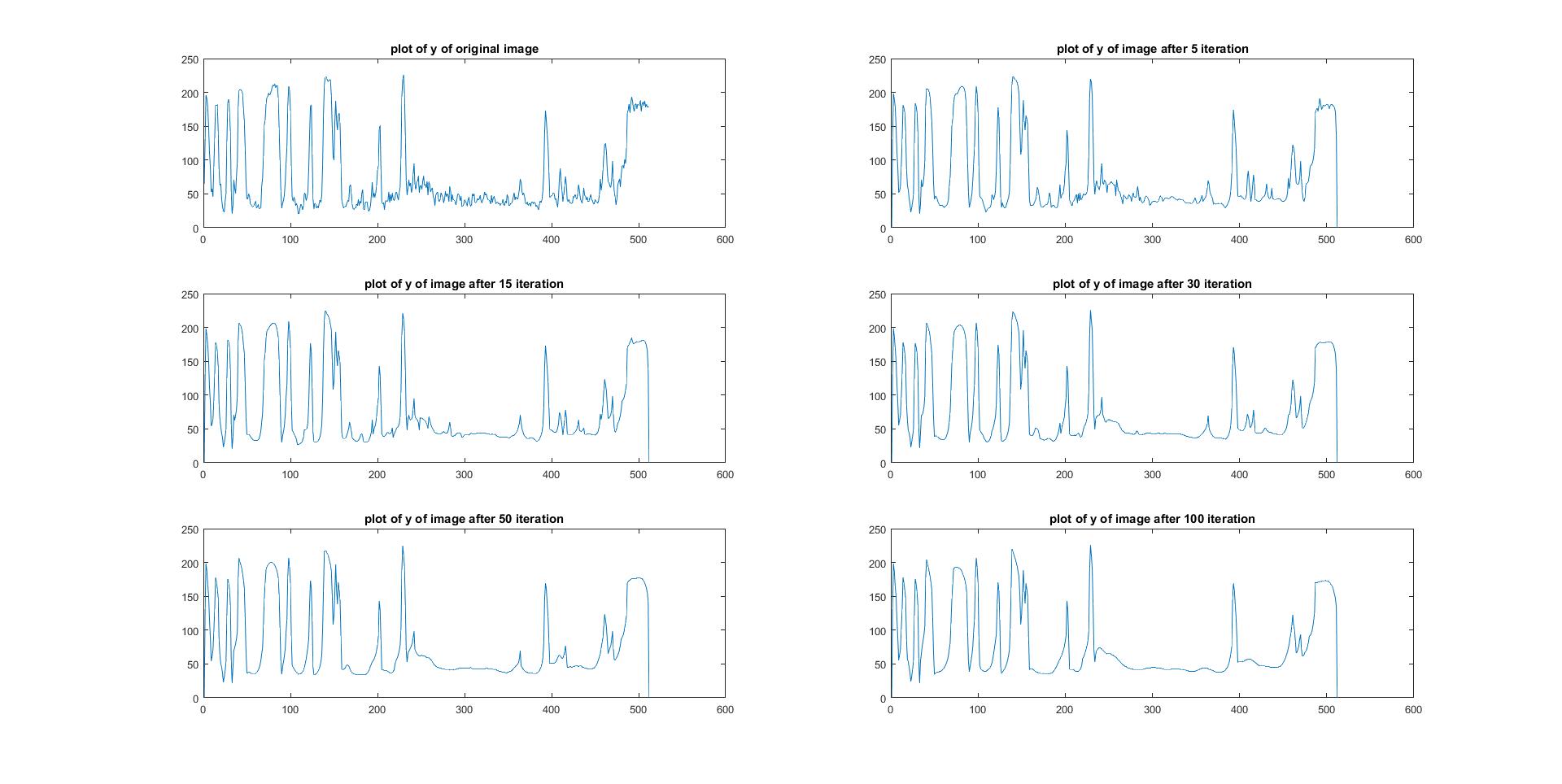
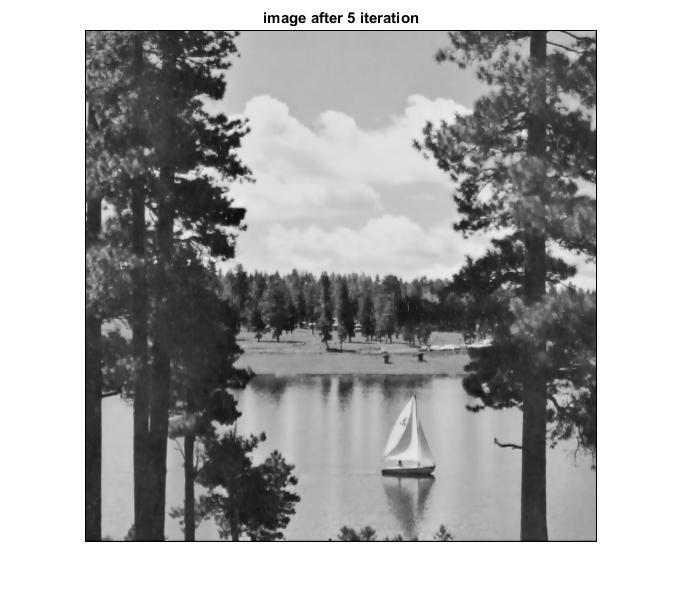


Figure 3.2.20 A plot of the line *y* = 128 through the image lake



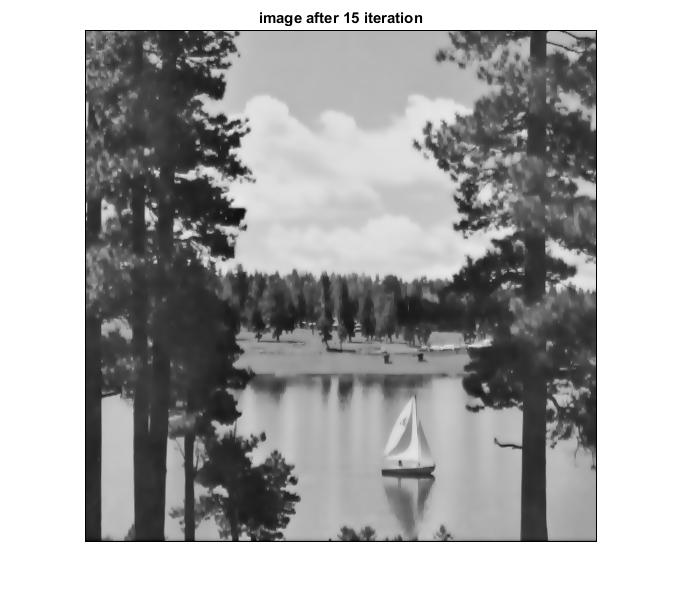




Figure 3.2.21 results after applying anisotropic diffusion filter to lake K=5, g(.)=quadratic, Lambda=0.25



Figure 3.2.22 A plot of the line *y* = 128 through the image lake

As the image is iterated the noise is removed but the image gets blurred in the process. However, with a proper usage of K the noise is removed and still the image can be distinctly viewed and analyzed. Also, it is possible to preserve few features from the image.

The anisotropic diffusion depends upon the value of K. With K greater than 10 it is possible to successfully filter the image. As K becomes less than 10 the diffusion decreases as observed with different value of g(.).

g(.) used in this project is exponential and quadratic. For same values of K both functions provide different results. Quadratic function gives a more diffused image as compared to the exponential function. Image after segmentation using quadratic function is better generated as compared to the exponential function due to better diffusion.

For a given value of K, cwheelnoise gives better results as compared to the aerial image. It is difficult to obtain image component by segmentation in aerial image. The image after 100 iteration of anisotropic diffusion is completely diffused. However, in case of cwheelnoise it is possible to better results get wheel components by image segmentation.

The plot of original aerial image is shown as below



Figure 3.3.1 Original image “aerial”

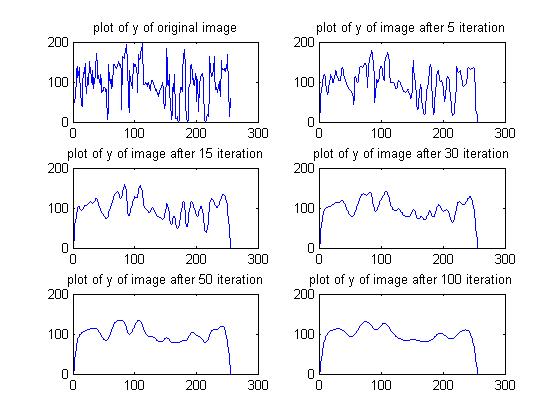


Figure 3.3.2 A plot of the line *y* = 128 through the image aerial

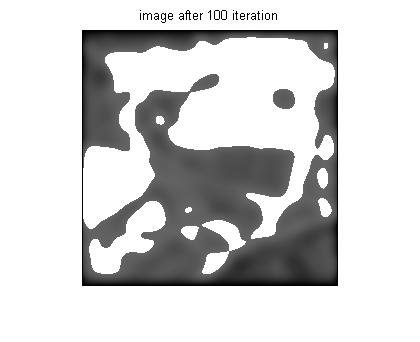


Figure 3.3.3 Segmentated Image after 100 iteration with g(.)=quadratic, k=25, lambda=0.25 at threshold = 100

* + 1. Conclusions

By comparing the results, we learned that nonlinear filters can clean out salt and pepper noise neatly, and preserve edges. We also learned that while anisotropic diffusion takes more time and iterations to achieve the same results compared to nonlinear filters. However, it can produce different results by changing the value of k.