E9-246 ADVANCED IMAGE PROCESSING

ASSIGNMENT 3: REPORT

NAME: HARSHIT DUBEY

SR. NO.: 04-03-06-10-51-23-1-23043

DEGREE: M.TECH AI

DEPARTMENT : COMPUTER SCIENCE AND AUTOMATION

Question 1:

As given in the question,

$$Y = X + Z \quad where,$$

X is distributed according to a Laplace distribution,

$$f_X(x) = \frac{1}{2\sigma_X} e^{-\frac{|x|}{\sigma_X}} \quad (\sigma_X = 1)$$

and Z is Gaussian white noise,

$$Z \sim \mathcal{N}(0, \sigma_Z^2), \quad (\sigma_Z^2 = 0.1)$$

Since MMSE is,

$$\hat{x} = E[X|Y] = \int x\, f_{X|Y}(x|y)\, dx$$

using bayes rule,

$$\hat{x} = E[X|Y] = \int x\, \frac{f_{Y|X}(y|x) f_X(x)}{f_Y(y)}\, dx$$

Here $f_{Y|X}(y|x)$ would be a gaussian distribution with mean x and variance as σ_Z^2, such that

$$f_{Y|X}(y|x) = \frac{1}{\sqrt{2\pi\sigma_Z^2}} \exp\left(-\frac{(y-x)^2}{2\sigma_Z^2}\right)$$

Now we have to solve the integration,

$$\hat{x} = \frac{\int x\, f_{X|Y}(x|y) f_X(x)\, dx}{\int f_{X|Y}(x|y) f_X(x)\, dx}$$

The integrals above involve the product of a Gaussian and a Laplacian density, since Gaussian and Laplacian belong to different families of distributions, and their product does not result in a standard distribution with a known closed-form expression.

While it is possible to them in terms of special functions or infinite series, these expressions are generally not considered closed-form solutions since they involve evaluating complex functions or summing infinite series.
Therefore we have to compute the MMSE estimate x(y) analytically.

Brief of Implementation:

The code given below performs numerical integration to approximate the MMSE estimate for a range of observed values (y) due to the lack of a closed-form solution.

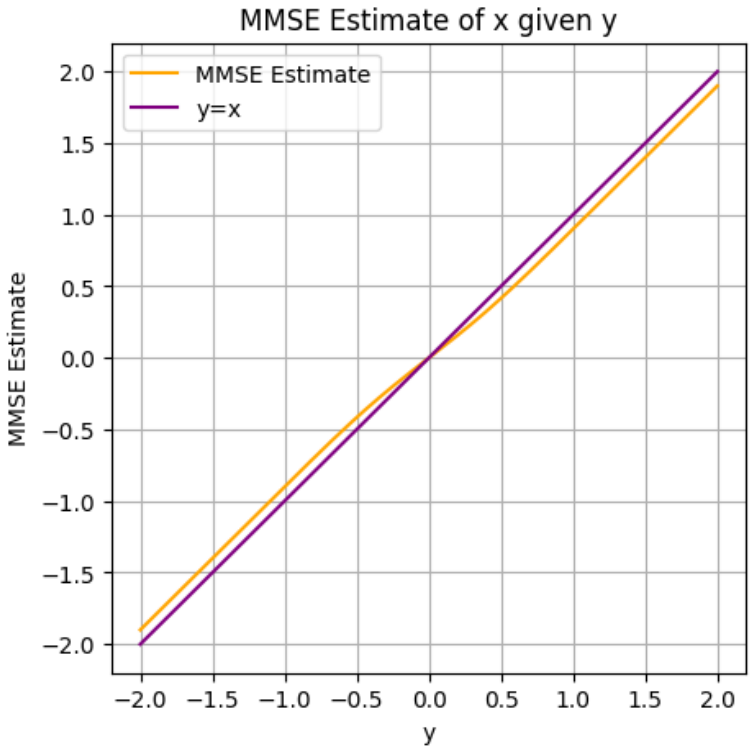
The code follows these steps:

- Define the parameters σ_X and σ_Z for the Laplace and Gaussian distributions, respectively.
- Define the joint density function g_joint_yx(x, y, σ_X, σ_Z) as the product of the Gaussian likelihood p(y|x) and the Laplacian prior p(x).
- Create a range of y values for which the MMSE estimate will be computed.
- For each y value, numerically compute the numerator and denominator of the MMSE estimate using numerical integration (quad function from scipy.integrate).
- Compute the MMSE estimate $\hat{x}(y)$ as the ratio of the numerator and denominator.

- Store the MMSE estimate values in a list.

Observation:

Output plot:



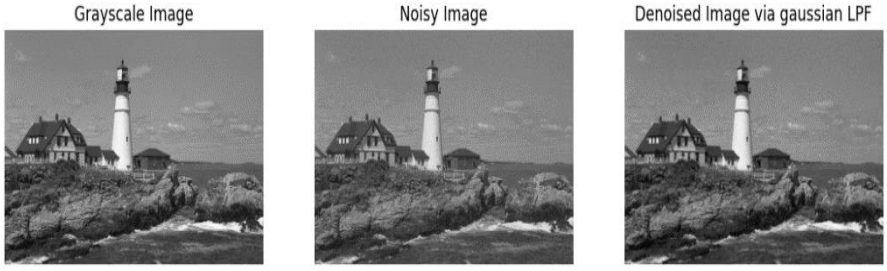
The output plot shows the MMSE estimate $\hat{x}(y)$ as a function of y. Since the source X follows a Laplacian distribution, which is heavier-tailed than the Gaussian, the MMSE estimate exhibits a non-linear behavior, especially for larger values of |y|. This non-linearity is due to the influence of the Laplacian prior, which favors sparse estimates for larger |y| values.

The similarity in shape between the obtained plot and the result reported in the reference paper by Simoncelli and Adelson (1996) provides confidence in the accuracy of the code-generated plot.

Question 2:

MSE corresponding to noisy image is 99.7934

Question 2a:



Brief of Implementation:

The code is designed to apply a Gaussian Low Pass Filter (LPF) to denoise an image and compute the Mean Squared Error (MSE) between the original and denoised images.

- The LPF_gaussian class is initialized with an original image and a noisy version of the same image.
- The MSE method computes the Mean Squared Error between two images.
- The compute_MSE method applies a Gaussian LPF to the noisy image using different kernel sizes and standard deviations, computes the MSE for each, and stores the results in a nested dictionary. It also keeps track of the minimum MSE along with the corresponding kernel size and standard deviation.

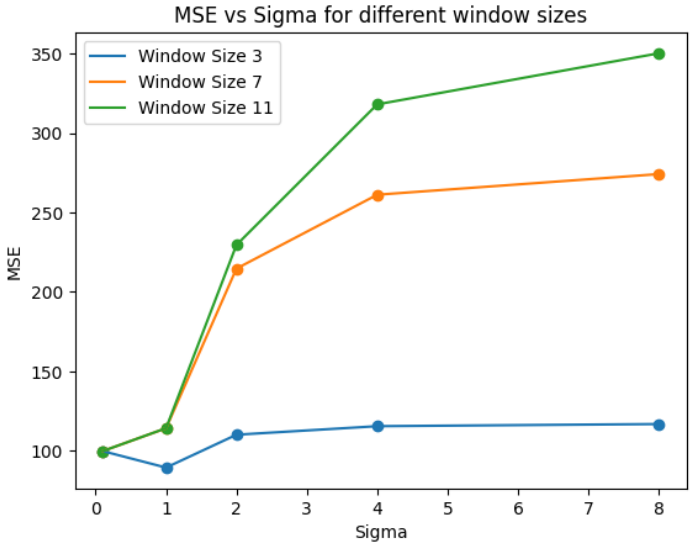
Observation:

MSE for different kernel sizes and standard deviations are given below:

```
For kernel size 3
Standard deviation is 0.1 and MSE is: 99.7934
Standard deviation is 1 and MSE is: 89.406
Standard deviation is 2 and MSE is: 110.109
Standard deviation is 4 and MSE is: 115.4819
Standard deviation is 8 and MSE is: 116.827
For kernel size 7
Standard deviation is 0.1 and MSE is: 99.7934
Standard deviation is 1 and MSE is: 114.1739
Standard deviation is 2 and MSE is: 214.7173
Standard deviation is 4 and MSE is: 261.2325
Standard deviation is 8 and MSE is: 274.2319
For kernel size 11
Standard deviation is 0.1 and MSE is: 99.7934
Standard deviation is 1 and MSE is: 114.2213
Standard deviation is 2 and MSE is: 229.6199
Standard deviation is 4 and MSE is: 318.1376
Standard deviation is 8 and MSE is: 350.2898

Minimum MSE observed is 89.406, with kernel size = 3 and standard deviation = 1
```

The minimum MSE of 89.406 is achieved with a kernel size of 3 and sigma of 1, indicating that this combination provides the best denoising performance while minimizing the deviation from the original image.



The graph shows the variation of Mean Squared Error (MSE) with respect to different sigma (standard deviation) values used for Gaussian low-pass filtering. Three different window/kernel sizes (3, 7, and 11) are plotted.

For a small kernel size of 3, the MSE initially decreases as sigma increases from 0.1 to 1, indicating better denoising performance. However, as sigma further increases beyond 1, the MSE starts to rise, suggesting over-smoothing and loss of image details.

On the other hand, for larger kernel sizes of 7 and 11, the MSE consistently increases with increasing sigma values. This implies that larger kernels tend to over-smooth the image, leading to higher deviations from the original image, even for small sigma values.

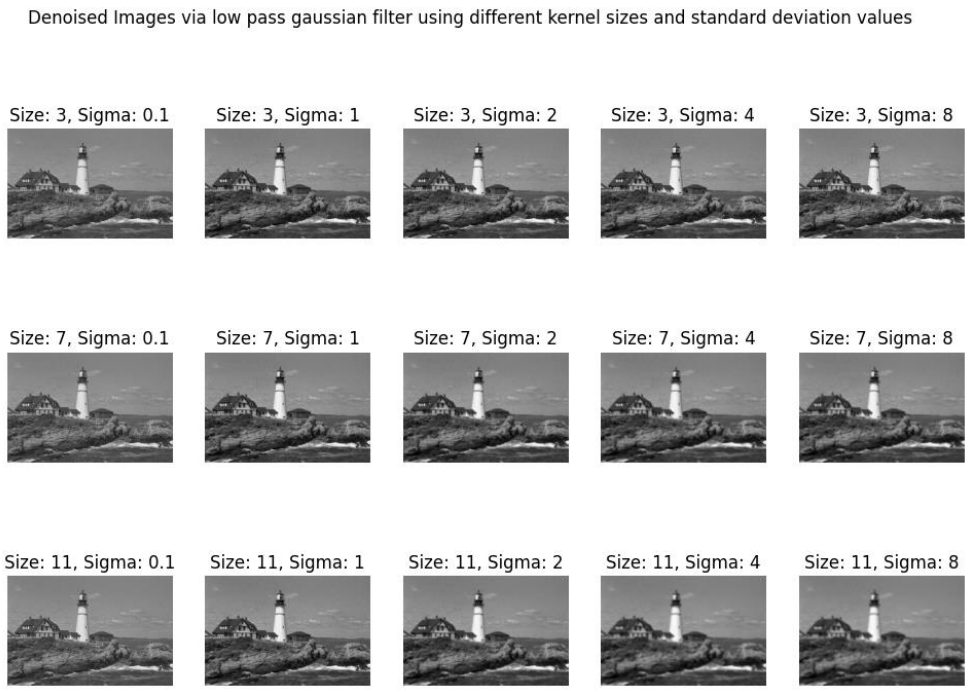


Image above visually demonstrates the denoised images obtained using different combinations of kernel sizes and standard deviations. It can be observed that:

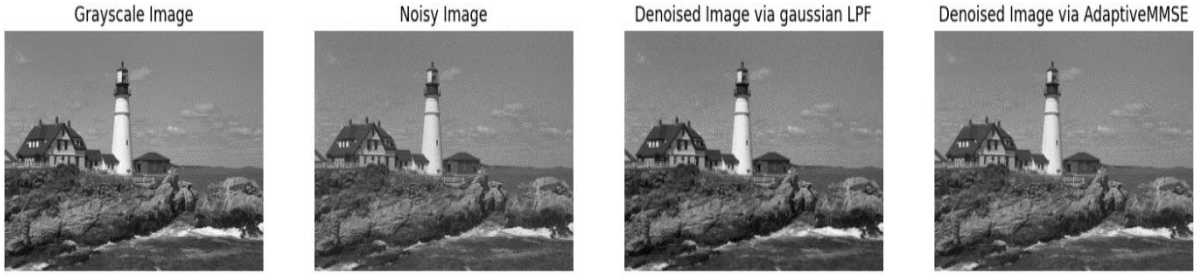
- For a kernel size of 3, the image denoised with sigma = 1 appears to strike a good balance between noise removal and detail preservation.
- As sigma increases for kernel size 3, the images become progressively over-smoothed, losing essential details.
- For larger kernel sizes (7 and 11), even small sigma values lead to significant over-smoothing and detail loss.

In summary, the analysis suggests that for the given noisy image, a smaller kernel size (e.g., 3) with a moderate sigma value (e.g., 1) is optimal for Gaussian low-pass filtering to effectively remove noise while preserving image details. Larger kernel sizes or higher sigma values tend to over-smooth the image, leading to increased deviations from the original image and loss of important details.

Question 2b:

Brief of Implementation:

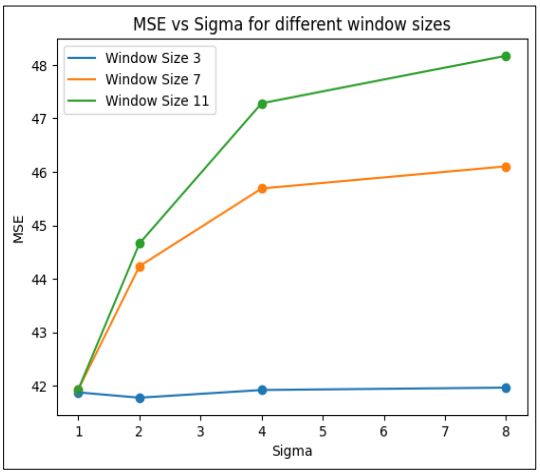
- The `AdaptiveMMSE` class in the code is designed to apply an Adaptive Minimum Mean Squared Error (MMSE) filter for image denoising. The class is initialized with several parameters, including the noisy image, filter size, standard deviation, noise variance, patch size, and step size.
- The class contains a method named `high_pass_filter`. This method applies a Gaussian blur to the noisy image, creating a low-pass filtered version. The original image is then subtracted from this low-pass image to create a high-pass filtered image. The Gaussian kernel used for the blur is also computed within this method.
- The class also includes a method named `compute_aMMSE`. This method applies the Adaptive MMSE filter to the high-pass filtered image. It iterates over the image in patches. For each patch, it computes the variances of the patch in the high-pass image and the noise. These variances are then used to compute the MMSE estimate of the high-pass image. This estimate is added to the low-pass image to produce the final denoised image.



Observation:

MSE for different kernel sizes and standard deviations are given below:

```
For kernel size 3
Standard deviation is 1 and MSE is: 41.8734
Standard deviation is 2 and MSE is: 41.7742
Standard deviation is 4 and MSE is: 41.9181
Standard deviation is 8 and MSE is: 41.9626
For kernel size 7
Standard deviation is 1 and MSE is: 41.9308
Standard deviation is 2 and MSE is: 44.2337
Standard deviation is 4 and MSE is: 45.6884
Standard deviation is 8 and MSE is: 46.1035
For kernel size 11
Standard deviation is 1 and MSE is: 41.9307
Standard deviation is 2 and MSE is: 44.6606
Standard deviation is 4 and MSE is: 47.2836
Standard deviation is 8 and MSE is: 48.1718
```



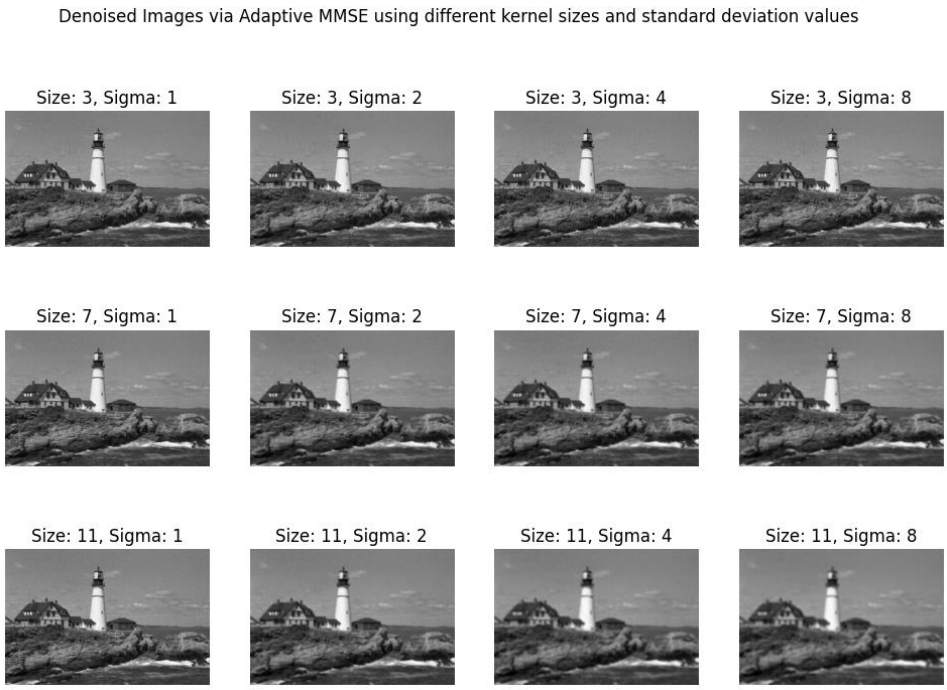
- For a kernel size of 3, the MSE values are relatively stable across different standard deviations, ranging from 41.7742 to 41.9626. This suggests that the denoising process is not significantly affected by the standard deviation in this case.
- As the kernel size increases to 7 and 11, the MSE increases across all standard deviations. This suggests that increasing the kernel size does not necessarily improve the denoising process; in fact, it appears to degrade the performance.
- Similarly, increasing the standard deviation also leads to higher MSE values, indicating that a lower standard deviation is preferable for this particular denoising task.
- The minimum MSE of 41.7742 is achieved with a kernel size of 3 and standard deviation of 2. This is the optimal combination as it results in the minimum MSE.

Problem with small standard deviation like 0.1:

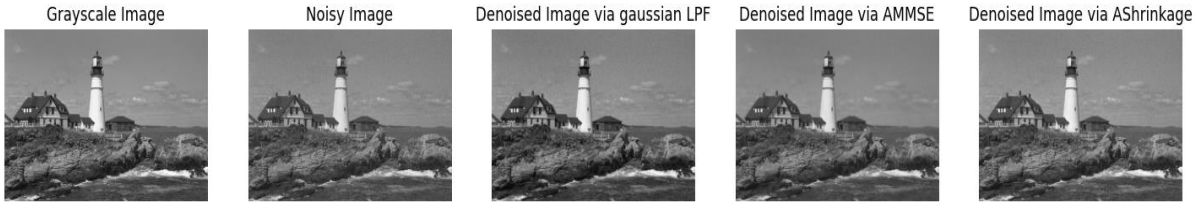
When the standard deviation is very small, the Gaussian filter becomes very narrow. This means that during the filtering process, the filter gives very high weight to the center pixel and almost zero weight to the surrounding pixels. This can lead to a situation where the filtered image is almost identical to the original image, leading to a very small or zero variance in the high-pass filtered image (y1).

In the compute_aMMSE method, the variance of the noise (var_z1) is computed based on the Gaussian kernel, which is also very narrow due to the small standard deviation. This can lead to a very small or zero noise variance.

The signal variance (var_x1) is then computed as the difference between the variances of y1 and var_z1. If both of these variances are very small or zero, var_x1 can also end up being very small, zero. This can cause problems in the subsequent computation of the MMSE estimate, leading to NaN values.



Question 2c:



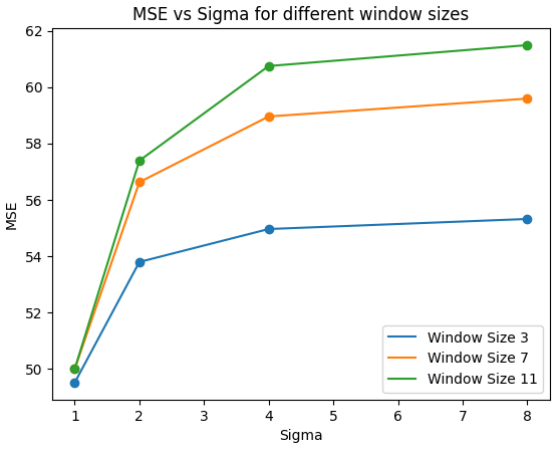
Brief of Implementation:

- The `AdaptiveShrinkage` class in the code is designed to apply an Adaptive Shrinkage filter to a noisy image for the purpose of image denoising.
- The class is initialized with several parameters including the noisy image, filter size, standard deviation, noise variance, patch size, and step size.
- Compute and Implement Threshold method calculates the optimal threshold for a given patch of the image and the noise variance. It then applies this threshold to the patch, shrinking the values towards zero.
- Compute Adaptive Shrinkage method applies the Adaptive Shrinkage filter to the noisy image. It first applies a Gaussian blur to the noisy image to create a low-pass filtered version and subtracts this from the original image to create a high-pass filtered image. It then iterates over the high-pass image in patches, applies the thresholding function to each patch, and combines these thresholded patches to create the final denoised image.

Observation:

MSE for different kernel sizes and standard deviations are given below:

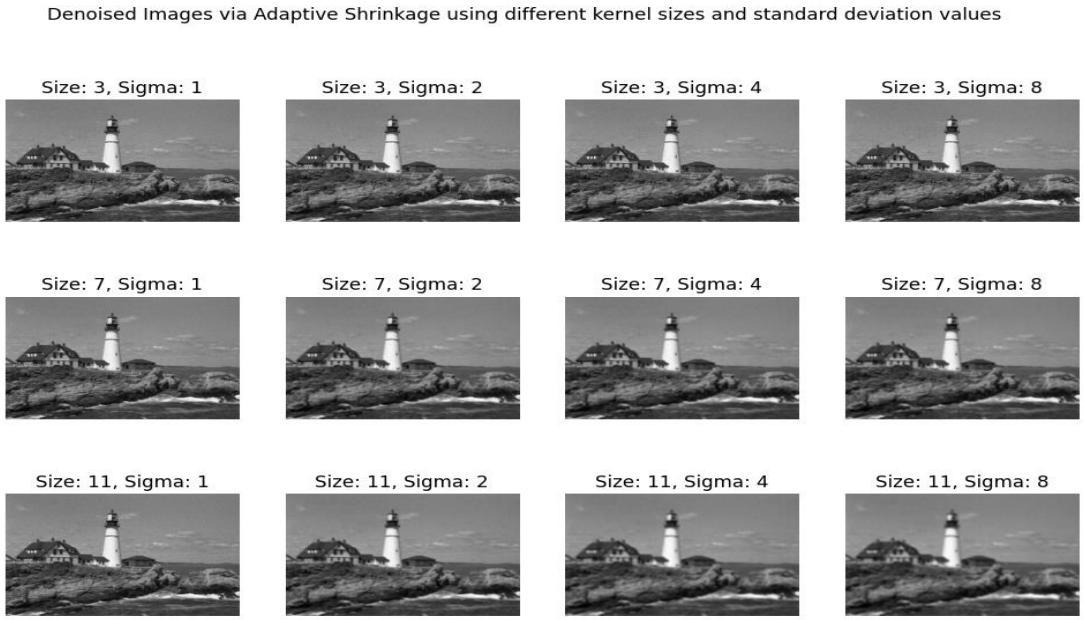
```
For kernel size 3
Standard deviation is 1 and MSE is: 49.508
Standard deviation is 2 and MSE is: 53.7911
Standard deviation is 4 and MSE is: 54.9609
Standard deviation is 8 and MSE is: 55.318
For kernel size 7
Standard deviation is 1 and MSE is: 49.9902
Standard deviation is 2 and MSE is: 56.625
Standard deviation is 4 and MSE is: 58.9565
Standard deviation is 8 and MSE is: 59.5912
For kernel size 11
Standard deviation is 1 and MSE is: 49.992
Standard deviation is 2 and MSE is: 57.3873
Standard deviation is 4 and MSE is: 60.7473
Standard deviation is 8 and MSE is: 61.4892
```



- For a kernel size of 3, the MSE values increase as the standard deviation increases from 1 to 8. This suggests that a lower standard deviation is preferable for this kernel size.

- As the kernel size increases to 7 and 11, the MSE increases across all standard deviations. This suggests that increasing the kernel size does not necessarily improve the denoising process; in fact, it appears to degrade the performance.
- Increasing the standard deviation results in a higher Mean Squared Error (MSE).
- The minimum MSE of 49.508 is achieved with a kernel size of 3 and standard deviation of 1. This is the optimal combination as it results in the minimum MSE.

In conclusion, a smaller kernel size and a lower standard deviation result in a lower MSE, making them more suitable for the denoising process in this case. However, these parameters might vary depending on the specific characteristics of the images and the noise. Therefore, it's always a good idea to experiment with different parameters to find the optimal combination.



Comparison between all 3 methods:

Gaussian Low Pass Filter (LPF):

- Generally has higher MSE values compared to AMMSE and Adaptive Shrinkage for all kernel sizes and standard deviations explored.
- Shows a trend of increasing MSE with increasing standard deviation, indicating more blurring and potentially worse noise reduction.
- Might not be the most effective method for this denoising task based on the MSE metric.

Adaptive Minimum Mean Squared Error (MMSE):

- Achieves consistently lower MSE values compared to Gaussian LPF across all kernel sizes and standard deviations.
- Shows minimal variation in MSE with changing standard deviation, suggesting robustness to filter parameter selection.
- Potentially a good choice for balancing noise reduction and detail preservation based on the MSE metric.

Adaptive Shrinkage:

- Has slightly higher MSE values than AMMSE but still significantly lower than Gaussian LPF.
- Similar to AMMSE, the MSE variation with standard deviation is minimal.
- Could be a viable alternative to AMMSE based on the MSE metric.

Note:
In theory, shrinkage methods have the potential to deliver superior results due to their adaptability to the specific characteristics of the image and noise. However, the practical effectiveness of these methods (as observed in the code) can be influenced by several factors:

1. Parameter Selection: The performance of both Adaptive Minimum Mean Squared Error (AMMSE) and shrinkage methods can be significantly influenced by the parameters chosen, such as the kernel size and standard deviation for the Gaussian filter. If these parameters are not optimally selected to suit the specific characteristics of the image and noise, it could adversely affect the performance of the denoising methods.

2. Noise Characteristics: The type and intensity of noise present in the image can also impact the performance of the denoising methods. Certain methods might be more effective for specific types of noise or noise levels.

3. Image Characteristics: The unique characteristics of the image, such as the amount and type of detail, can also influence the performance of the denoising methods. Some methods might be more suitable for certain types of images.

Overall:

- From an MSE perspective, AMMSE seems to be the most effective method for this image denoising task.
- The impact of standard deviation on MSE appears to be more significant for the Gaussian LPF method compared to AMMSE and Adaptive Shrinkage. This reinforces the idea that AMMSE and Adaptive Shrinkage might be more robust choices for denoising this particular image.

Note: All three denoising methods (Gaussian LPF, AMMSE, Adaptive Shrinkage) significantly improve image quality compared to the original noisy image.