Data-Driven Analysis of Fluid Flows

FLUID MECHANICS

MEL2040

Authors:

Sonal Raj (B22ME062)
Sanjay Bijarniya (B22ME057)

April 15, 2024

Contents

Introduction

In this project, we are exploring machine learning algorithms in the field of fluid mechanics. By combining both fields ML and fluid, we can develop models that can predict fluid behavior in complex systems, such as turbulent flows, where traditional analytical methods may not be feasible. Our project will not only demonstrate the promise of machine learning in fluid mechanics but will also shed light on the obstacles and limitations of these techniques. We hope that this effort will spark additional research in this fascinating interdisciplinary topic.

1 Review of Machine Learning in Fluids

Fluid mechanics, being a domain abundant in data and intricate issues, offers an optimal setting for harnessing machine learning techniques. Machine learning utilises optimisation and regression methods to construct models from data, providing solutions to a range of difficulties in fluid mechanics. These issues encompass tasks such as enhancing aerodynamic designs, approximating flow fields based on restricted observations, and managing turbulence to enhance efficiency in diverse applications.

Machine learning is not universally applicable; instead, it necessitates the involvement of experts at each step of the process. Human expertise is essential in every step of machine learning applications in fluid mechanics, including problem selection, data curation, model architecture design, loss function formulation, and optimisation algorithm implementation.

The utilisation of machine learning in fluid mechanics often follows a series of established steps, each of which presents possibilities for integrating existing physical knowledge. The processes involve the selection of a problem to model, the curation of relevant data, the choice of suitable machine learning architectures, the design of effective loss functions, and the implementation of optimisation algorithms.

An essential component of machine learning in fluid mechanics is the careful and deliberate process of choosing and organising data. The performance of the machine learning model is highly influenced by both the quality and amount of the data. Different machine learning architectures, such as deep neural networks, depend on a variety of training data to accurately apply their knowledge to new situations.

Machine learning architectures are essential for representing and modelling the training data. Neural networks are frequently employed in fluid mechanics applications due to their capacity to approximate intricate functions. Various neural network topologies, such as convolutional networks for systems that exhibit translational invariance and recurrent networks for systems that evolve over time, are designed to address specific problem domains.

Physically-informed neural networks (PINNs) are a specific type of topologies that include governing physical rules into the process of supervised learning. These models integrate machine learning with physics-based limitations to guarantee more precise and scientifically meaningful answers.

The loss function is an essential element in machine learning for fluid mechanics as it measures model performance and directs the optimisation process. Integrating past knowledge about the physical properties into the loss function aids in mitigating overfitting and encourages the selection of simpler model solutions.

Optimisation techniques are employed to train machine learning models, typically in spaces that have a high number of dimensions and are not convex. Stochastic gradient descent methods are frequently used to efficiently navigate these intricate optimisation environments.

Researchers are constantly developing custom optimisation algorithms and loss functions that are specifically designed for fluid mechanics problems. These advancements aim to improve the performance and accuracy of models used in this field. These methods frequently integrate physical limitations and prior information to enhance the resilience of machine learning models.

Machine learning has the potential to advance our understanding of fluid mechanics by using data-driven methods. Researchers can create accurate models of fluid systems by integrating expert knowledge with machine learning approaches. This interdisciplinary approach shows potential for addressing long-standing difficulties and promoting innovation in the field of fluid mechanics research.

SVD through examples.

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ -1 & 1 \end{bmatrix}$$

$$U = V\Sigma V^T$$

Consider the matrix A^TA:

$$A^TA = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

The characteristic equation (C.E.) of A^TA is given by:

$$\lambda^2 - S_1\lambda + S_2 = 0$$

Where S_1 is the trace of A^TA:

$$S_1 = \text{trace}(A^TA) = 2 + 3 = 5$$

And S_2 is the determinant of A^TA:

$$S_2 = \det \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} = 2 \cdot 3 - 0 \cdot 0 = 6$$

So the characteristic equation becomes:

$$\lambda^2 - 5\lambda + 6 = 0$$

Solving for λ:

$$\lambda = 3, 2$$

For $\lambda = 2$, let $\mathbf{X_2} = \begin{bmatrix} a \\ b \end{bmatrix}$ be the eigenvector of A^TA.
Then $(A^TA)\mathbf{X_2} = \lambda\mathbf{X_2}$:

$$\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 2a \\ 2b \end{bmatrix}$$

Simplify the equation:

$$2a = 2a$$

$$0a + 3b = 2b$$

From Equation 2:

$$0a + b = 0$$

Setting $a = 1$ and $b = 0$, we obtain:

$$\mathbf{X_2} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Now $N(\mathbf{X_1}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $N(\mathbf{X_2}) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.
Therefore, $\mathbf{V} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.
And the transpose of $\mathbf{V}$, denoted as $\mathbf{V^T}$, is given by:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Step 2: Compute Σ.
The order of Σ is equal to the order of A.
Now, $\sigma_1 = \sqrt{\lambda_1} = \sqrt{3}$ and $\sigma_2 = \sqrt{\lambda_2} = \sqrt{2}$.
The number of non-zero eigenvalues is equal to the rank of the matrix, which is 2.
Therefore, Σ is a diagonal matrix whose diagonal entries are σ_1 and σ_2:

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\Sigma = R \times R = 2 \times 2$$

$$\Sigma = \begin{bmatrix} \sqrt{3} & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \end{bmatrix}$$

Step 3: Compute $\mathbf{U}$.
Consider the matrix AA^T:

$$AA^T = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Let $\lambda^3 - S_1\lambda^2 + S_2\lambda - S_3 = 0$ be the characteristic equation (C.E.) of AA^T.
The coefficients are calculated as follows:

$$S_1 = 2 + 1 + 2 = 5$$

$$S_2 = 1 + 4 + 1 = 6$$

$$S_3 = 2 \cdot 1 - 1 \cdot 2 = 0$$

Thus, the characteristic equation becomes:

$$\lambda^3 - 5\lambda^2 + 6\lambda - 0 = 0$$

This can be factored into:

$$(\lambda - 2)(\lambda^2 - 3\lambda) = 0$$

So, $\lambda = 3, 2, 0$.

For $\lambda = 3$, let $\mathbf{X_1} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ be the eigenvector of AA^T.

Then $(AA^T)\mathbf{X_1} = \lambda \cdot \mathbf{X_1}$:

$$\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 3 \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Simplify the equations:

$$2a + b + 0c = 3a \quad \text{Eq. 1}$$

$$a + b + c = 3b \quad \text{Eq. 2}$$

$$b + c = 3c \quad \text{Eq. 3}$$

From these equations, we can solve for $\mathbf{X_1}$:

$$\mathbf{X_1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Normalizing $\mathbf{X_1}$, we obtain:

$$N(\mathbf{X_1}) = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix}$$

— Eq. a
Similarly, we find $\mathbf{X_2}$ and $\mathbf{X_3}$:

$$\mathbf{X_2} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Normalizing $\mathbf{X_2}$:

$$N(\mathbf{X_2}) = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ -1/\sqrt{2} \end{bmatrix}$$

— Eq. b

$$\mathbf{X_3} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

Normalizing $\mathbf{X_3}$:

$$N(\mathbf{X_3}) = \begin{bmatrix} \frac{1}{\sqrt{6}} \\ -2 \\ \frac{1}{\sqrt{6}} \end{bmatrix}$$

— Eq. c

From equations a, b, and c, we obtain the matrix $\mathbf{U}$:

$$\mathbf{U} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \end{bmatrix}$$

2 Any New Ideas

2.1 Physics-guided machine learning for turbulence modeling

Turbulence models can be improved by machine learning by applying physical restrictions and principles. This approach combines the advantages of physics-based models and data-driven methodologies, resulting in more accurate and efficient simulations. A recent study on this topic is "Physics-guided machine learning using simplified theories."

Turbulence modeling in PGML involves incorporating physical knowledge into deep learning models in a structured manner. This is accomplished by designing physics-informed deep learning models that break down turbulent flow into distinct scale components, with trainable modules for simulating each component. For instance, the TF-Net framework combines a popular CFD technique, RANS-LES coupling, with custom-designed deep neural networks, resulting in substantial improvements in prediction accuracy and energy spectrum compared to baseline models. This method not only reduces error in predictions but also generates physically meaningful fields that adhere to essential physical properties such as mass conservation, accurately emulating the turbulent kinetic energy field and spectrum, which are crucial for accurate turbulent flow prediction.

2.2 Data-driven prediction of enhanced surface performance

Machine learning can forecast the performance of improved surfaces in fluid mechanics applications. Machine learning algorithms can identify and optimize the most promising designs by analyzing data from a large parametric search space. A recent study on this topic is "Data-driven prediction of the performance of enhanced surfaces from an extensive CFD-generated parametric search space".

In addition to predicting the performance of enhanced surfaces, there is also interest in developing surrogate models for the rapid performance prediction of novel enhanced microsurfaces. These models can be used to explore singular morphologies and add geometrical diversity to the search space, which is usually limited to a simplified subset of basic shapes to reduce complexity and dimension.

2.3 Nonlinear mode decomposition with convolutional neural networks

Machine learning can be used to separate fluid flow data into nonlinear modes, allowing for more efficient and accurate simulation. Convolutional neural networks can be used to detect spatial and temporal patterns in data, resulting in more accurate predictions and simulations. A recent publication on this topic is "Nonlinear mode decomposition with convolutional neural networks for fluid dynamics".

A mode-decomposing CNN autoencoder (MD-CNN-AE) has also been proposed, which is a CNN structure that can decompose flow fields in a nonlinear manner and visualize the decomposed fields. The MD-CNN-AE has been applied to flow around a circular cylinder at ReD=100, and the flow field was mapped into two values

and restored by adding the two decomposed fields. The two decomposed fields contain multiple POD modes, and the characteristics of higher modes are retained, which results in a lower reconstruction error than for the POD with the first two modes only.

2.4 Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations

Machine learning can be used to infer velocity and pressure fields in fluid flows from visuals, allowing for more accurate and efficient simulations. This method can be used to extract information from photos and videos of fluid flows, resulting in more accurate predictions and simulations. "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations" is a recently published work on this topic.
The proposed algorithm achieves accurate predictions of pressure and velocity fields in both two and three-dimensional flows for several benchmark problems motivated by real-world applications, demonstrating its potential for extracting valuable quantitative information (e.g., lift and drag forces or wall shear stresses in arteries) for which direct measurements may not be possible. HFM has been successfully applied to various physical and biomedical problems, including the analysis of magnetic resonance imagery of blood flow through a brain aneurysm to compute the stress placed on an arterial wall, the calculation of wind speeds and barometric pressures from satellite imagery of hurricanes, and the prediction of crack propagation in structures using ultrasound data and equations describing the mechanical behavior of materials.

3 Proper Orthogonal Decomposition

3.1 Image Generation

We have been given a video of flow over three cylinders. Firstly we need to extract the frames from the video, for that we used a computer vision library OpenCV.
Using this library we have extracted each 10^{th} frame from the video. So total we got 75 images and these images show different flow patterns at different intervals during the flow.(see Figure 1)

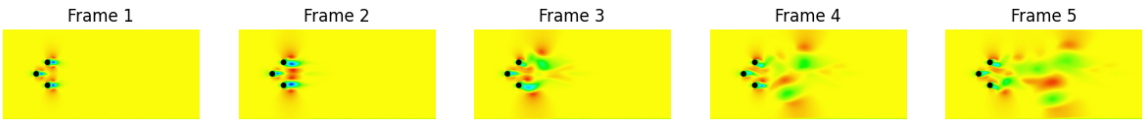

Figure 1: Some extracted frames from video

3.2 Execute POD

POD includes many steps, those are:

- Loading Images

  Initially, images that we get from extracting frames from the video (ref: 3.1) are saved in a folder in .jpg format. Therefore, we need to convert these images from .jpg format to arrays of pixels.

  Note: Dimension of each image = (708$\times$1558) Refer the code file.

- Stacking Images

  After the images are converted to a pixel array. we need to stack those images array into a single array, Row as images and columns as features of images.

  Note: Dimension of stacked image = (75$\times$1103064).Where 75 is the number of images and 1103064 is the number of pixels. Refer to the code file.

- Mean image

After stacking images, we need to find the mean image. The mean image can be found by calculating the mean of all features related to all images.

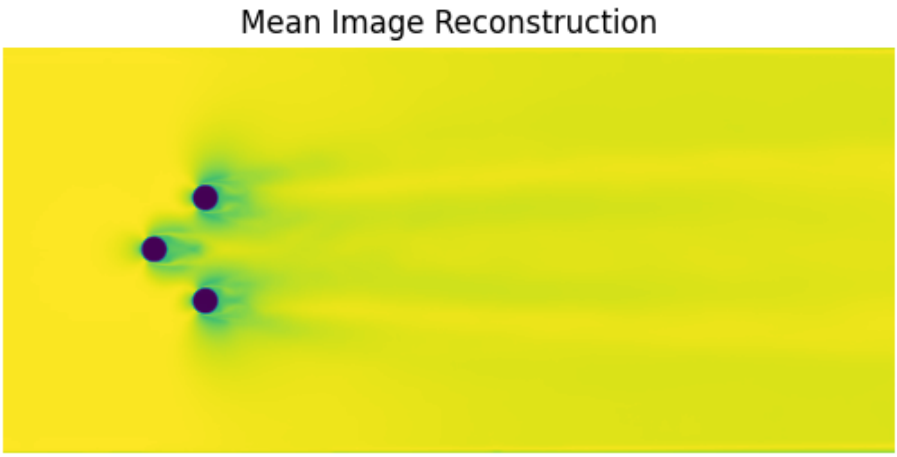

Figure 2: Mean image

This shows showing image reconstructed using the mean values of all features.

- Mean subtracted image

  After calculating the mean image, we need to make each image mean subtracted images. This task can be done simply by subtracting each image feature from the mean feature respectively.

$$MeanSubtractedImage_i = Image_i - meanImage \tag{1}$$

  Making our dataset mean-centric helps in various aspects like removing noise, improving convergence, and better comparison.

- SVD decomposition

  For the dimension reduction technique, we have used Singular Value Decomposition. In this method, we decompose our dataset (Image Matrix) into three matrices: U, Σ, and V^T.

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}$$

$$m = 75$$

$$n = 1103064$$

$$A_{75\times 1103064} = U_{75\times 75}\Sigma_{75\times 1103064}V^T_{1103064\times 1103064}$$

  Matrix U: Represents spatial modes or patterns of the flow.

  Matrix Σ: Contains singular values, indicating the importance of variability of each mode.

  Matrix V^T: Represents temporal modes or dynamics of the flow.

  Note: Refer section 1 for mathematical intitution.

- Cumulative variance ratio

  The cumulative variance ratio allows us to determine how much of the overall variance in the data is explained by each primary component. It determines the number of primary components to keep during dimensionality reduction, ensuring that we capture a significant portion of the variation while simplifying the data.

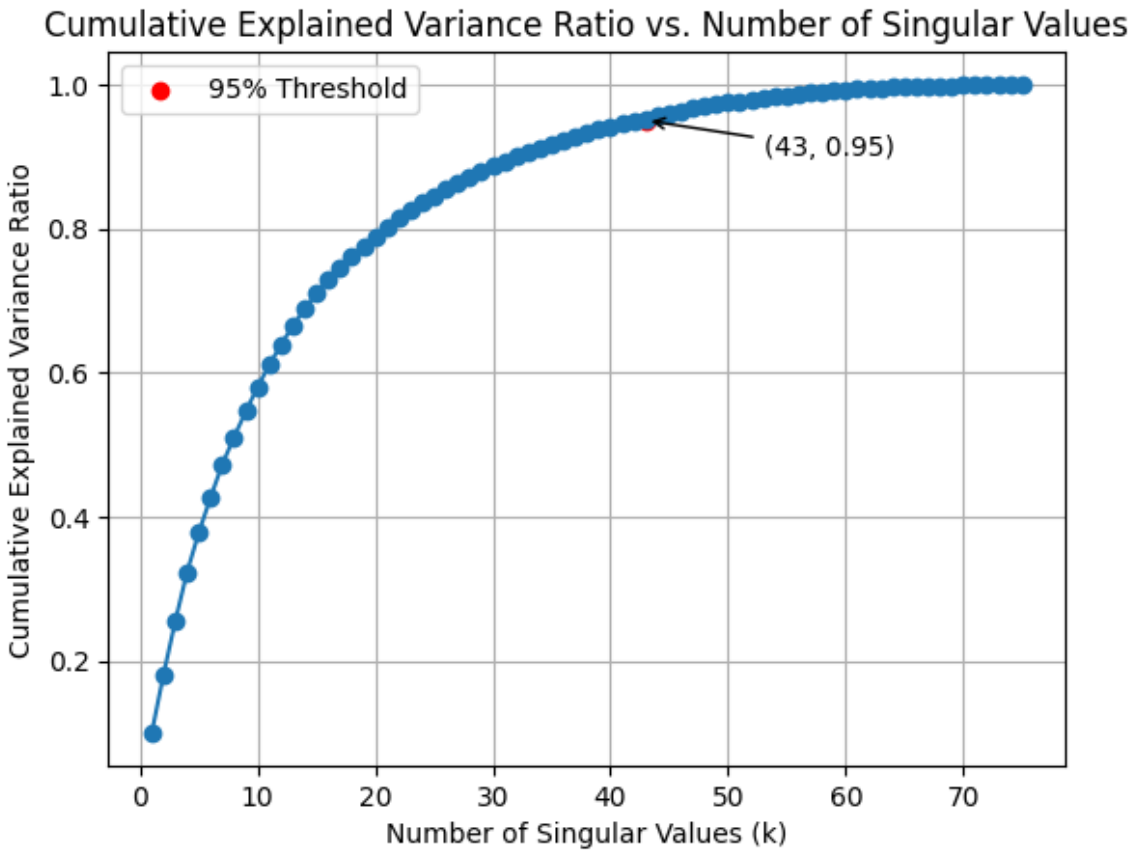

Figure 3: Cumulative variance ratio Vs K

- Dimension reduction

  In Figure 3 Cumulative variance ratio above 95% is shown for K = 43. It means 43 components are sufficient to get 95% accuracy in our computation. This reduces the computation cost. So dimension reduction is a very useful technique for complex and large data.

Listing 1: Code for reducing dimension

```
def reduce_dimensions(U, S, Vt, k):
    U_reduced = U[:, :k]
    print('Shape of U after dim reduction ',U_reduced.shape)
    S_reduced = np.diag(S[:k])
    print('Shape of S after dim reduction ',S_reduced.shape)
    Vt_reduced = Vt[:k, :]
    print('Shape of Vt after dim reduction ',Vt_reduced.shape)
    return U_reduced, S_reduced, Vt_reduced

U_reduced, S_reduced, Vt_reduced= reduce_dimensions(U, S, Vt, k_threshold)
```

After reducing dimension our reduced dimension will become:

$$A_{75\times1103064} = U_{75\times43}\Sigma_{43\times43}V^T_{43\times1103064}$$

3.3 Analyse POD Modes

After reducing the dimension, we now reconstruct the image from our reduced dimension.
Here are 10 images of the top modes possessing the highest energy.

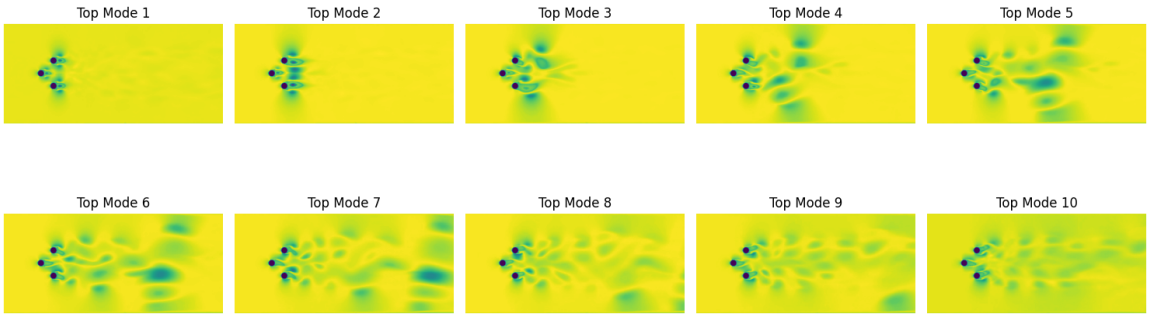

Figure 4: Top modes

- Total energy related to flow:

$$E_{\text{Total}} = \sum_{i=1}^{n} S_i^2 \tag{2}$$

- Total energy of top 10 modes:

$$E_{10} = \sum_{i=1}^{10} S_i^2 \tag{3}$$

These 10 modes posses about 60.88% energy of total energy.[Refer code file]

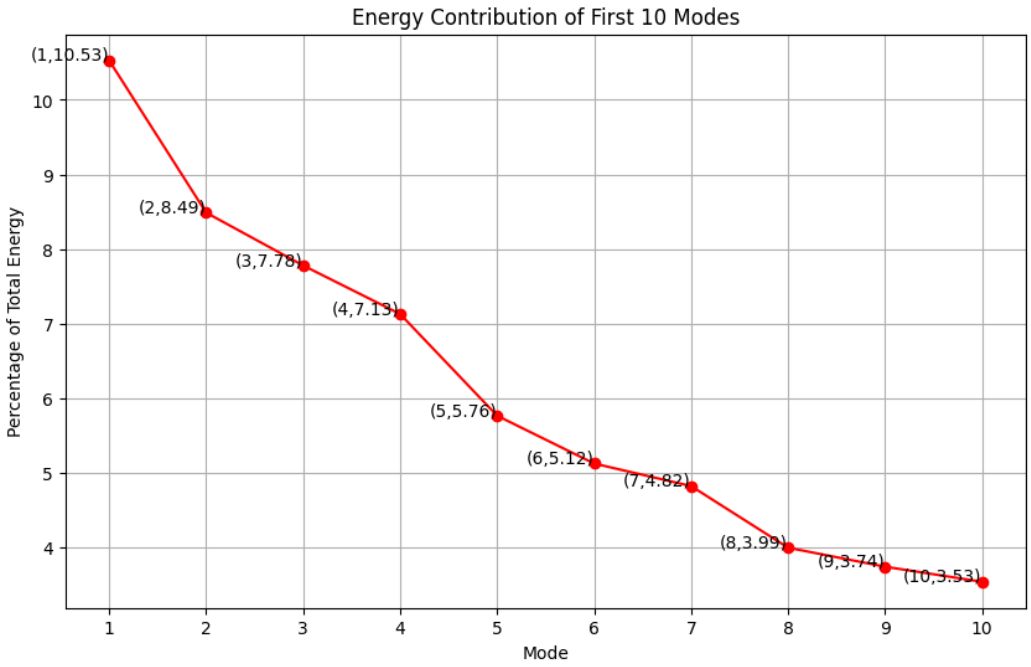

Figure 5: Energy of each mode

As we can see in Figure 5 the energy of the first mode is 10.53%, the second mode is 8.49% and so on goes decreasing showing that the flow that has the highest energy is the first flow mode. As you can see using this technique we came to know the highest energy-contributing flow. Flow modes help identify dominant patterns or modes of variability in the data.

Cumulative Variance Ratio	Energy Contribution by 10 modes.
43	60.88%

Table 1: Table for POD on real modes

4 Noise

4.1 Adding Noise

Adding Gaussian Noise

Gaussian noise, represented by the Gaussian distribution, is a bell-shaped curve with a defined mean and variance. It is widely used in signal processing and statistical analysis to simulate random fluctuations in data. This noise type is additive, contributing separately at each point, and is used in a variety of fields due to its ability to model real-world uncertainty.

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- x is the random variable representing the noise,
- μ is the mean of the distribution,
- σ^2 is the variance (or standard deviation squared).

$$I(x,y) = J(x,y) + N(x,y)$$

Where:

- $I(x,y)$ represents the noisy image,
- $J(x,y)$ denotes the true image without noise, and
- $N(x,y)$ is the additive Gaussian noise affecting each pixel.

Listing 2: Gaussian Noise code

```
def add_gaussian_noise(image, magnitude):
    mean = 0
    std_dev = magnitude * 255 / 100
    noisy_image = image.copy()
    cv2.randn(noisy_image, mean, std_dev)
    noisy_image = cv2.add(image, noisy_image)
    return np.clip(noisy_image, 0, 255).astype(np.uint8)
```

Here are images of the flow after adding noise.

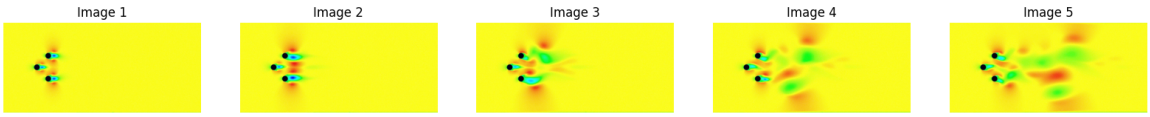

Figure 6: 20% Gaussian noised image

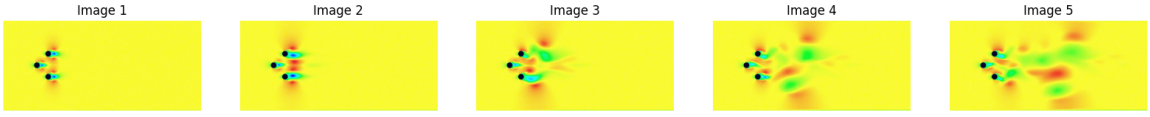

Figure 7: 40% Gaussian noised image

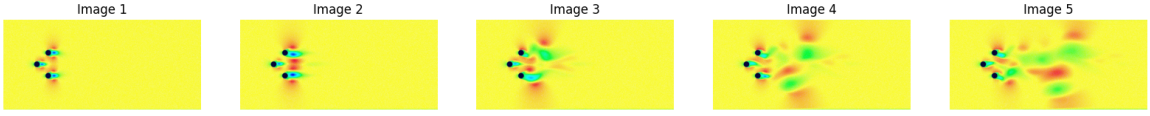

Figure 8: 60% Gaussian noised image

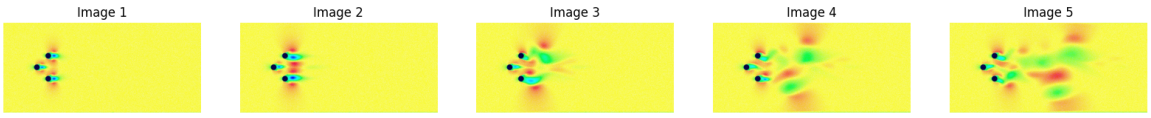

Figure 9: 80% Gaussian noised image

Adding Speckle Noise

Speckle noise, a typical occurrence in imaging, appears as a grainy or granular artifact. Unlike Gaussian noise, it is multiplicative rather than additive, causing distortion in both dark and bright areas of a picture. Speckle noise is commonly produced in radar and ultrasound imaging due to interference patterns. It complicates image processing operations like segmentation and feature extraction. Median filtering and adaptive filtering are used to reduce speckle noise while keeping key image information. Understanding and properly dealing with speckle noise is essential for improving image quality in a variety of applications.

$$I(x, y) = J(x, y) \times N(x, y)$$

Where:

- $I(x, y)$ represents the noisy image,
- $J(x, y)$ denotes the true image without noise, and
- $N(x, y)$ is the multiplicative speckle noise affecting each pixel.

Listing 3: Speckle Noise code

```
def add_speckle_noise(image, magnitude):

    # standard deviation
    std_dev = magnitude * 255 / 100

    # random noise with the same size as the input image
    noise = np.random.randn(*image.shape) * std_dev
    # adding noise to image
    noisy_image = image + image * noise

    return np.clip(noisy_image, 0, 255).astype(np.uint8) #limit the pixel value of noi
```

Here are images of the flow after adding noise.

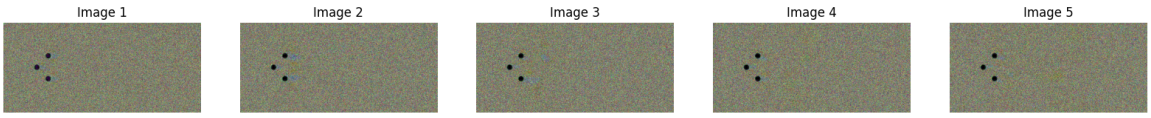

Figure 10: 20% Speckle noised image

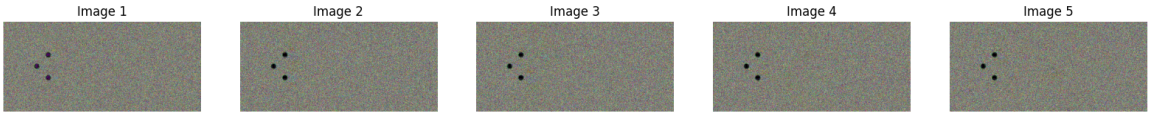

Figure 11: 40% Speckle noised image

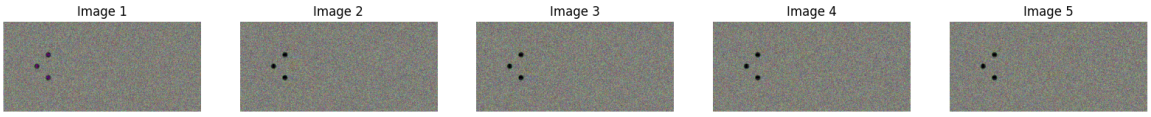

Figure 12: 60% Speckle noised image

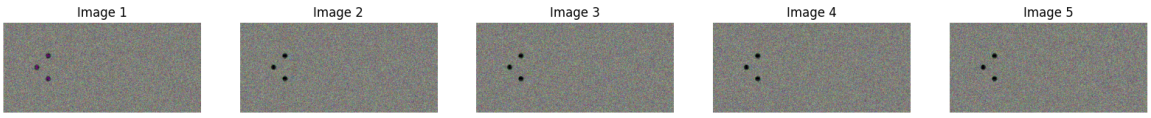

Figure 13: 80% Speckle noised image

4.2 Effect on POD Modes

POD on Gaussian noise added images

POD for 20% Gaussian noise
Doing the same as we have done in subsection 3.3.

- Mean image

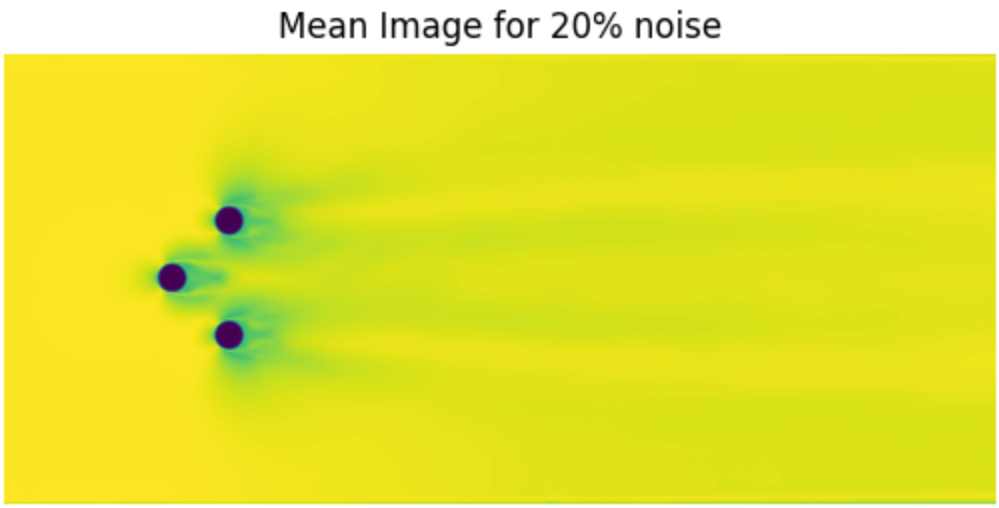

Figure 14: Mean 20% Gaussian Noise

- Cumulative Variance Ratio

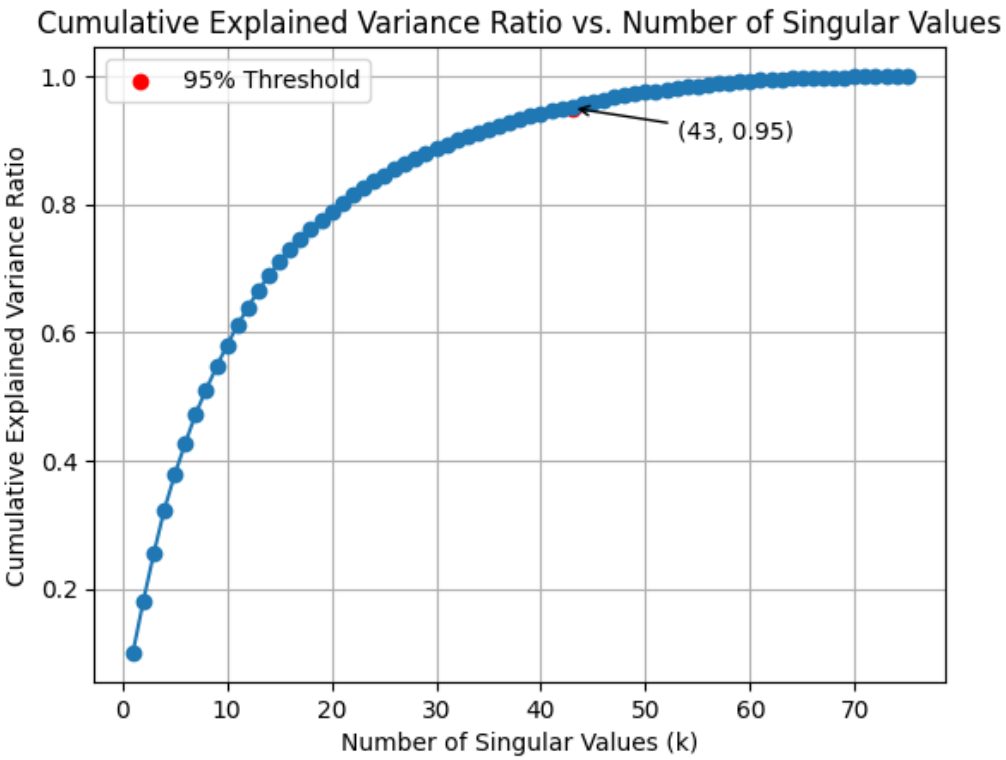

Figure 15: Cumulative Variance Ratio 20% Gaussian Noise

- Top 10 flow modes of 20% Gaussian Noise image

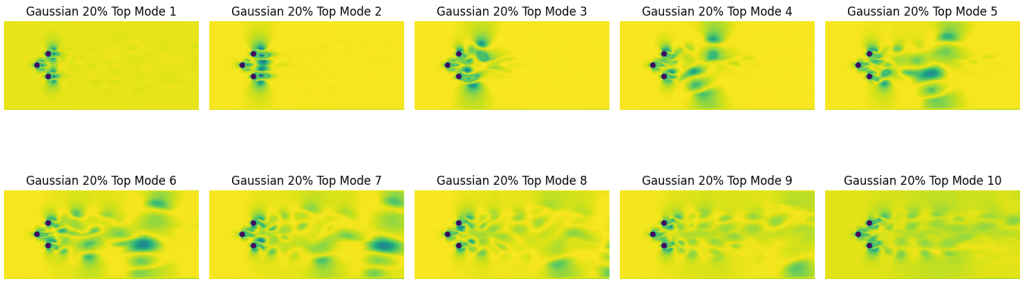

Figure 16: Top 10 modes 20% Gaussian Noise

- Total Energy by each 10 modes for Gaussian 20% noise

  These 10 modes posses about 60.88% energy of total energy.[Refer code file]

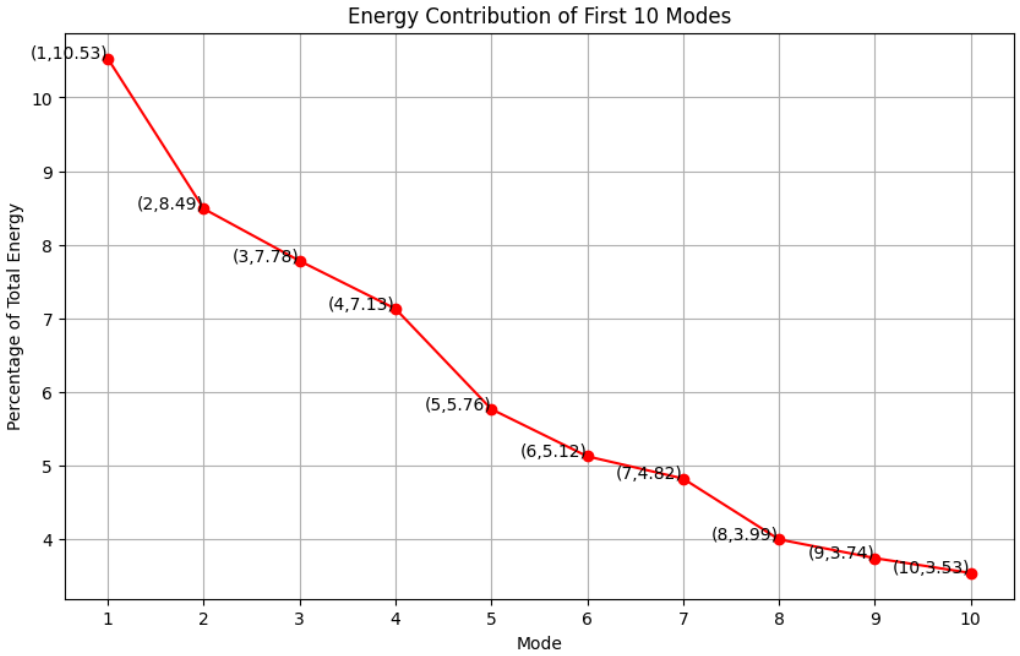

Figure 17: Energy of each 10 modes for 20% Gaussian Noise

POD for 40% Gaussian noise

Doing the same as we have done in subsection 3.3.

- Mean image

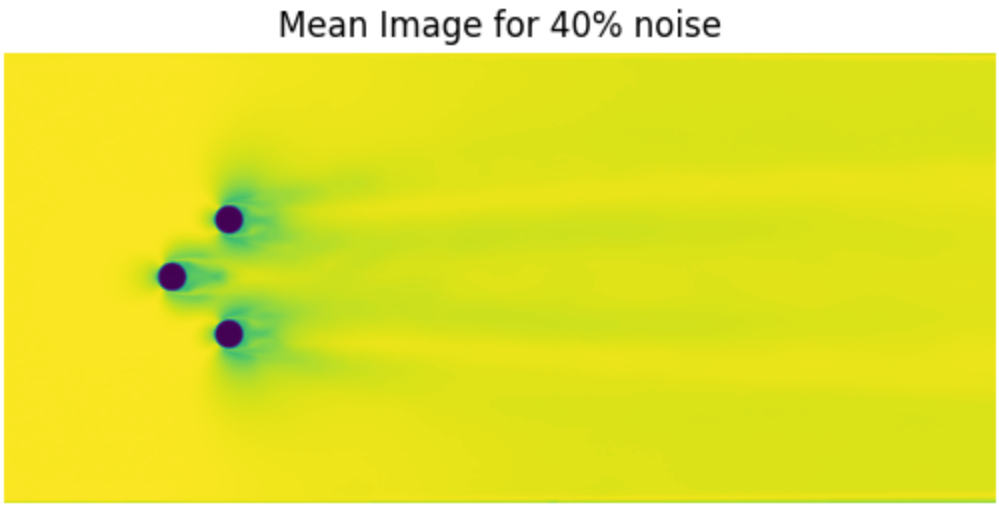

Figure 18: Mean 40% Gaussian Noise

- Cumulative Variance Ratio

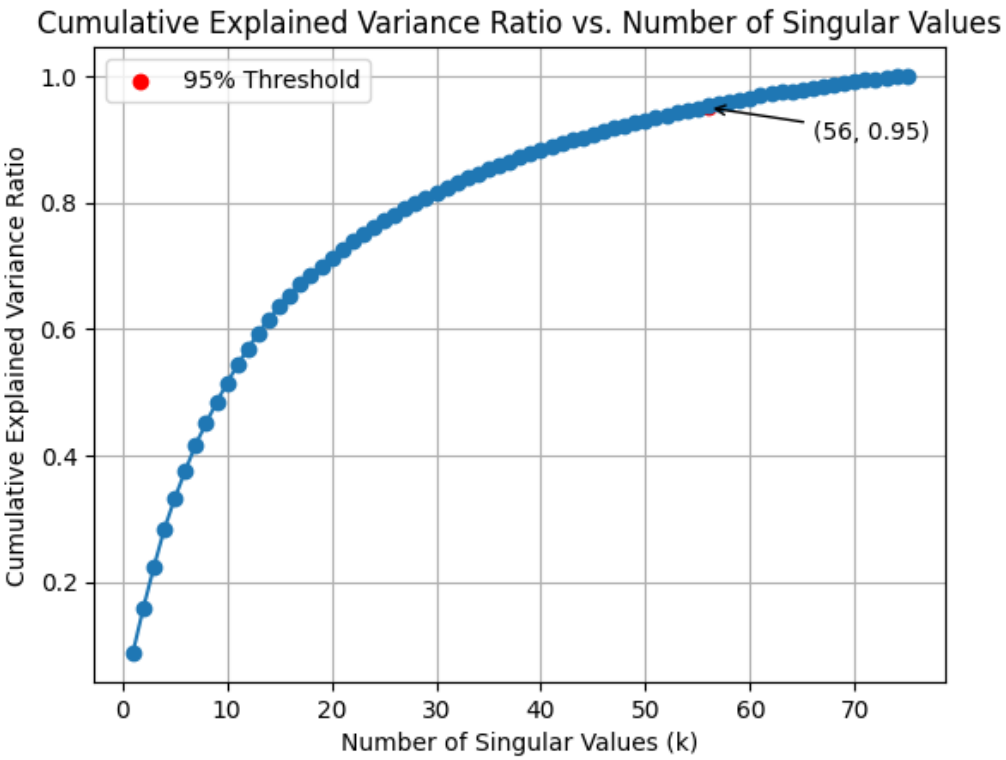

Figure 19: Cumulative Variance Ratio 40% Gaussian Noise

- Top 10 flow modes of 40% Gaussian Noise image

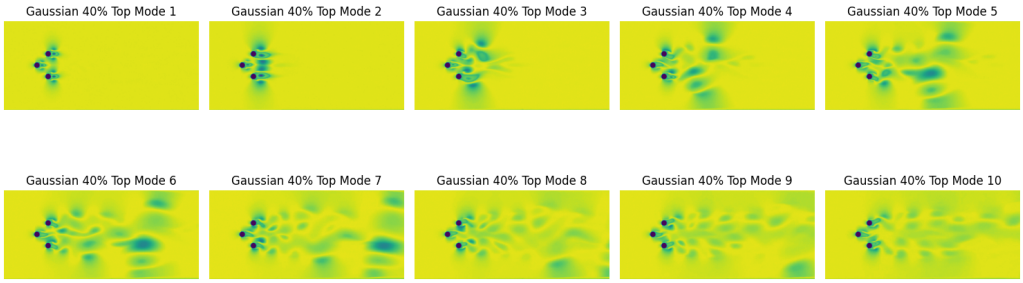

Figure 20: Top 10 modes 40% Gaussian Noise

- Total Energy by every 10 modes for Gaussian 40% noise

  These 10 modes posses about 53.98% energy of total energy.[Refer code file]

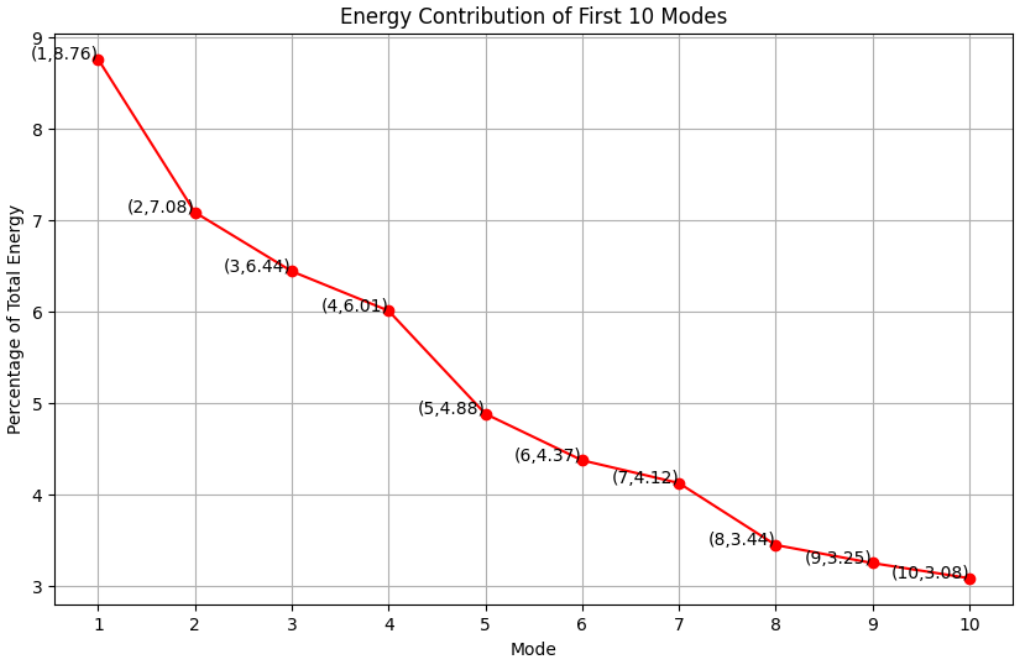

Figure 21: Energy of each 10 modes for 40% Gaussian Noise

POD for 60% Gaussian noise

Doing the same as we have done in subsection 3.3.

- Mean image

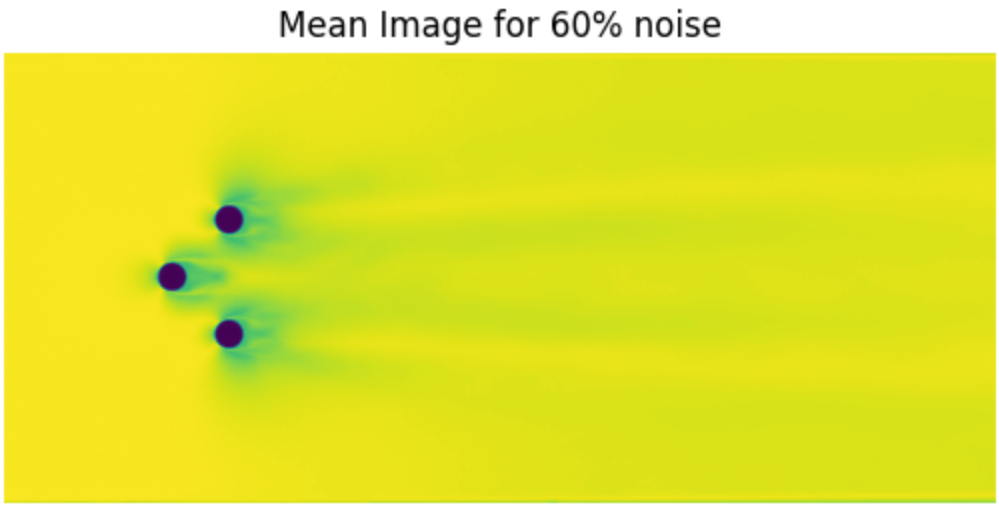

Figure 22: Mean 60% Gaussian Noise

- Cumulative Variance Ratio

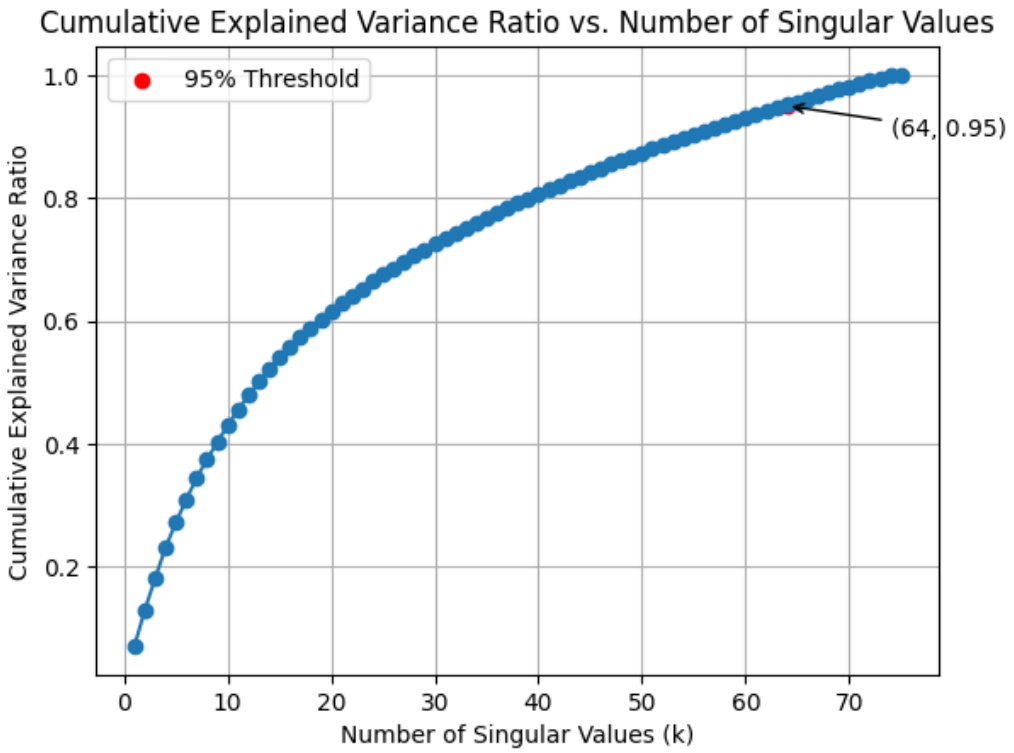

Figure 23: Cumulative Variance Ratio 60% Gaussian Noise

- Top 10 flow modes of 20% Gaussian Noise image

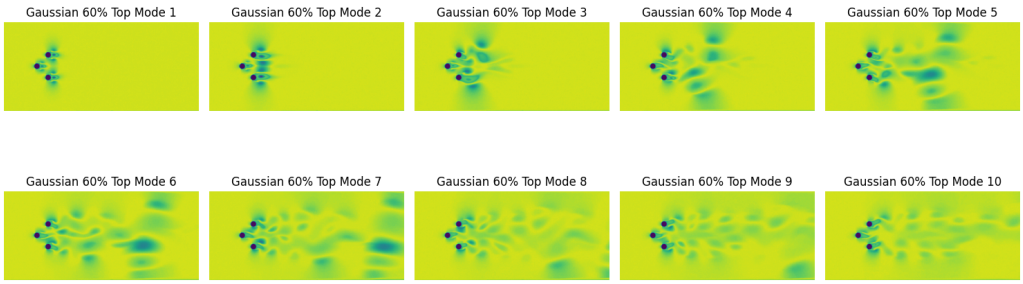

Figure 24: Top 10 modes 60% Gaussian Noise

- Total Energy by each 10 modes for Gaussian 60% noise

  These 10 modes posses about 45.19% energy of total energy.[Refer code file]

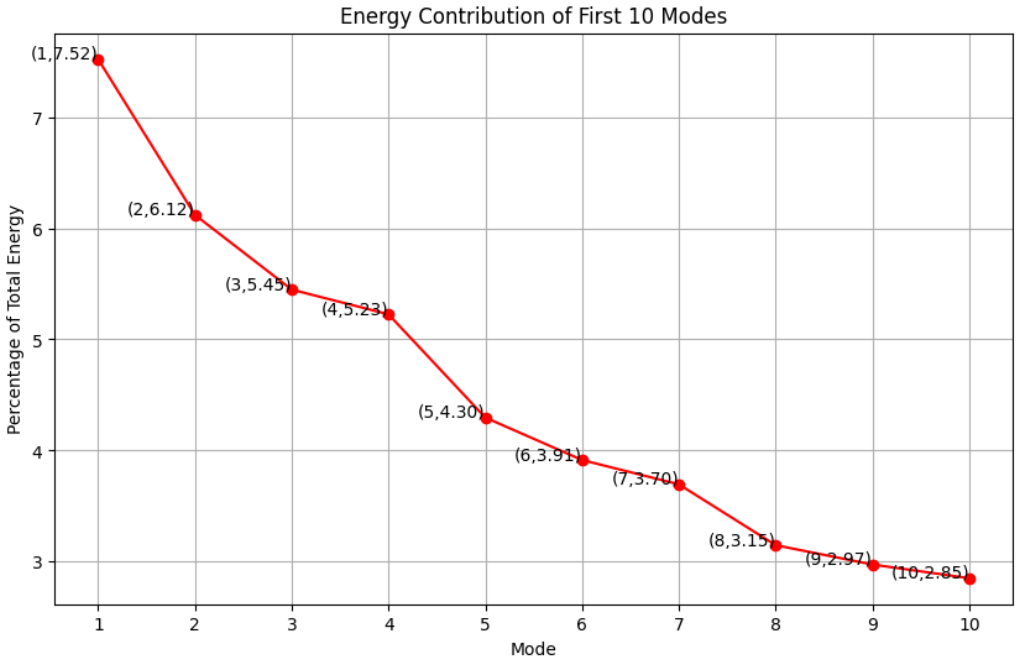

Figure 25: Energy of each 10 modes for 60% Gaussian Noise

POD for 80% Gaussian noise

Doing the same as we have done in subsection 3.3.

- Mean image

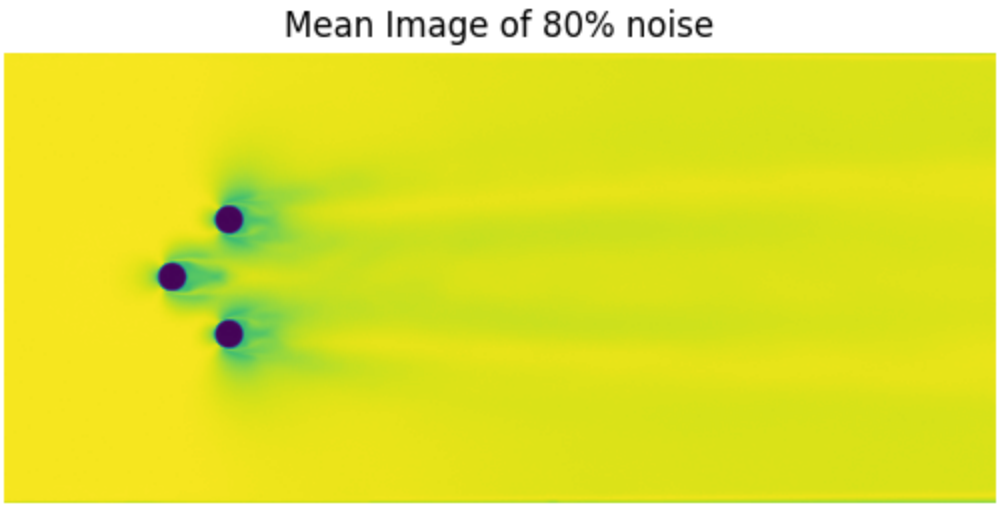

Figure 26: Mean 80% Gaussian Noise

- Cumulative Variance Ratio

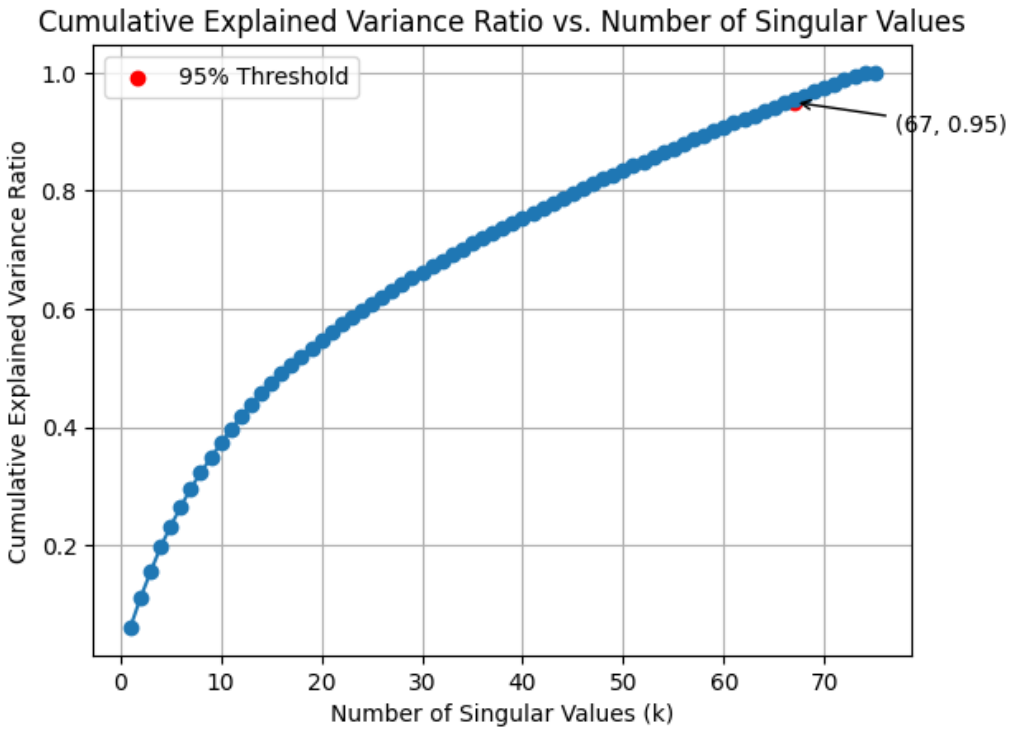

Figure 27: Cumulative Variance Ratio 80% Gaussian Noise

- Top 10 flow modes of 80% Gaussian Noise image

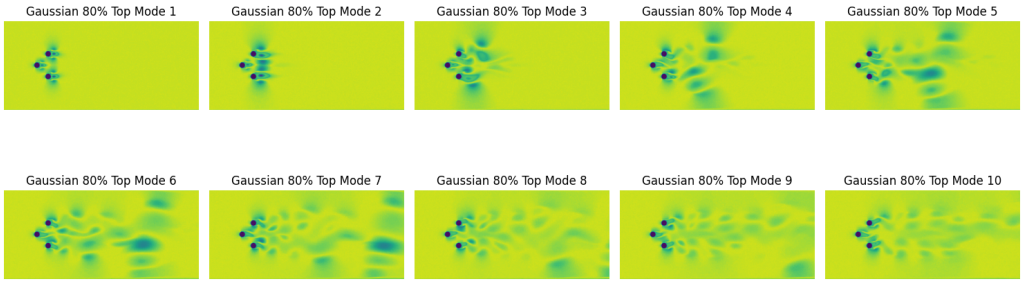

Figure 28: Top 10 modes 80% Gaussian Noise

- Total Energy by each 10 modes for Gaussian 80% noise

  These 10 modes posses about 38.97% energy of total energy.[Refer code file]

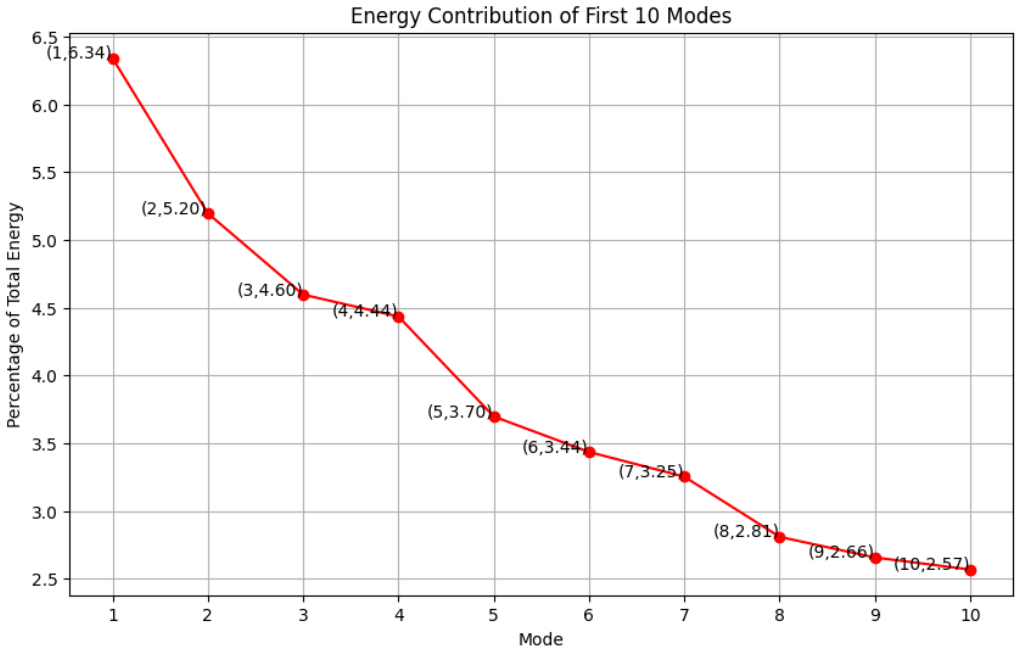

Figure 29: Energy of each 10 modes for 80% Gaussian Noise

% Noise	Cumulative variance ration	Total energy by 10 modes	Energy of first mode
20	43	60.88%	10.53%
40	56	53.98%	8.76%
60	64	45.19%	7.52%
80	67	38.97%	6.34%

Table 2: POD table for Gaussian noisy images

From Table 2 it can be observed that adding noise of different magnitudes significantly affects the energy of modes. As the noise level is increasing the energy is decreasing. Refer:subsection 4.1. It is important to note here as noise is increasing the singular values (k) increase showing that the number of features will increase if we increase the noise. Leading high computations cost.

- How does the energy transcend through modes?

  For different noise levels, we can see that on increasing noise the energy of modes goes on decreasing, having the least noise has the highest energy of each mode and vice versa.
  Note: Refer code file.

- Can you give any statistical evidence of the energy changes in the modes?

  Refer to these images. Figure 17 Figure 21 Figure 25 Figure 29

- Do different noise types have different types of response in terms of how the energy of the modes changes?

  Yes, different noise levels have a different response because in this case noise is added to the pixel value. It is clearly observable that increasing noise, and energy are decreasing. Refer: Table 2

- How does the magnitude of noise affect the modal energy?

$$Noise \propto \frac{1}{Energy}$$

- What sort of noise is best suited to be used for POD for artificial analysis?

  Addative types of noise are best for fluid flow analysis. e.g. Gaussian Noise

POD on Speckle noise added images

POD for 20% Speckle noise

Doing the same as we have done in subsection 3.3.

- Mean image

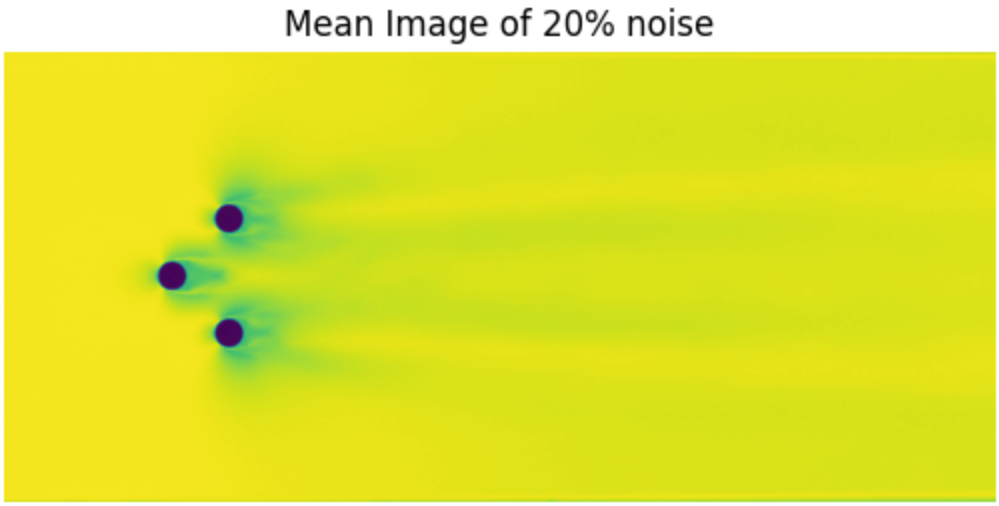

Figure 30: Mean 20% Speckle Noise

- Cumulative Variance Ratio

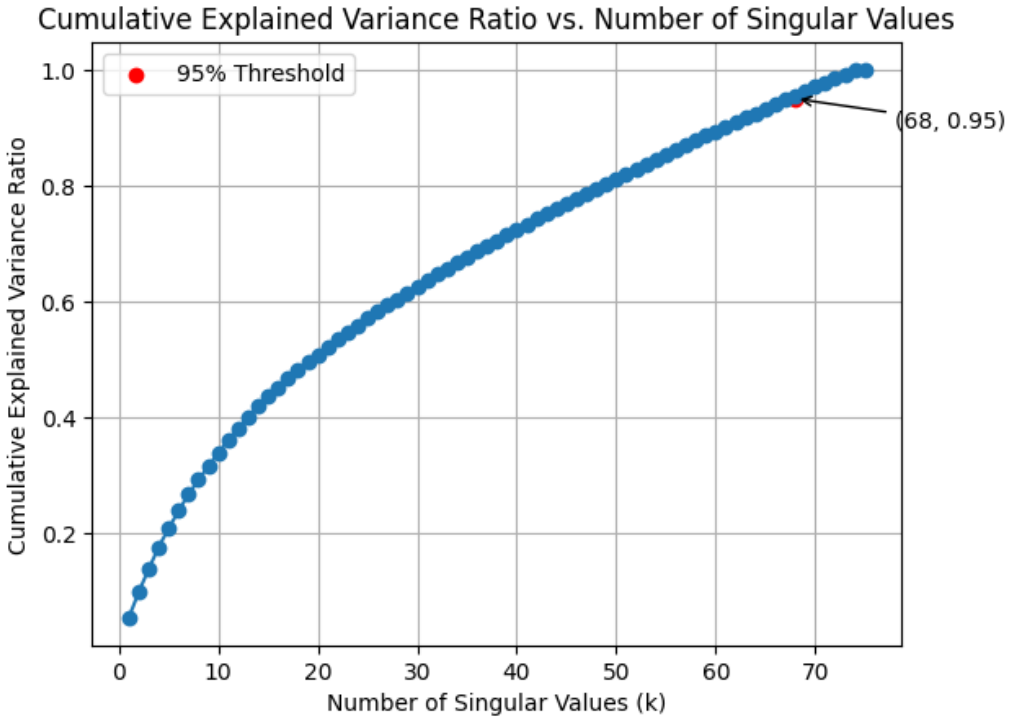

Figure 31: Cumulative Variance Ratio 20% Speckle Noise

- Top 10 flow modes of 20% Speckle Noise image

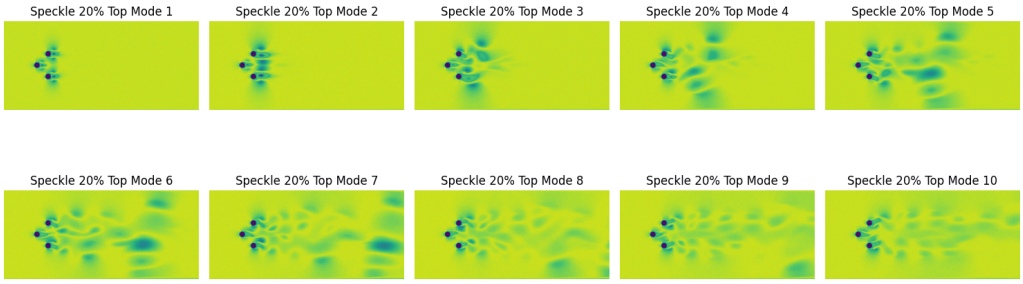

Figure 32: Top 10 modes 20% Speckle Noise

- Total Energy by each 10 modes for Speckle 20% noise

  These 10 modes posses about 35.44% energy of total energy.[Refer code file]

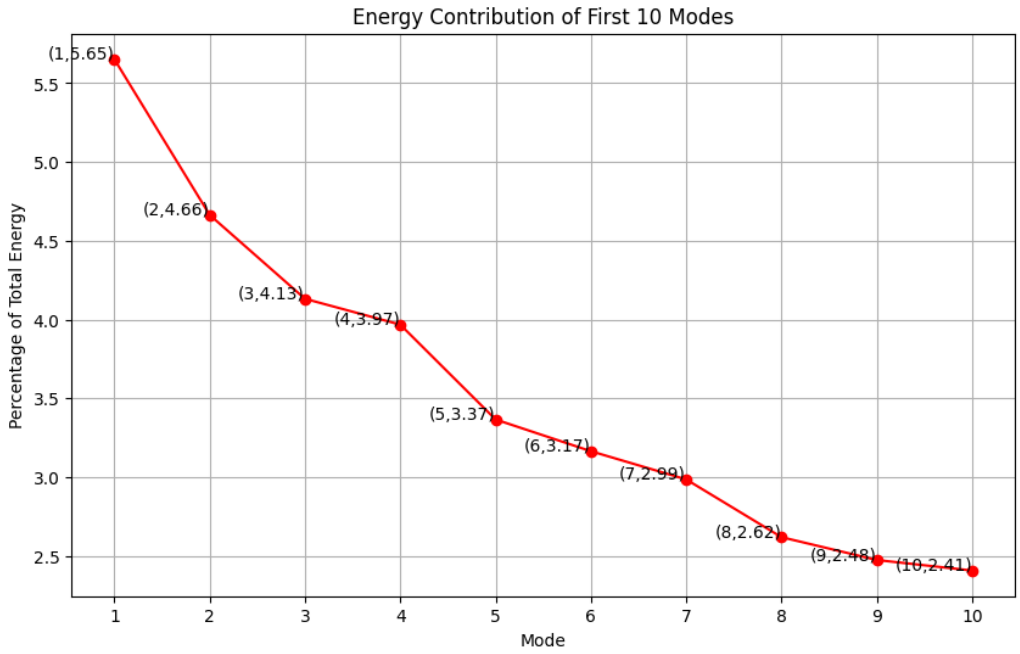

Figure 33: Energy of each 10 modes for 20% Speckle Noise

POD for 40% Speckle noise

Doing the same as we have done in subsection 3.3.

- Mean image

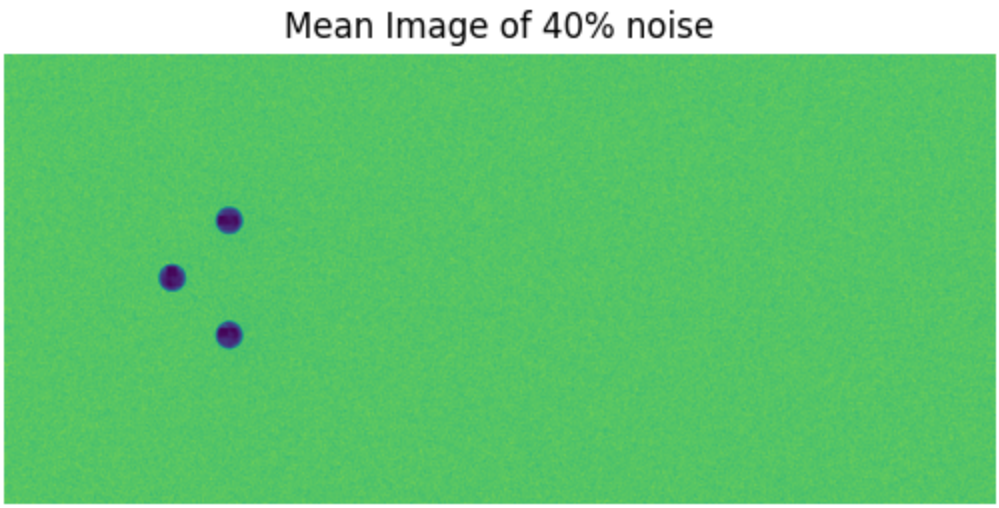

Figure 34: Mean 40% Speckle Noise

- Cumulative Variance Ratio

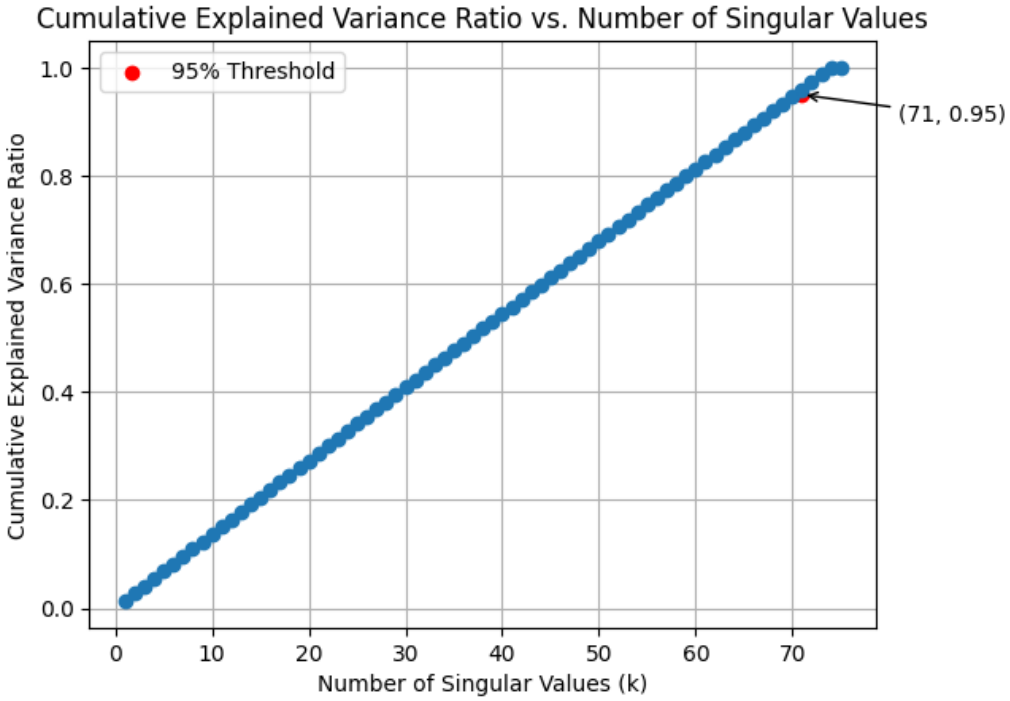

Figure 35: Cumulative Variance Ratio 40% Speckle Noise

- Top 10 flow modes of 40% Speckle Noise image

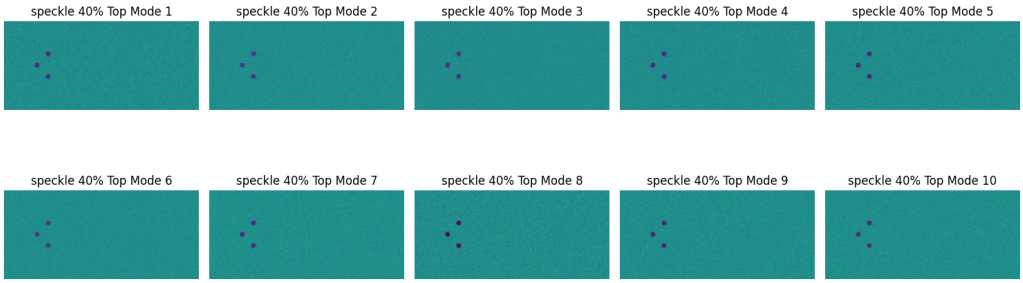

Figure 36: Top 10 modes 40% Speckle Noise

- Total Energy by each 10 modes for Speckle 40% noise

  These 10 modes posses about 14.25% energy of total energy.[Refer code file]

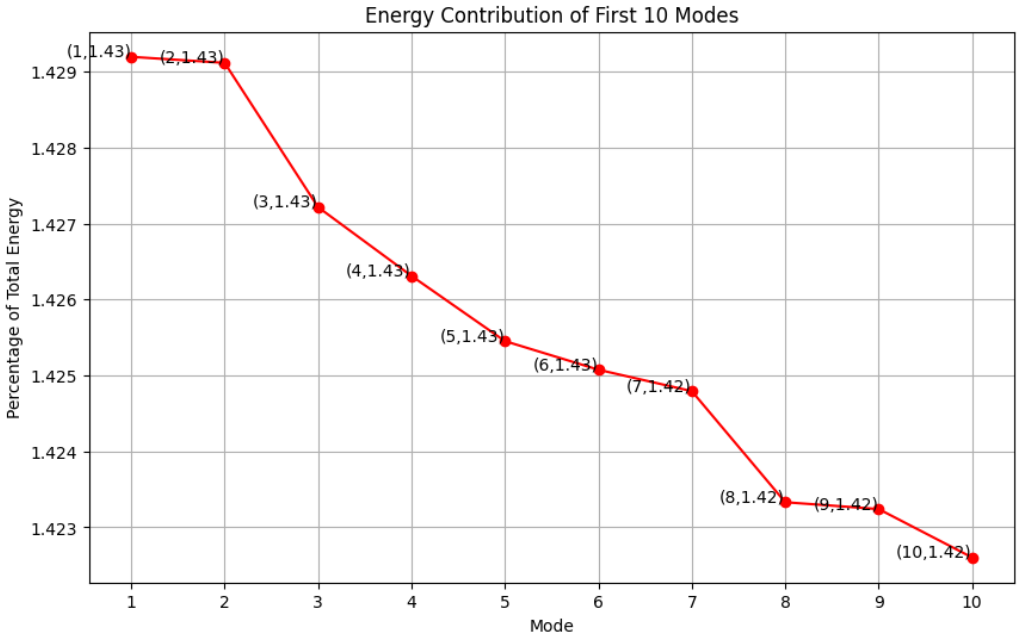

Figure 37: Energy of each 10 modes for 40% Speckle Noise

POD for 60% Speckle noise

Doing the same as we have done in subsection 3.3.

- Mean image

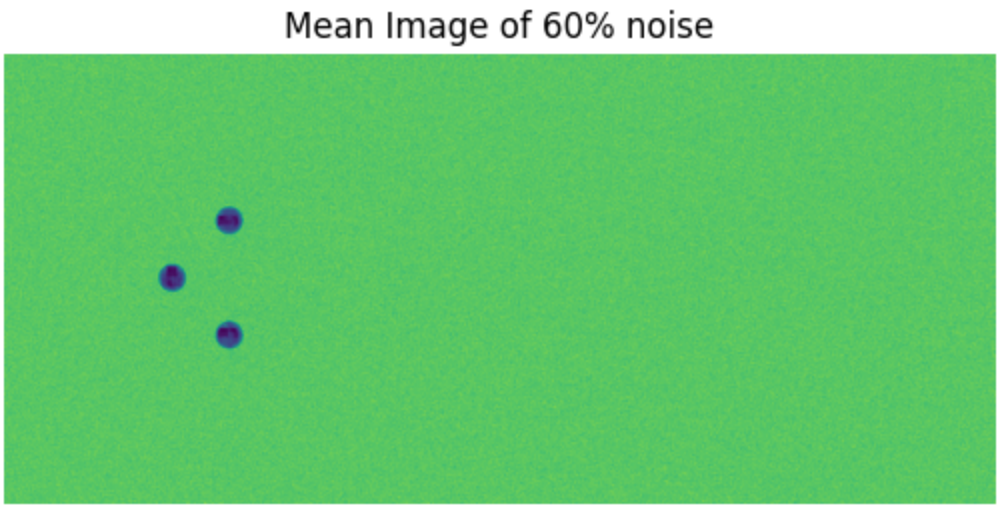

Figure 38: Mean 60% Speckle Noise

- Cumulative Variance Ratio

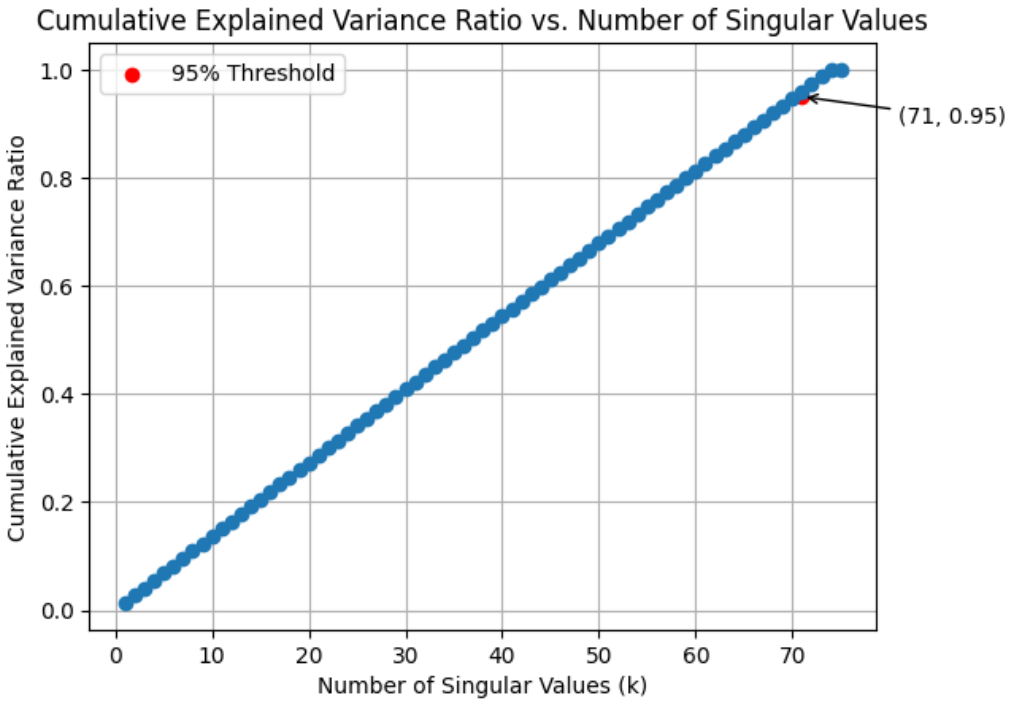

Figure 39: Cumulative Variance Ratio 60% Speckle Noise

- Top 10 flow modes of 60% Speckle Noise image

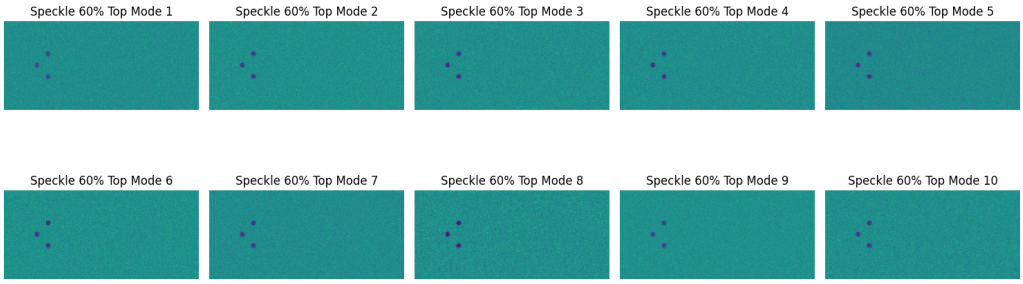

Figure 40: Top 10 modes 60% Speckle Noise

- Total Energy by each 10 modes for Speckle 60% noise

  These 10 modes posses about 14.25% energy of total energy.[Refer code file]

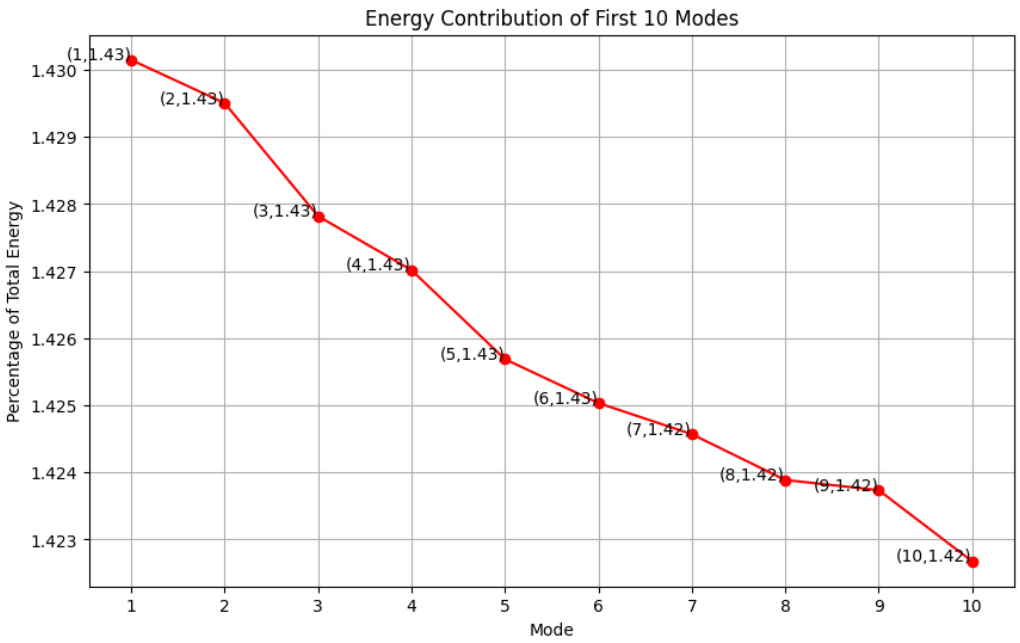

Figure 41: Energy of each 10 modes for 60% Speckle Noise

POD for 80% Speckle noise

Doing the same as we have done in subsection 3.3.

- Mean image

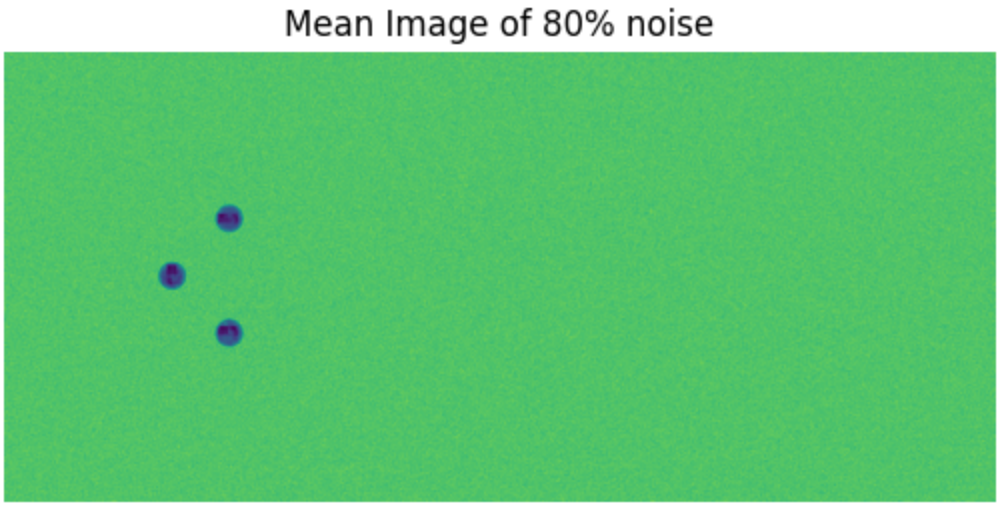

Figure 42: Mean 80% Speckle Noise

- Cumulative Variance Ratio

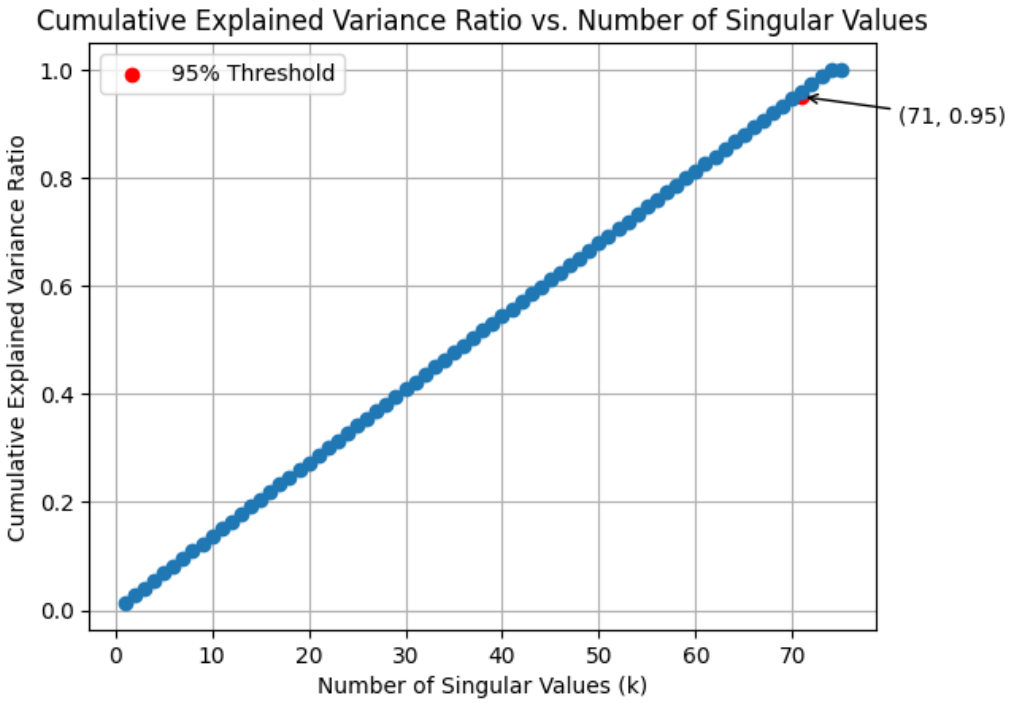

Figure 43: Cumulative Variance Ratio 80% Speckle Noise

- Top 10 flow modes of 80% Speckle Noise image

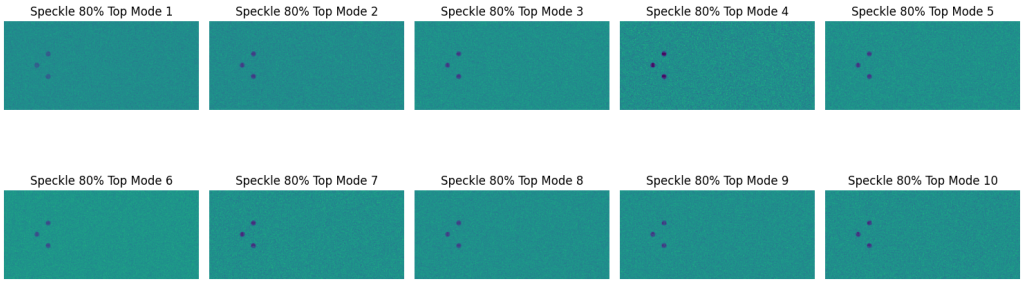

Figure 44: Top 10 modes 80% Speckle Noise

- Total Energy by each 10 modes for Speckle 80% noise

  These 10 modes posses about 14.25% energy of total energy.[Refer code file]

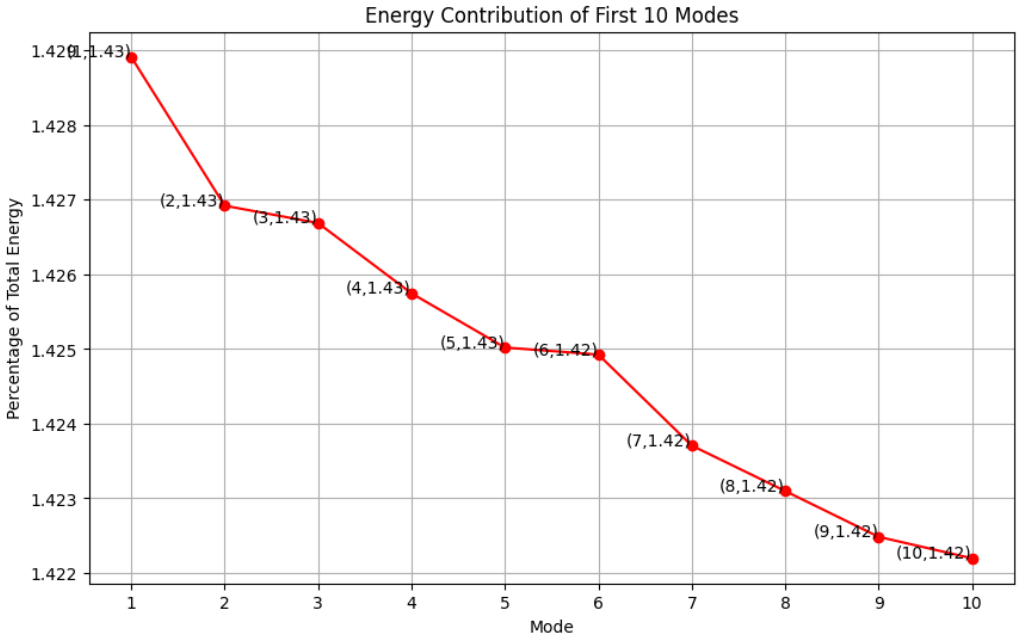

Figure 45: Energy of each 10 modes for 80% Speckle Noise

% Noise	Cumulative variance ration	Total energy by 10 modes	Energy of first mode
20	68	35.44%	5.64%
40	71	14.25%	1.43%
60	71	14.26%	1.43%
80	71	14.26%	1.43%

Table 3: POD table for Speckle noisy images

From Table 3 it can be observed that after adding noise of more than 20%, there is no variation in the value of singular values, energy, and energy of the first mode. This is because speckle noise is a multiplicative noise, It multiplies by the pixel value and wherever it is finding the very low value of pixels it becomes approximately zero on multiplying. Refer:subsection 4.1. As the noise level is increasing the number of noisy pixels is increasing and most of them are tending to zero, so we are not getting any variation on adding the noises at different levels.

- How does the energy transcend through modes?

  For different noise levels, we can see that on increasing noise the energy of modes goes on decreasing, having the least noise has the highest energy of each mode and vice versa.
  Note: Refer code file.

- Can you give any statistical evidence of the energy changes in the modes?

  Refer to these images. Figure 33 Figure 37 Figure 41 Figure 45

- Do different noise types have different types of response in terms of how the energy of the modes changes?

  Yes, different noise levels have a different response because in this case noise is multiplied by pixel value, and in most of the cases it is approx zero. So we are not observing any significant change in energies. But where it will be additive noise e.g. Gaussian then it will affect more. Refer: Table 3

- How does the magnitude of noise affect the modal energy?

$$Noise \propto \frac{1}{Energy}$$

- What sort of noise is best suited to be used for POD for artificial analysis?

  Addative types of noise are best for fluid flow analysis. e.g. Gaussian Noise

5 Super-Resolving

5.1 Denoised using Gaussian Blur

POD for 20% Gaussian noise images, removed using Gaussian blur
Doing the same as we have done in subsection 3.3.

- Mean image

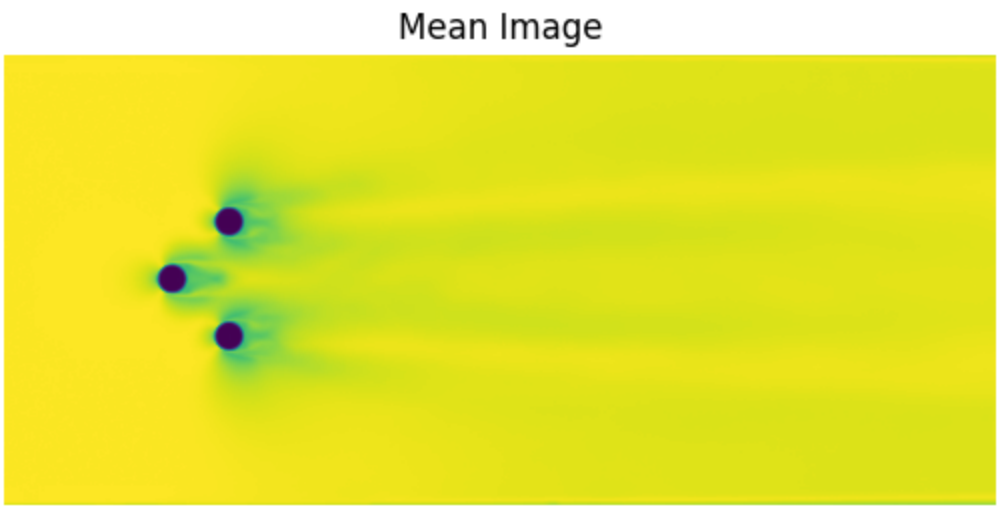

Figure 46: Mean 20% Gaussian Blur

- Cumulative Variance Ratio

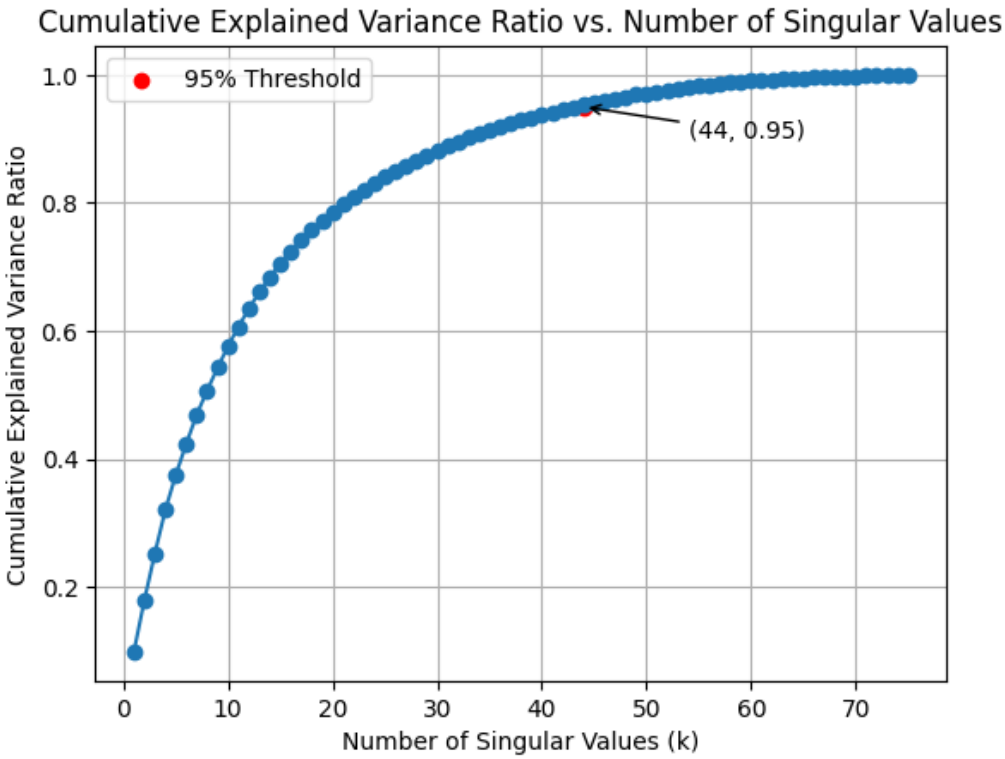

Figure 47: Cumulative Variance Ratio for 20% noisy images, removed images using Gaussian blur

- Top 10 flow modes of 20% noisy images, removed images using Gaussian blur.

Figure 48: Top 10 modes 20% noise removed images using Gaussian blur

- Total Energy by every 10 modes for 20% noisy images, removed images using Gaussian blur.

  These 10 modes posses about 60.45% energy of total energy.[Refer code file]

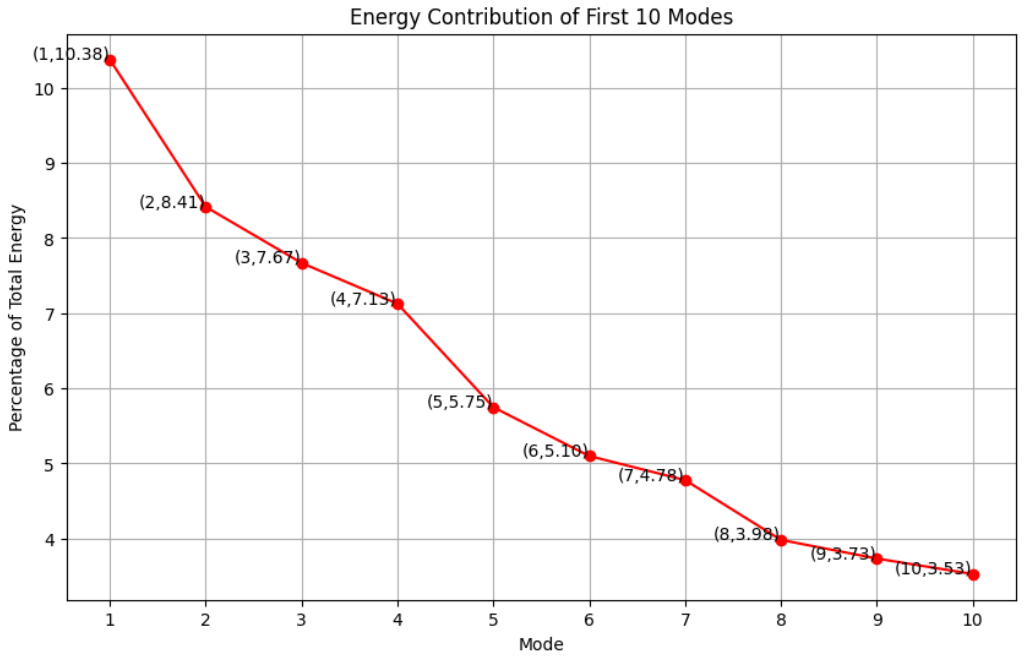

Figure 49: Energy of each 10 modes for 20% noisy images, removed using gaussian blur

POD for 40% Gaussian noise images using Gaussian blur
Doing the same as we have done in subsection 3.3.

- Mean image

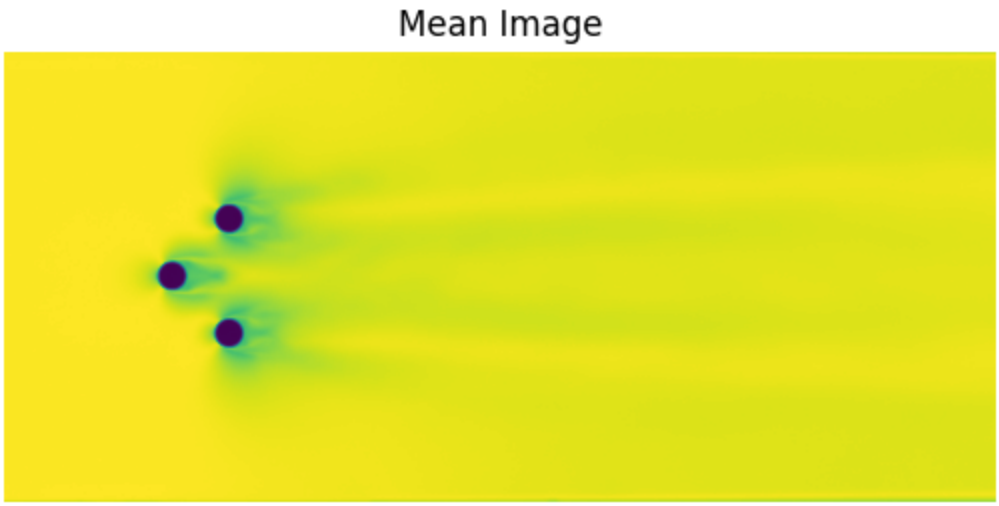

Figure 50: Mean 40% Gaussian Blur

- Cumulative Variance Ratio

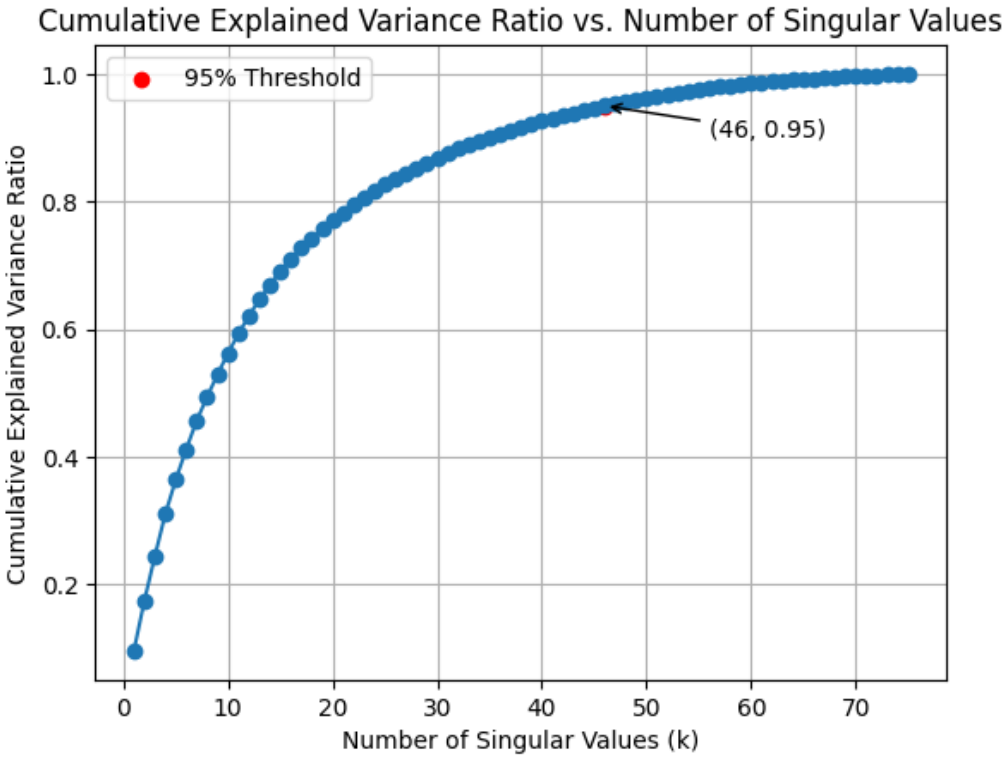

Figure 51: Cumulative Variance Ratio for 40% noisy images, removed images using Gaussian blur

- Top 10 flow modes of 40% noisy images, removed images using Gaussian blur.

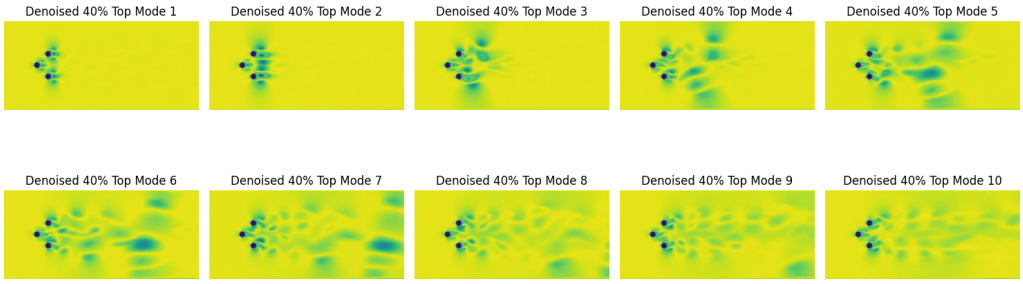

Figure 52: Top 10 modes 40% noise removed images using Gaussian blur

- Total Energy by every 10 modes for 40% noisy images, removed images using Gaussian blur.

  These 10 modes posses about 59.19% energy of total energy.[Refer code file]

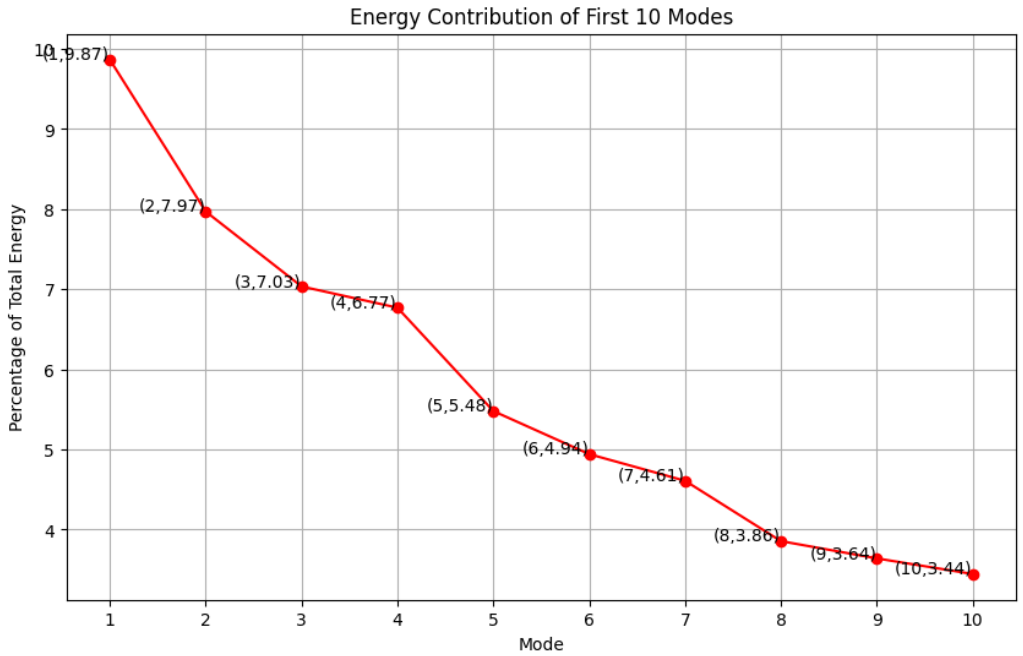

Figure 53: Energy of each 10 modes for 40% noisy images, removed using gaussian blur

POD for 60% Gaussian noise images using Gaussian blur
Doing the same as we have done in subsection 3.3.

- Mean image

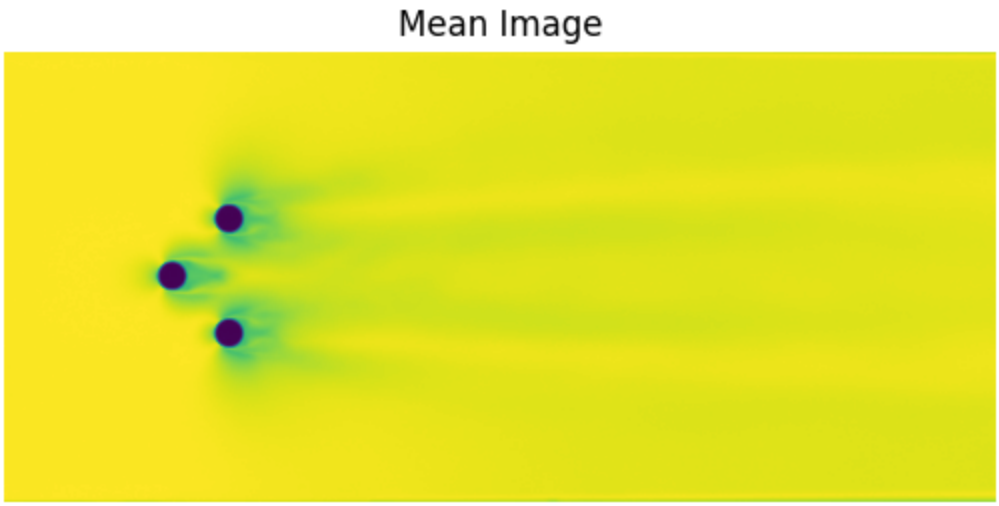

Figure 54: Mean 60% Gaussian Blur

- Cumulative Variance Ratio

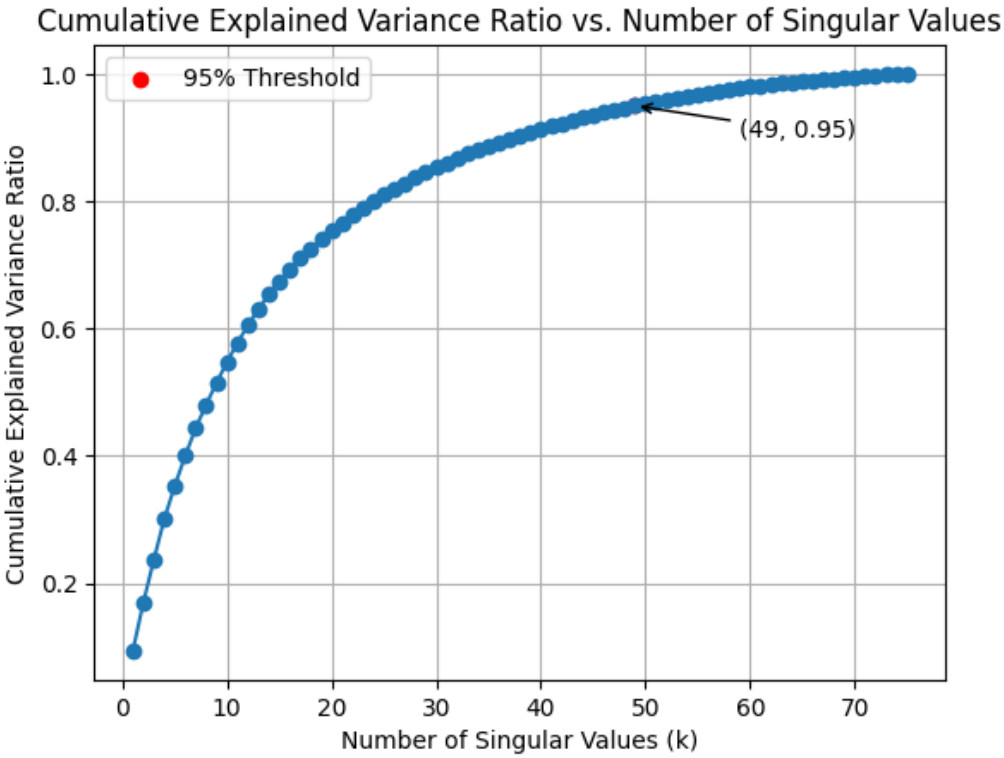

Figure 55: Cumulative Variance Ratio for 60% noisy images, removed images using Gaussian blur

- Top 10 flow modes of 60% noisy images, removed images using Gaussian blur.

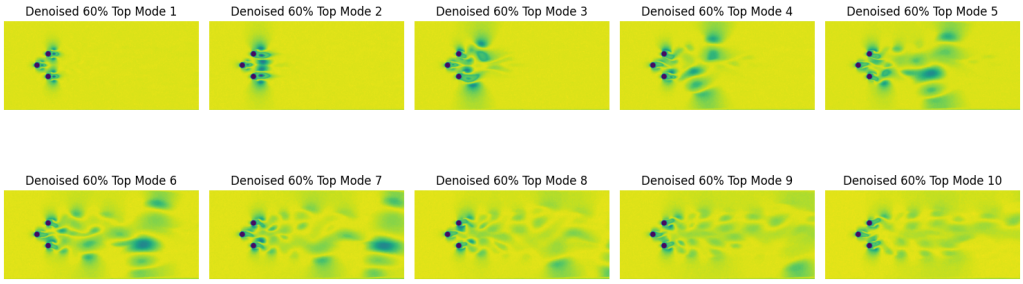

Figure 56: Top 10 modes 60% noise removed images using Gaussian blur

- Total Energy by every 10 modes for 60% noisy images, removed images using Gaussian blur.

  These 10 modes posses about 57.61% energy of total energy.[Refer code file]

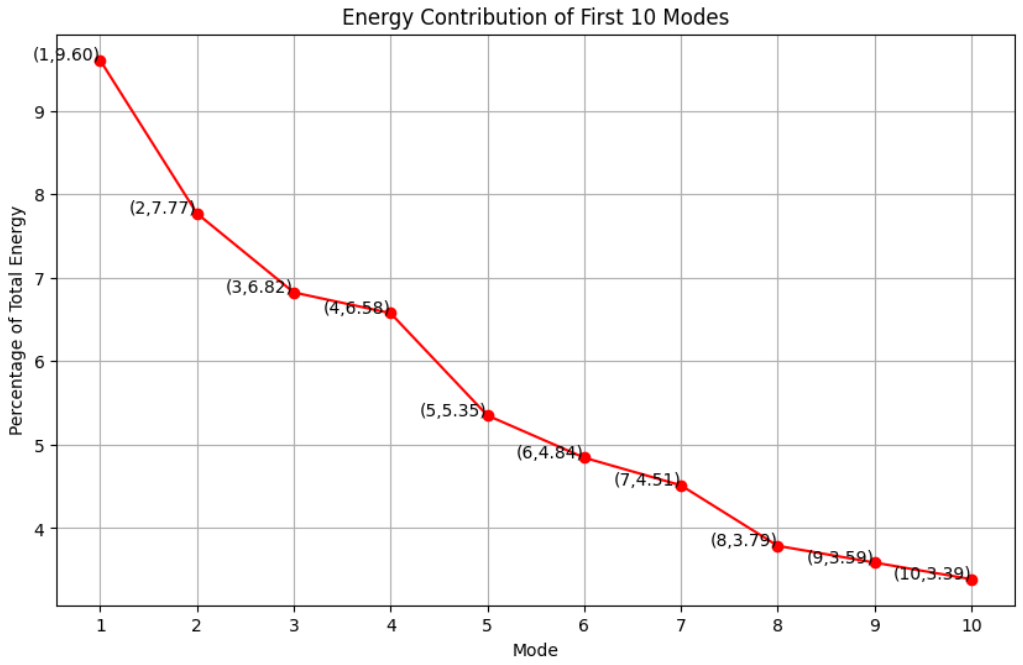

Figure 57: Energy of each 10 modes for 60% noisy images, removed using gaussian blur

POD for 80% Gaussian noise images using Gaussian blur
Doing the same as we have done in subsection 3.3.

- Mean image

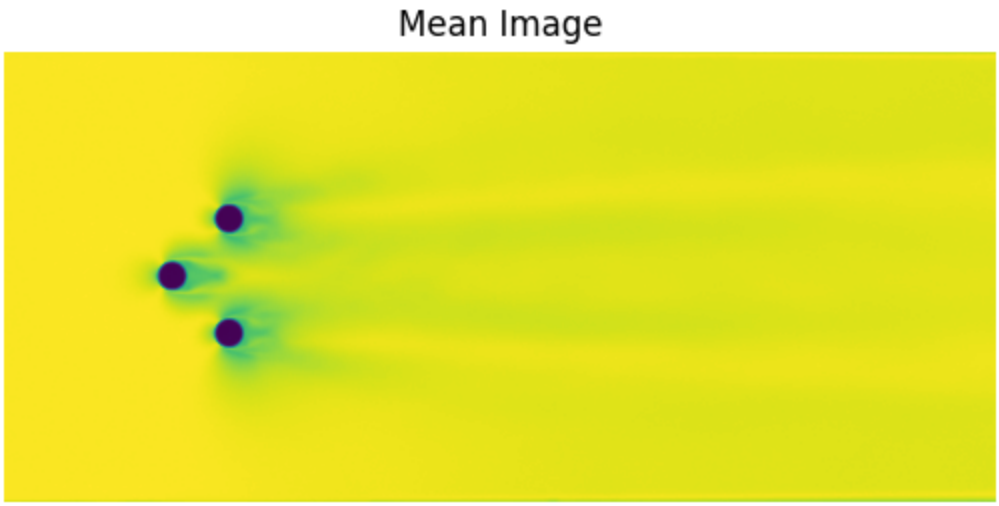

Figure 58: Mean 20% Gaussian Blur

- Cumulative Variance Ratio

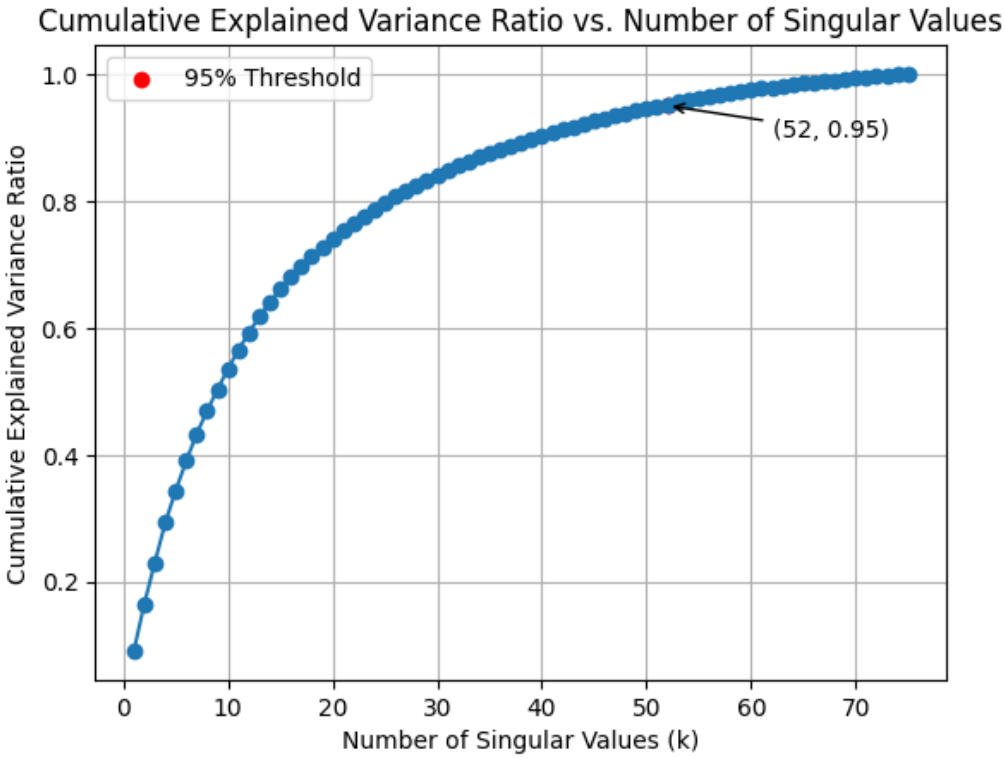

Figure 59: Cumulative Variance Ratio for 80% noisy images, removed images using Gaussian blur

- Top 10 flow modes of 80% noisy images, removed images using Gaussian blur.

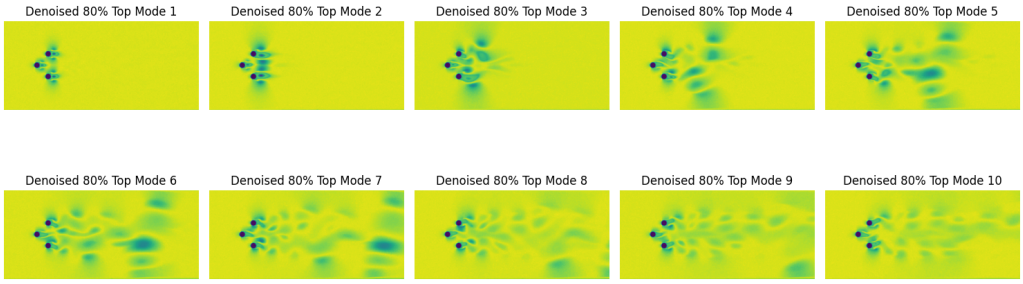

Figure 60: Top 10 modes 80% noise removed images using Gaussian blur

- Total Energy by every 10 modes for 80% noisy images, removed images using Gaussian blur.

  These 10 modes posses about 56.25% energy of total energy.[Refer code file]

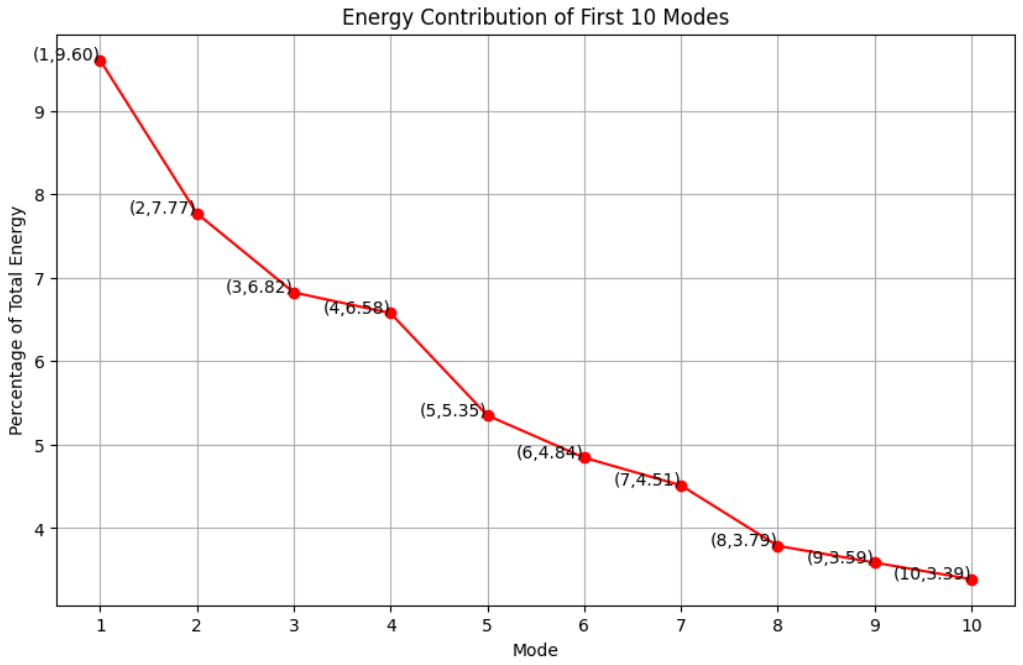

Figure 61: Energy of each 10 modes for 80% noisy images, removed using gaussian blur

RealImageEnergy = 60.88%	20%	40%	60%	80%
Noisy Image energy %	60.88%	53.98%	45.19%	38.97
Denoised image energy %	60.45%	59.19%	57.61%	56.25

Table 4: Comparison of energies of noisy images and Denoised images

As we can see energy contribution values [Refer: Table 4] are nearly about real energy value i.e. 60.88%. But for higher noise values it is lagging a little bit because It applies a convolution operation with a Gaussian kernel to the image, which effectively averages the pixel values in the neighbourhood of each pixel. During operation averaging around high pixel values makes it smaller leading to potential loss in image detail.

5.2 Denoised using NLM

POD for 20% Gaussian noise removed images using NLM
Doing the same as we have done in subsection 3.3.

- Mean image

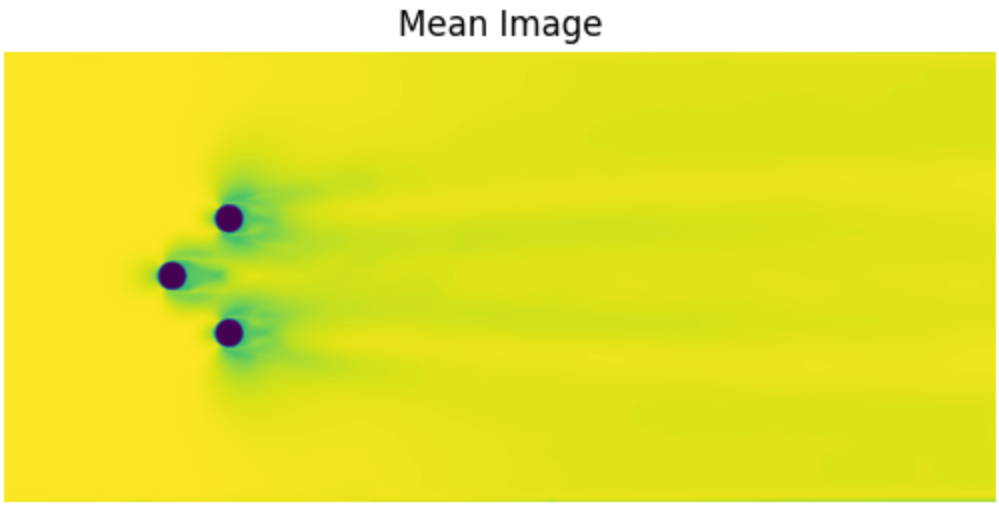

Figure 62: Mean 20% NLM

- Cumulative Variance Ratio

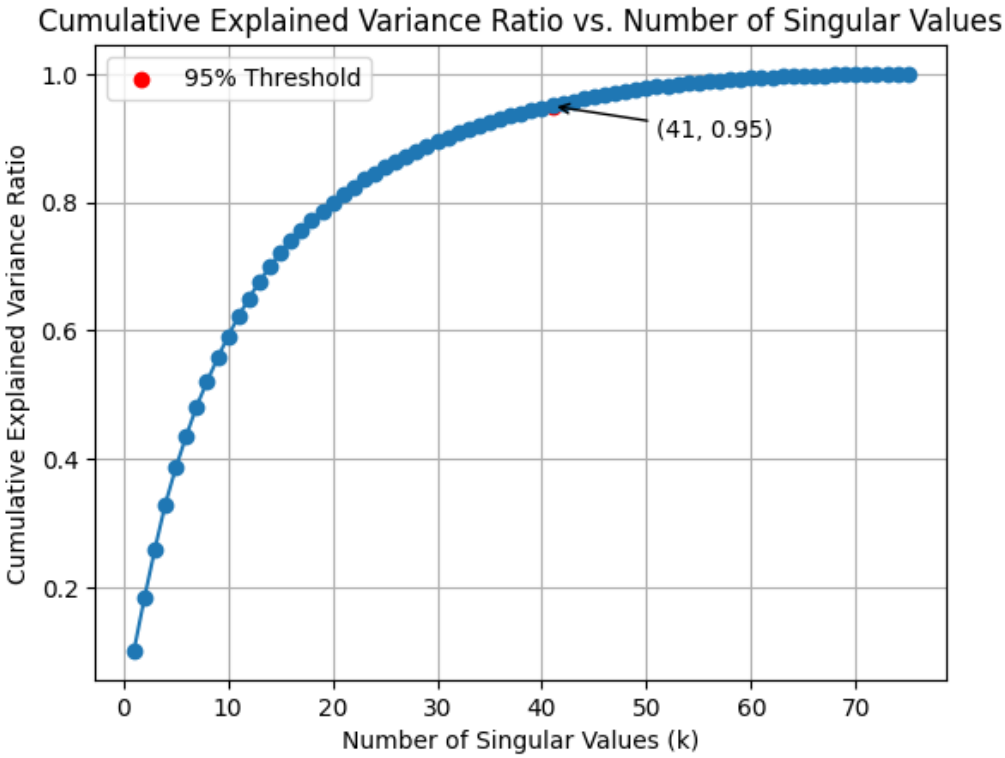

Figure 63: Cumulative Variance Ratio for 20% noisy images, removed images using NLM

- Top 10 flow modes of 20% noisy images, removed images using NLM.

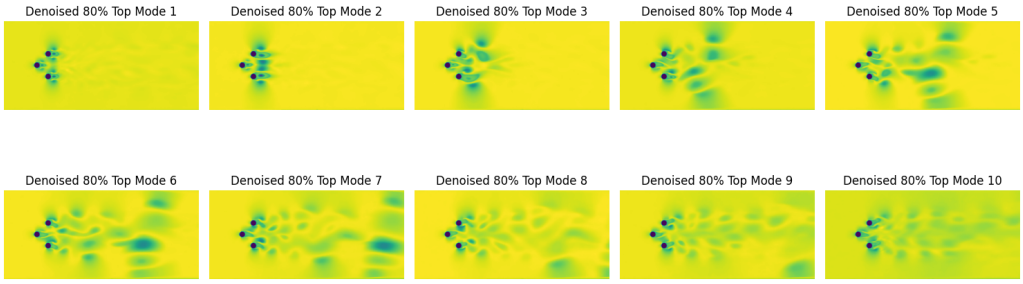

Figure 64: Top 10 modes 20% noise removed images using NLM

- Total Energy by every 10 modes for 20% noisy images, removed images using NLM.

  These 10 modes posses about 62.15% energy of total energy.[Refer code file]

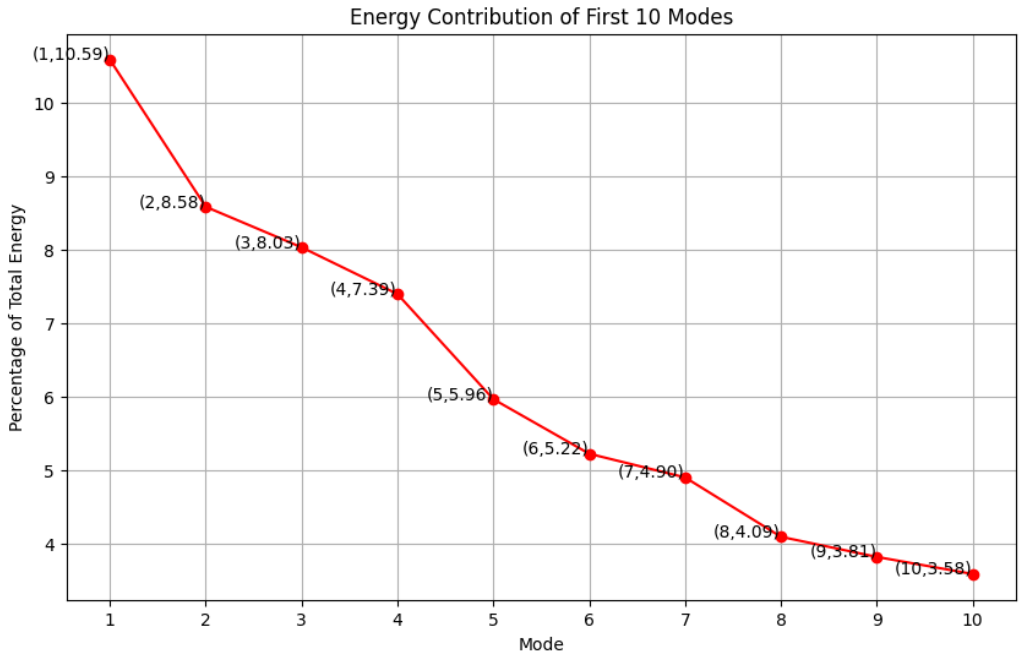

Figure 65: Energy of each 10 modes for 20% noisy images, removed using NLM

POD for 40% Gaussian noise images using NLM
Doing the same as we have done in subsection 3.3.

- Mean image

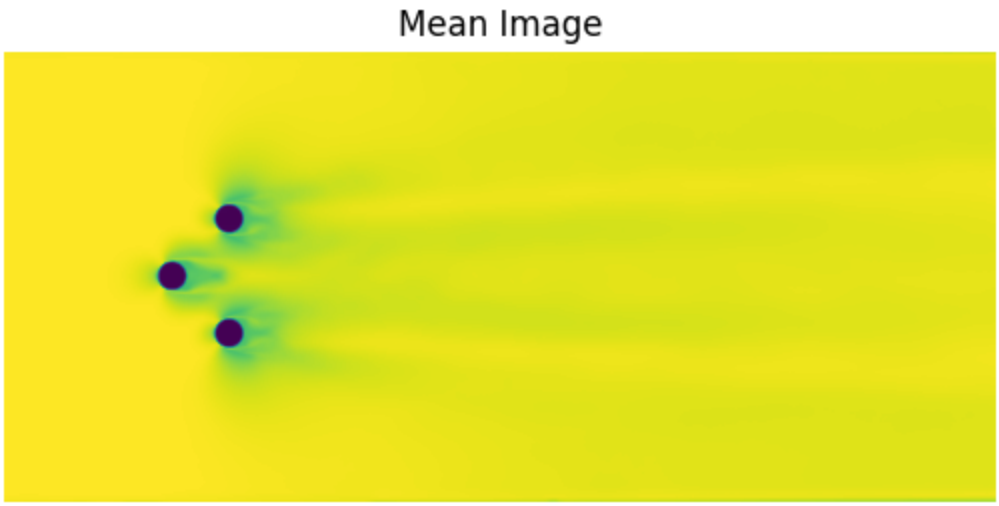

Figure 66: Mean 40% NLM

- Cumulative Variance Ratio

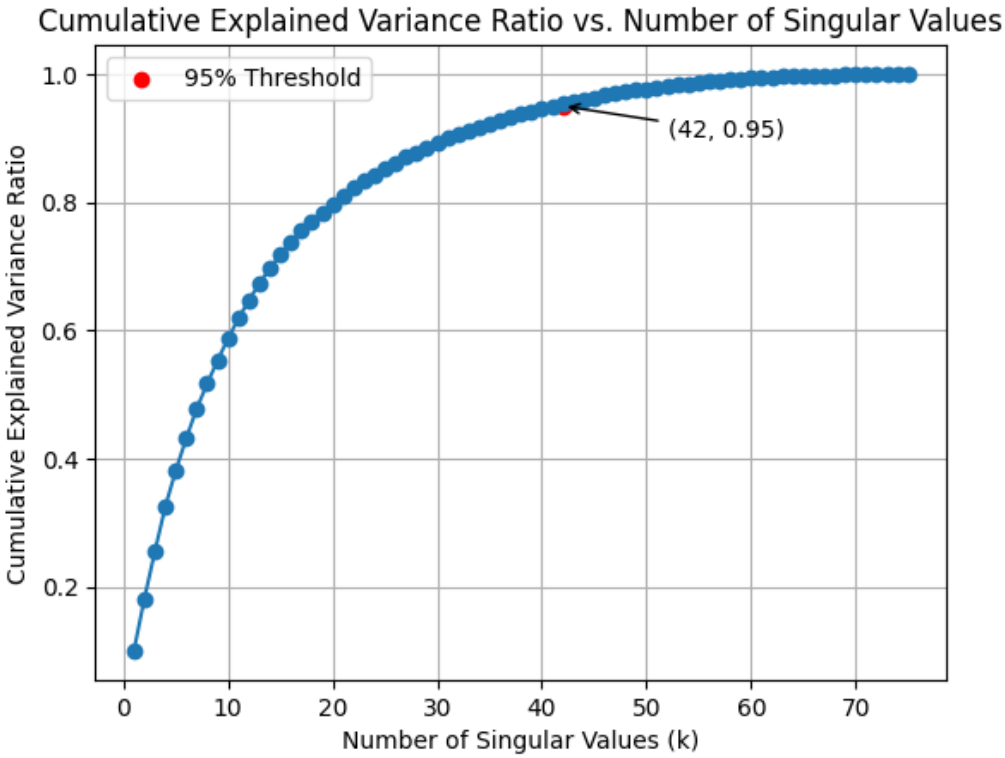

Figure 67: Cumulative Variance Ratio for 40% noisy images, removed images using NLM

- Top 10 flow modes of 40% noisy images, removed images using NLM.

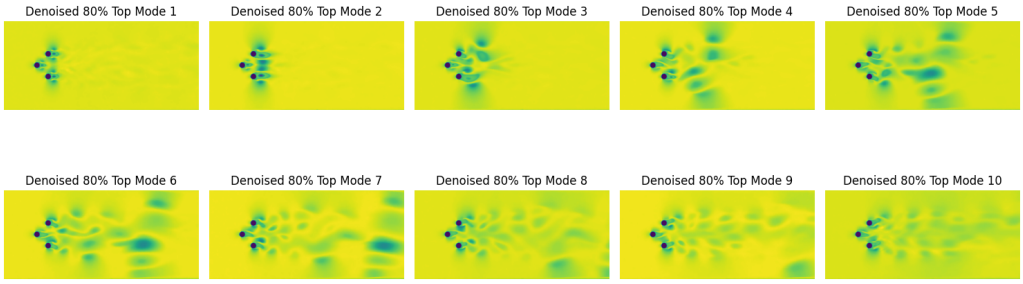

Figure 68: Top 10 modes 40% noise removed images using NLM

- Total Energy by every 10 modes for 40% noisy images, removed images using NLM.

  These 10 modes posses about 61.43% energy of total energy.[Refer code file]

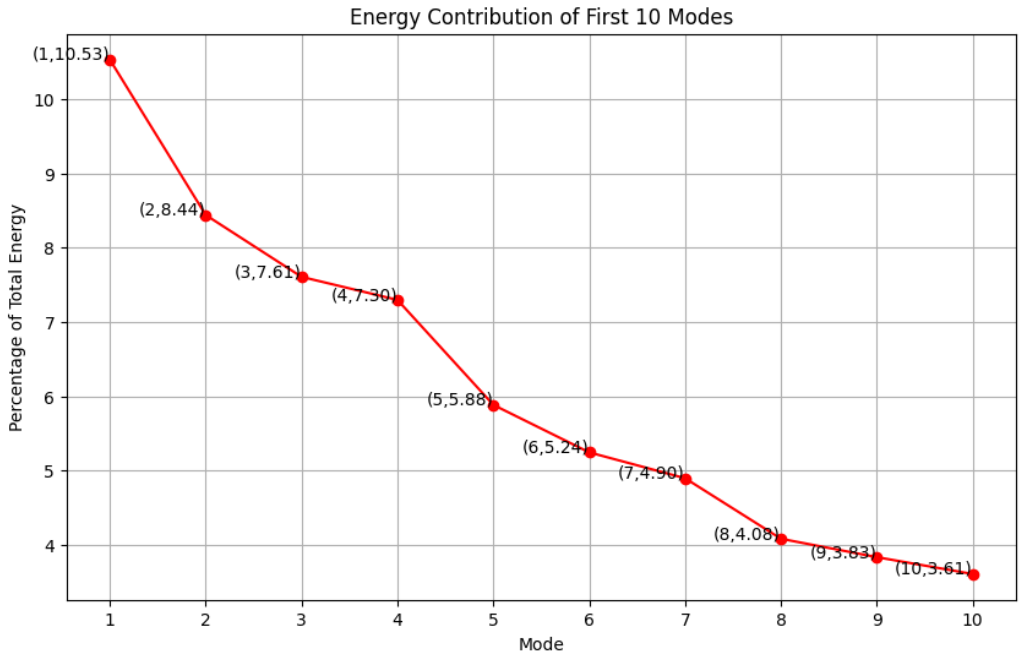

Figure 69: Energy of each 10 modes for 40% noisy images, removed using NLM

POD for 60% Gaussian noise images using NLM
Doing the same as we have done in subsection 3.3.

- Mean image

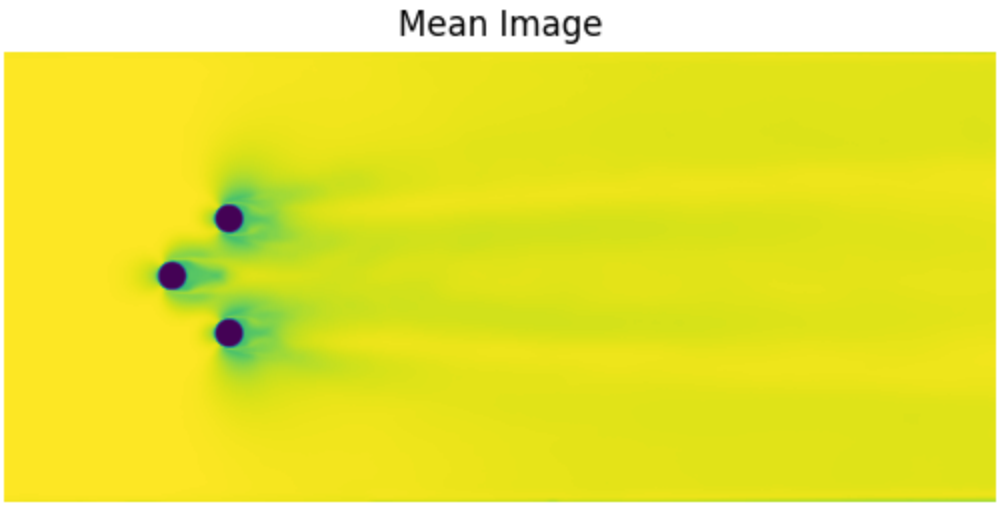

Figure 70: Mean 60% NLM

- Cumulative Variance Ratio

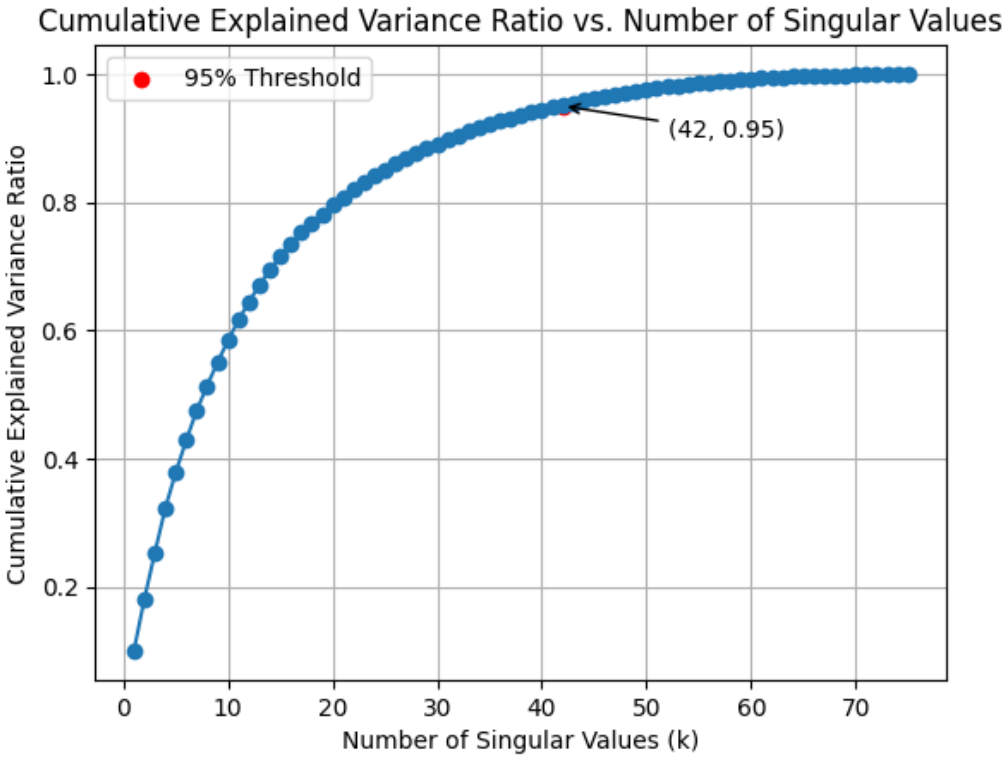

Figure 71: Cumulative Variance Ratio for 60% noisy images, removed images using NLM

- Top 10 flow modes of 60% noisy images, removed images using NLM.

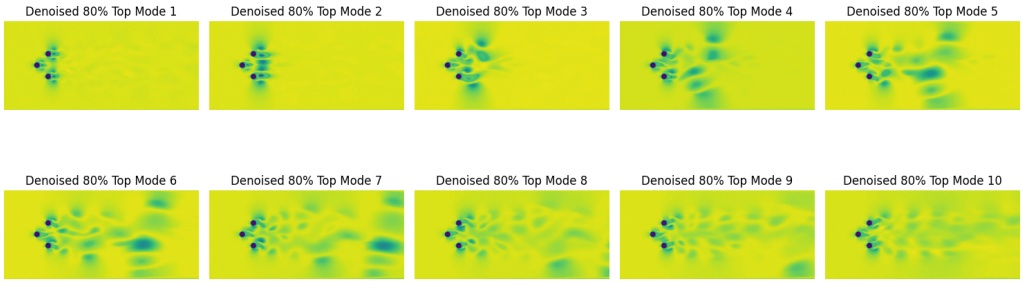

Figure 72: Top 10 modes 60% noise removed images using NLM

- Total Energy by every 10 modes for 60% noisy images, removed images using NLM.

  These 10 modes posses about 61.42% energy of total energy.[Refer code file]

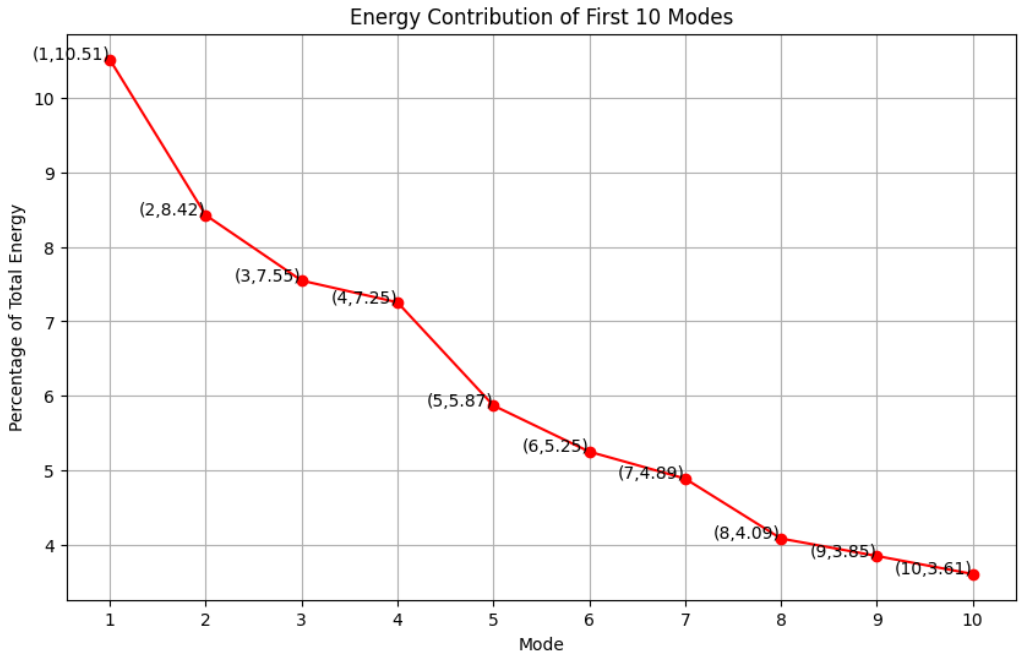

Figure 73: Energy of each 10 modes for 60% noisy images, removed using NLM

POD for 80% Gaussian noise images using NLM
Doing the same as we have done in subsection 3.3.

- Mean image

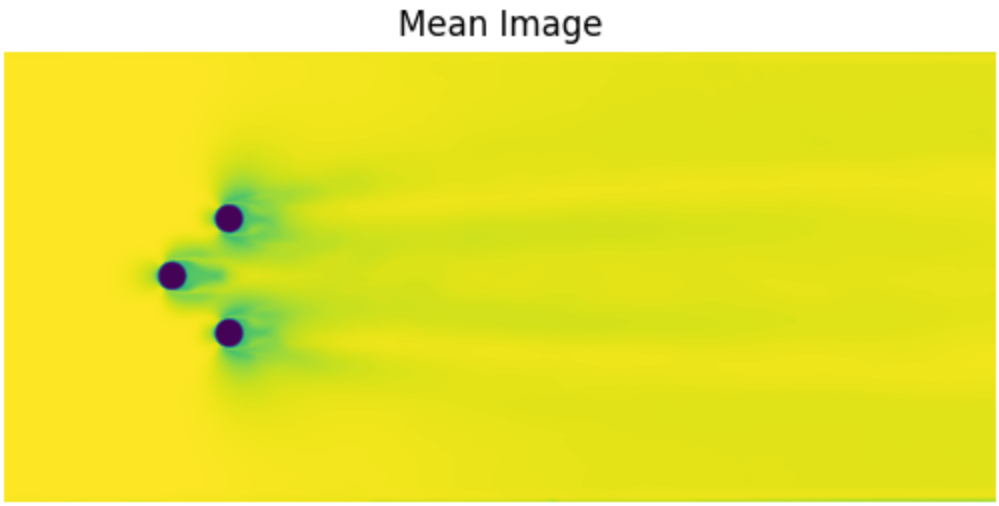

Figure 74: Mean 20% NLM

- Cumulative Variance Ratio

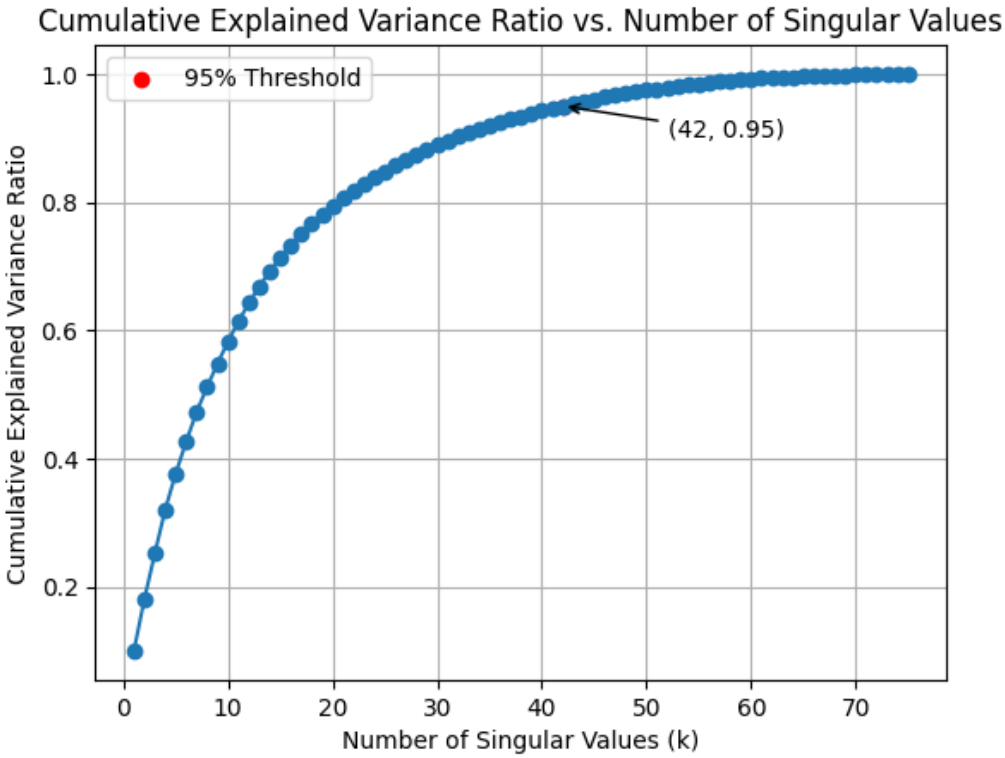

Figure 75: Cumulative Variance Ratio for 80% noisy images, removed images using NLM

- Top 10 flow modes of 80% noisy images, removed images using NLM.

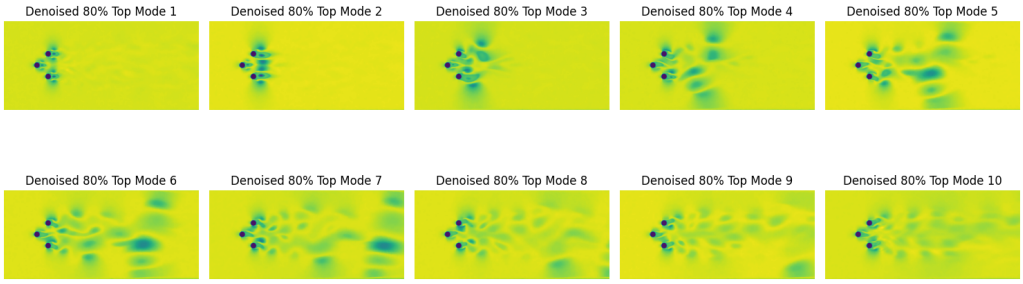

Figure 76: Top 10 modes 80% noise removed images using NLM

- Total Energy by every 10 modes for 80% noisy images, removed images using NLM.

  These 10 modes posses about 61.30% energy of total energy.[Refer code file]

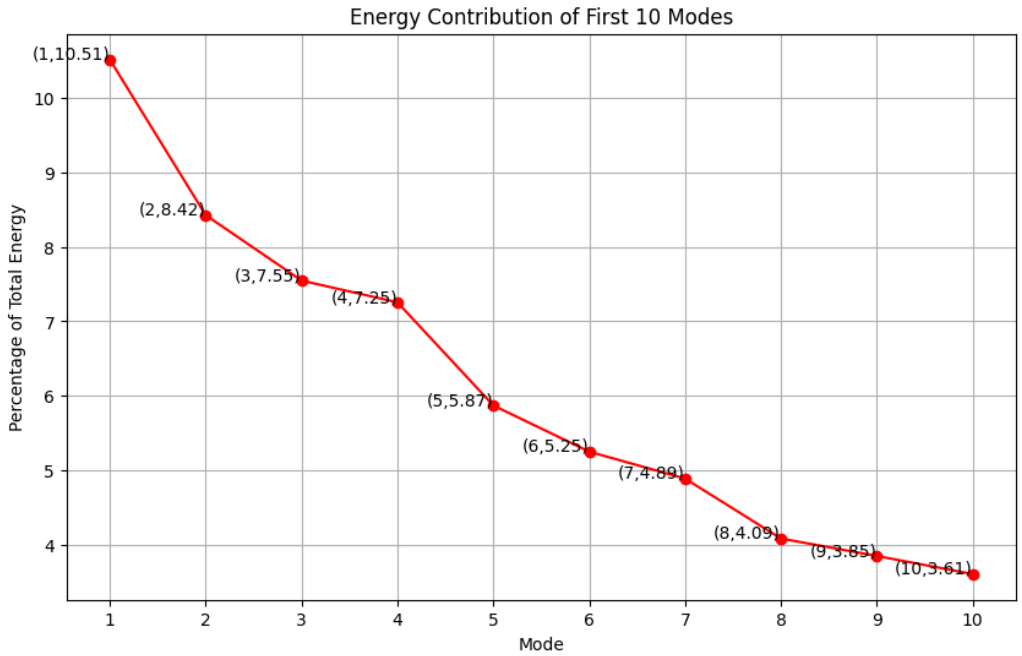

Figure 77: Energy of each 10 modes for 80% noisy images, removed using NLM

RealImageEnergy = 60.88%	20%	40%	60%	80%
Noisy Image energy %	60.88%	53.98%	45.19%	38.97
Denoised image energy %	62.15%	61.63%	61.42%	61.30

Table 5: Comparison of energy of noisy images and denoised images

As we can see energy contribution values [Refer: Table 5] is nearly about real energy value i.e. 60.88%. Here it can be observed that energy even increased after denoising. This is because NLM operates by comparing patches of pixels across the image and averaging similar patches to estimate the noise-free value for each pixel. Which allows it to preserve edges and fine details better than simple filtering methods like Gaussian blur.

References

[1] Abdi, H. (2007). Singular value decomposition (svd) and generalized singular value decomposition. *Encyclopedia of measurement and statistics*, 907(912):44.

[2] Brunton, S. (2022). Applying machine learning to study fluid mechanics. *Acta Mechanica Sinica*, 37.

[3] Larrañaga, A., Brunton, S. L., Martínez, J., Chapela, S., and Porteiro, J. (2023). Data-driven prediction of the performance of enhanced surfaces from an extensive cfd-generated parametric search space. *Machine Learning: Science and Technology*, 4(2):025012.

[4] Murata, T., Fukami, K., and Fukagata, K. (2020). Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. *Journal of Fluid Mechanics*, 882:A13.

[5] Pawar, S., San, O., Aksoylu, B., Rasheed, A., and Kvamsdal, T. (2021). Physics guided machine learning using simplified theories. *Physics of Fluids*, 33(1).

[6] Raissi, M., Yazdani, A., and Karniadakis, G. E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030.

[1] [2] [3] [4] [5] [6]

SVD through example

GeeksForGeeks Article on SVD

Medium Article on SVD