MEL2040 Project
Data-Driven Analysis of Fluid Flows

Aditya Pote B22ME005 and Shardul Date B22me077

April 16, 2024

Abstract

Your abstract.

1 Review of Machine Learning in Fluids

Introduction to Fluid Mechanics in Machine Learning :- The field of fluid mechanics, which is a fundamental discipline within engineering and science, handles large volumes of data from laboratory experiments, numerical simulations and field measurements. Although data processing techniques have advanced rapidly over the years, the analysis of fluid mechanics data has relied on domain-specific knowledge, stochastic methods and rule-based algorithms. However, this trend is changing following exponential growth in the amount of available data coupled with developments in computational hardware as well as new algorithms. This has reignited interest in machine learning (ML) application to fluid mechanics-related research.

Historical Context :- The intersection of ML and fluid dynamics traces back to the early 1940s when Kolmogorov identified turbulence as a prime application domain for statistical learning theory. While ML experienced significant advancements in the mid-20th century, interest waned due to limitations in capabilities and computational resources. Pioneering work by Rechenberg and Schwefel in fluid mechanics laid the groundwork for future applications of randomized optimization techniques. The 1974 Lighthill report, critiquing AI programs in the UK, indirectly affected funding and interest in ML, leading to the "AI winter." However, the late 1980s saw a resurgence of interest in ML with the development of the backpropagation algorithm, enabling the training of neural networks with multiple layers. In the early 1990s, NNs found applications in flow-related problems such as trajectory analysis and classification for particle tracking velocimetry (PTV) and particle image velocimetry (PIV).

Applications of Machine Learning in Fluid Mechanics :-

3.1. Clustering and Classification

Clustering and classification techniques play a pivotal role in data analysis in fluid dynamics research. K-Means clustering has been employed to discretize high-dimensional phase spaces, enabling tractable models for temporal flow evolution. Classification techniques, such as neural networks, have been utilized for wake topology classification and flow disturbance detection, showcasing their versatility in discerning various flow behaviors and regimes.

3.2. Sparse and Randomized Methods

Sparse optimization and randomized linear algebra techniques complement ML in fluid mechanics, facilitating efficient acquisition and reconstruction of sparse signals. These methods offer accelerated computations and accurate matrix decompositions, essential for reducing computational costs and enhancing computational efficiency in fluid dynamics simulations.

3.3. Superresolution and Flow Cleansing Superresolution techniques, leveraging ML algorithms, aim to infer high-resolution images from low-resolution measurements, thereby enhancing the fidelity of flow field measurements in experimental fluid dynamics. Flow cleansing methodologies, enabled by ML, address challenges such as noise, outliers, and missing data in experimental PIV and particle tracking, ensuring the accuracy and reliability of flow measurements.

3.4. In fluid dynamics, Proper Orthogonal Decomposition (POD) provides an empirical-based orthogonal basis for complex geometries, simplifying computations through singular value decomposition. Originally introduced by Sirovich in 1987, POD extends beyond fluid dynamics, influencing

modern computer vision. It closely aligns with Principal Component Analysis (PCA), a cornerstone in applied statistics and machine learning, which compresses high-dimensional data into a compact representation resembling POD/PCA decomposition. Utilizing the universal approximation theorem, deep neural networks (DNNs) offer nonlinear embeddings for intricate fluid flows. While DNNs excel in interpolation, extrapolation may be challenging if data distributions differ from training data. Despite the fluid mechanics community's distance from this paradigm, curated large and labeled fluid databases could enable the deployment of deep learning algorithms in future research.

The resurgence of interest in ML, coupled with advancements in deep learning architectures, has propelled the integration of ML techniques into various aspects of fluid mechanics research. From clustering and classification to sparse and randomized methods, and super-resolution and flow cleansing techniques, ML offers versatile tools for enhancing data analysis, simulation, and experimental measurements in fluid dynamics. This convergence of ML and fluid mechanics holds promise for transforming our understanding of complex flow phenomena and driving innovation in fluid dynamics research.

Deep reinforcement learning (DRL) has recently been adopted in a wide range of physics and engineering domains for its ability to solve decision-making problems that were previously out of reach due to a combination of non-linearity and high dimensionality. In the last few years, it has spread in the field of computational mechanics, and particularly in fluid dynamics, with recent applications in flow control and shape optimization. In this work, we conduct a detailed review of existing DRL applications to fluid mechanics problems. In addition, we present recent results that further illustrate the potential of DRL in Fluid Mechanics. The coupling methods used in each case are covered, detailing their advantages and limitations. Our review also focuses on the comparison with classical methods for optimal control and optimization. Finally, several test cases are described that illustrate recent progress made in this field. The goal of this publication is to provide an understanding of DRL capabilities along with state-of-the-art applications in fluid dynamics to researchers wishing to address new problems with these methods.

2 New Ideas

2.1 use of ML for prediction of new shapes to optimise flows

2.1.1

Absolutely, machine learning can be utilized to predict new shapes for optimizing flow in fluid mechanics. Here are several ways this could be achieved:

1. Generative Adversarial Networks (GANs) for Shape Generation: Generative Adversarial Networks (GANs) can be trained on a dataset of known fluid flow shapes and their corresponding performance metrics (e.g., drag coefficient, pressure distribution). Once trained, the GAN can generate novel shapes that are likely to optimize flow characteristics based on the learned patterns from the training data.

2. Reinforcement Learning for Shape Optimization: Reinforcement learning algorithms can be employed to iteratively optimize fluid flow shapes by rewarding designs that improve flow characteristics and penalizing those that degrade performance. This approach could be particularly useful for complex geometries where traditional optimization methods may struggle.

3. Topology Optimization with Machine Learning: Traditional topology optimization methods aim to find the optimal distribution of material within a given design space to achieve desired mechanical or fluid flow properties. Machine learning can enhance this process by learning patterns from previous optimization iterations and guiding the search towards promising regions of the design space.

4. Surrogate Models for Shape Optimization: Surrogate models, such as Gaussian Process Regression or neural networks, can be trained to approximate the relationship between shape parameters and flow performance metrics. These surrogate models can then be used within optimization algorithms to efficiently explore the design space and identify shapes that optimize flow characteristics.

By leveraging machine learning techniques, engineers and researchers can explore a vast space of potential fluid flow shapes, identify novel designs that enhance performance, and accelerate the optimization process compared to traditional methods. These approaches have the potential to revolutionize the design of fluid flow systems in various applications, including aerospace, automotive, and energy.

2.2 ML can be used to shift from symbollic represnetation to distuributed representation of data and can this be used in ML

1. From Symbolic to Distributed Representation: Symbolic representation often relies on explicit rules and structures to represent knowledge, which can be limited in capturing complex patterns and relationships in data. Machine learning, particularly deep learning, offers an alternative approach by learning distributed representations of data through neural networks. These representations capture features and relationships in a distributed manner across multiple dimensions, allowing for more nuanced and flexible representations of complex data.

2. Word Embeddings and Natural Language Processing (NLP): In natural language processing tasks, machine learning models such as Word2Vec, GloVe, and BERT learn distributed representations (embeddings) of words and sentences from large text corpora. These embeddings capture semantic and syntactic similarities between words and enable more effective processing of natural language data compared to traditional symbolic approaches.

3. Representation Learning in Computer Vision: In computer vision, convolutional neural networks (CNNs) learn hierarchical representations of visual data through layers of convolutional and pooling operations. These learned representations capture increasingly abstract features of images, allowing CNNs to perform tasks such as object recognition, image classification, and semantic segmentation without relying on handcrafted symbolic features.

4. Graph Neural Networks (GNNs): Graph neural networks are a class of machine learning models designed to work with graph-structured data, such as social networks, citation networks, and molecular structures. GNNs learn distributed representations of nodes and edges in a graph, enabling tasks such as node classification, link prediction, and graph classification without relying on symbolic graph representations or explicit feature engineering.

By shifting from symbolic representation to distributed representation, machine learning models can learn more expressive and generalizable representations of data, leading to improved performance across various tasks. Additionally, incorporating representation learning techniques within machine learning algorithms enables more effective processing and understanding of complex data types such as text, images, and graphs.

2.3 Autonomous Fluid Flow Control Systems for Energy Harvesting in Urban Environments

Idea Overview: Develop machine learning algorithms to autonomously control fluid flow in urban environments, such as air flow around buildings or water flow in urban waterways, to optimize energy harvesting from renewable sources.

Details: 1. *Sensor Networks*: Deploy dense networks of sensors across urban areas to collect real-time data on fluid flow patterns, environmental conditions, and energy availability from renewable sources like wind and water currents.

2. *Machine Learning Models*: Develop machine learning models that analyze sensor data to predict optimal fluid flow configurations for maximizing energy harvesting from renewable sources. These models could be based on reinforcement learning, where the system learns over time to adjust fluid flow based on feedback from energy harvesting systems.

3. *Fluid Flow Control Mechanisms*: Implement autonomous control mechanisms, such as adjustable airfoil surfaces or smart waterway gates, that can dynamically manipulate fluid flow patterns based on recommendations from the machine learning models.

4. *Integration with Renewable Energy Systems*: Integrate the autonomous fluid flow control system with renewable energy harvesting systems, such as wind turbines or hydroelectric generators, to directly convert optimized fluid flow into usable energy.

Benefits: - *Maximized Energy Harvesting*: By autonomously optimizing fluid flow patterns, the system can significantly increase the efficiency of renewable energy harvesting in urban environments. - *Improved Sustainability*: Increased reliance on renewable energy sources reduces dependence on fossil fuels, leading to reduced greenhouse gas emissions and improved environmental sustainability. - *Urban Livability*: Enhanced energy harvesting contributes to a more sustainable and resilient urban infrastructure, improving the quality of life for city residents.

This idea leverages the capabilities of machine learning to optimize fluid flow dynamics in urban environments, leading to more efficient energy harvesting from renewable sources and contributing to

a cleaner and more sustainable future.

2.4 Smart Fluid-Based Traffic Management Systems for Urban Mobility

Idea Overview: Utilize machine learning algorithms to optimize traffic flow and reduce congestion in urban areas by dynamically controlling fluid-like traffic patterns.

Details: 1. *Traffic Flow Modeling*: Develop machine learning models that simulate traffic flow dynamics using principles from fluid mechanics. These models would consider factors such as vehicle density, speed, and road network topology to predict how traffic patterns evolve over time.

2. *Real-Time Data Integration*: Integrate real-time data sources, such as traffic cameras, GPS devices, and vehicle sensors, to continuously update the traffic flow models. This data provides insights into current traffic conditions and allows the system to make dynamic adjustments.

3. *Control Strategies*: Implement control strategies inspired by fluid dynamics to optimize traffic flow. This could involve dynamically adjusting traffic signals, lane configurations, and speed limits to alleviate congestion and prevent traffic jams.

4. *Adaptive Learning*: Utilize machine learning techniques to adaptively learn from historical traffic data and optimize control strategies over time. The system would continuously improve its effectiveness at managing traffic based on feedback from real-world performance.

Benefits: - *Reduced Congestion*: By optimizing traffic flow in real-time, the system can reduce congestion and travel times for commuters, leading to improved efficiency and productivity in urban areas. - *Lower Emissions*: Smoother traffic flow reduces stop-and-go driving, which can help lower fuel consumption and emissions from vehicles, contributing to improved air quality and reduced environmental impact. - *Enhanced Safety*: By preventing traffic congestion and reducing the likelihood of accidents, the system can enhance overall road safety and reduce the risk of collisions.

This idea harnesses machine learning to create smarter, more efficient traffic management systems that emulate the fluid-like behavior of traffic flow. By optimizing traffic patterns in real-time, the system can help create more sustainable and livable urban environments.

3 Proper Orthogonal Decomposition or POD

First we created the dataset from the video given to us which had CFD simulation of fluid flowing over three cylinder. We want to apply POD. POD in a very broad sense, decomposes any dynamical system into its spatial and temporal components.

Similarity between PCA and POD: Both PCA and POD are mathematical techniques used for dimensionality reduction and feature extraction. They aim to decompose a dataset into its constituent components to capture the most important information while discarding noise or less relevant features. They both involve finding orthogonal vectors (principal components or modes) through linear transformations and eigen analysis, with the goal of representing the data in a lower-dimensional space while preserving as much variance as possible.

Dissimilarity between PCA and POD: While PCA is a general-purpose technique applicable to various fields, POD is specifically tailored for analyzing fluid dynamics, particularly turbulent flows. The principal components in PCA may not have a clear physical interpretation, whereas the modes in POD represent coherent spatial and temporal structures in the flow field. PCA imposes orthogonality constraints on the principal components, whereas orthogonality is a fundamental property of the modes in POD. Additionally, PCA can be applied to diverse data types, while POD is primarily used for spatiotemporal data sets in fluid dynamics.

We performed POD using singular value decomposition (SVD).The process begins by constructing a snapshot matrix. By performing SVD on this matrix, it is decomposed into three matrices: the left singular vectors (spatial modes), singular values, and right singular vectors (temporal modes). The singular values represent the magnitudes of the spatial and temporal modes and capture the amount of variance explained by each mode. Overall, SVD serves as a fundamental tool in POD analysis, enabling the extraction of meaningful spatial and temporal modes from high-dimensional data sets, facilitating insights into complex dynamical systems.

3.1 Image Generation

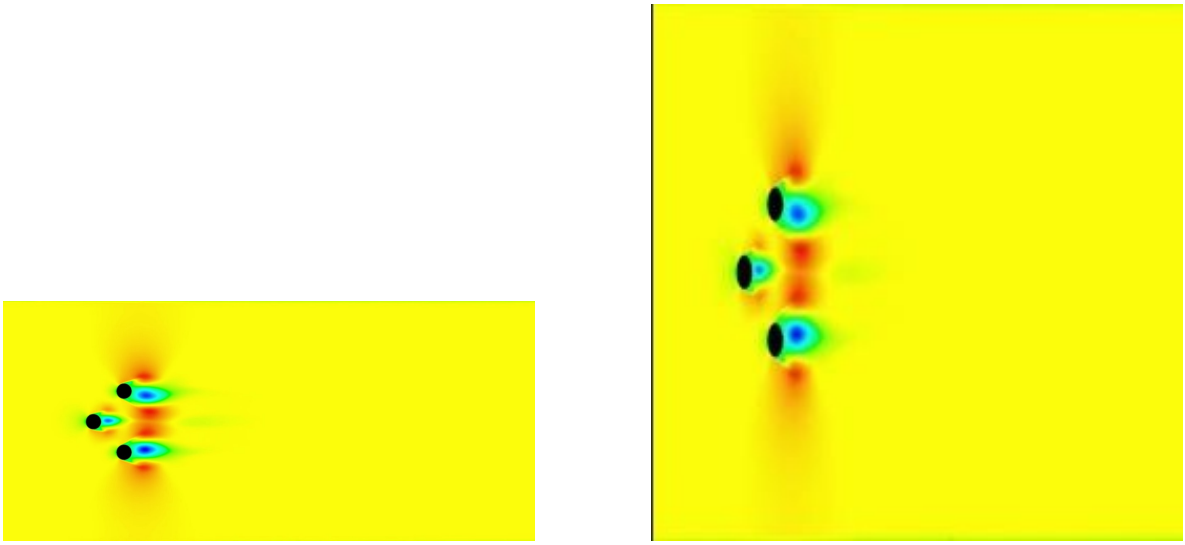
1) Taken out the frames from the Video. Video captured in 24fps and exact duration was 31.25 seconds. Therefore we get total 24*31.25 equals to 751 frames in total. and stacked them time wise frame by frame.

2) Downscaled the image to 256*256 resolution as POD on original dataset was tough due to high dimentionality.

3) Vectorized the images into 1-D array SO that it will be easy to apply mathematics and ML on it.

```python
def extract_frames(video_path, scale_size=(256, 256)):
    frames = []
    cap = cv2.VideoCapture(video_path)
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        frame = cv2.resize(frame, scale_size)
        frames.append(frame)
    cap.release()
    return frames
```

Code to extract frames from Video



(a) Original instantanious image

(b) Downscaled Image

Figure 1: Comparison of a and b images

3.2 Execute POD

Executed the POD. Used Singular Value Decomposition (SVD) to compute U,S,V matrices. We tried to write the code of SVD from scratch, but computations were very difficult and hence we decided to use inbuild functions for computing U,S,V matrices.

Mathematical Formulations :

$$X' = X - \bar{X}$$

Where:

- X' is the mean-subtracted data matrix.
- X is the original data matrix.
- $\bar{X}$ is the mean vector, where each element of $\bar{X}$ corresponds to the mean value of the corresponding feature across all samples.

$$X' = U\Sigma V^T$$

Where:

- X' is the mean-subtracted data matrix.
- U is an $n \times n$ matrix containing the left singular vectors (modes).
- Σ is an $n \times m$ diagonal matrix containing the singular values (energy).
- V^T is the transpose of an $m \times m$ matrix containing the right singular vectors.

Reducing Dimensionality : To reduce the dimensionality of the data while retaining most of its information, truncate the matrices U, Σ, and V^T to keep only the first k columns (modes), where k is the desired number of dimensions.

This can be done by keeping only the first k columns of U, Σ, and V^T, resulting in truncated matrices U_k, Σ_k, and V_k^T.

Code snippet :- This is the POD code. We tried to apply SVD from scratch but there problems due to heavy computations, our system was crashing , memory error, etc. So we decided to use inbuild function for POD which helped us to compute U,S,V matrices.

```
def perform_pod(frames_stack):
    # Reshape the stack of frames into a 2D array
    num_frames = frames_stack.shape[0]
    frames_flattened = frames_stack.reshape(-1,num_frames)

    # Compute mean and subtract it from the data
    mean_frame = np.mean(frames_flattened, axis=1)
    # print(mean_frame.shape)
    frames_mean_subtracted = np.zeros(frames_flattened.shape)
    for i in range(num_frames):
        frames_mean_subtracted[:,i] = frames_flattened[:,i] - mean_frame

    # Singular Value Decomposition (SVD)
    U, S, Vt = np.linalg.svd(frames_mean_subtracted, full_matrices=False)

    # Compute the POD modes
    pod_modes = U

    return U, S, mean_frame, Vt
```

3.3 Analyse POD Modes

On performing POD, we get the 751 Singular values of POD as well same number of modes. Every mode have different value while the values are very close to each other. We Plot Singular value vs indices of singular value. In fluid dynamics and machine learning, Proper Orthogonal Decomposition (POD) is a technique used to analyze and decompose complex flow fields into a set of spatial patterns and corresponding temporal coefficients. When applying POD to fluid flow data, it's common to observe oscillatory behavior in the POD modes.

These oscillations typically arise due to the inherent dynamics of the fluid flow system. Fluid flows often exhibit oscillatory behavior, such as vortex shedding behind obstacles, oscillations in boundary layers, or periodic instabilities in the flow field. These oscillations manifest themselves in the dominant spatial patterns identified by the POD analysis.

In the context of machine learning applied to fluid flow analysis, understanding and characterizing these oscillatory properties in the POD modes can provide valuable insights into the underlying flow physics. It can help in tasks such as flow prediction, turbulence modeling, or flow control strategies.

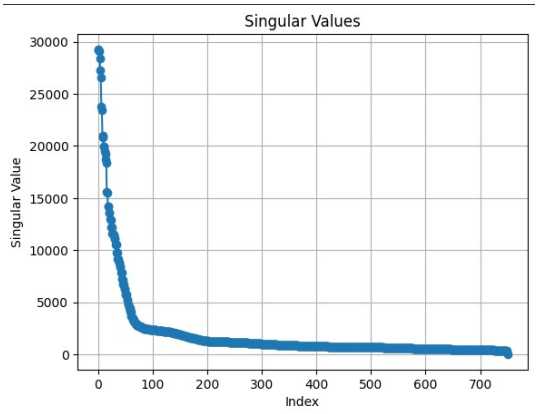
Researchers often study the temporal dynamics of the POD modes to extract information about the frequencies, amplitudes, and phases of the oscillations. This information can then be used to model and predict the behavior of the fluid flow more accurately.

Overall, recognizing the oscillatory nature of POD modes in fluid flow analysis is essential for leveraging POD effectively as a tool for understanding and modeling complex flow phenomena.

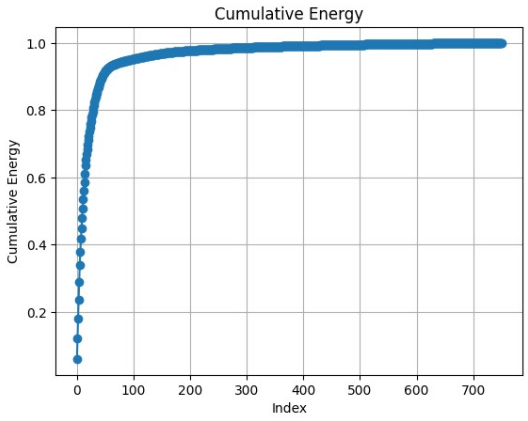
Top 10 modes by energy are

```
[ 0.00029306  0.00148329  0.00121147 ... -0.00013126  0.00016805
 -0.00122615]
[ 0.00032099  0.00055456 -0.00043679 ... -0.00131702 -0.00114378
  0.00034639]
[-0.00085145 -0.00022534 -0.00019212 ...  0.00083721  0.00118353
 -0.00494907]
[-7.16928914e-05 -5.76548159e-04  3.27797943e-04 ...  1.61450010e-03
  2.42924731e-03 -5.26417169e-04]
[ 5.28227197e-04 -8.61463231e-05 -1.68920413e-04 ...  9.08048308e-04
  1.48876966e-04 -2.46632236e-04]
[-4.63030200e-04  7.07775483e-05 -2.44799520e-04 ...  2.81047106e-04
  1.24031143e-03  2.56953069e-03]
[-0.00040827  0.00086147  0.00094082 ...  0.00081267  0.00105914
  0.00026869]
[-1.37555372e-03  2.03715353e-04 -3.66950657e-04 ...  3.90235975e-04
 -2.02888752e-05  9.05469557e-04]
[ 2.51973088e-05  5.75831869e-04  5.28852982e-04 ...  1.00895652e-04
  1.37110189e-03 -1.97565096e-04]
[-0.00167612  0.00108102  0.00052149 ... -0.0005155  -0.00013252
  0.00166281]
```

Figure 2: Top 10 modes values of matrix



(a) Singular Value vs index plot



(b) Cumilative energy vs Plot

From Graph (a) below we can Conclude that that value of singular value in S matrix keep on decreasing and tends to 0. This shows that importance of the mode vector corresponding to bigger

Singular value contributes more and vice versa. Also from Graph (b) we can conclude that aproximately 250 modes are enough to represent the image or energy state of velocity flow field. while to achieve 80 percent of retention, aproximately 20 modes are enough to represent the energy state.

4 Noise!

In computational fluid dynamics (CFD) simulations and experimental techniques like particle image velocimetry (PIV) and Schlieren imaging, various sources of noise can significantly affect the accuracy of modal studies and subsequent analysis. The following types of noise are commonly encountered:

Digital Noise:

Arises from errors in data acquisition and processing. In CFD simulations, it can result from discretization errors and numerical instability. In PIV and similar techniques, it may stem from camera noise and interpolation errors. Impact: Introduces spurious fluctuations in data, obscuring true flow structures and affecting mode identification. Sensory Errors:

Refers to inaccuracies in sensors or instruments used for data capture. Includes errors in pressure sensors, temperature probes, and velocity measurement devices. Impact: Introduces biases or distortions in data, leading to inaccuracies in modal analysis and mode interpretation. Equipment Imperfections:

Optical distortions in PIV systems and mechanical vibrations in wind tunnels are common examples. Impact: Introduces spatial or temporal variations in data, distorting mode shapes and frequencies, and complicating flow behavior interpretation. Understanding the effects of noise on modal studies is essential for assessing data reliability and uncertainty. Techniques such as noise filtering, error estimation, and sensitivity analysis can mitigate noise impact and enhance the accuracy of modal analysis in fluid dynamics.

4.1 Adding Noise

So adding the noise to our images. We are using two different methods for adding noise. First is using Gaussian blur noise and second is rayleigh noise. We are adding different percentage of noises with respect to original images. These are 0.2, 0.4, 0.6, 0.8 the maximum magnitude of the individual frames.

1) Gaussian Noise:

- Electronic Noise: Often exhibits a Gaussian distribution due to the random nature of electronic fluctuations.
- Optical Noise: Imperfections like scattering and diffraction can contribute to noise with a Gaussian distribution.
- Environmental Noise: Variations in environmental conditions like temperature and pressure can introduce Gaussian noise into measurements.

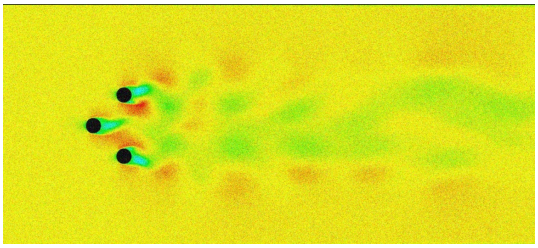
2) Rayleigh Noise:

Optical Noise: Rayleigh scattering, which is proportional to the inverse fourth power of the wavelength, can contribute to noise in optical systems.

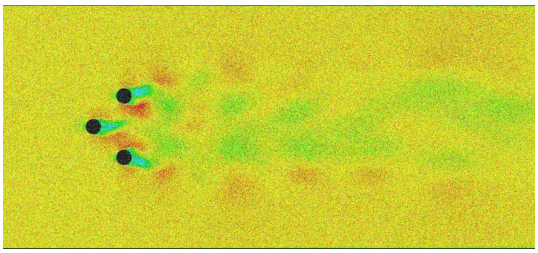
- We added Rayleigh noise as sown in figure.
- noise percentages are [20, 40, 60 , 80]
- Stored all images in folder and performed POD on it.

4.2 Effect on POD Modes

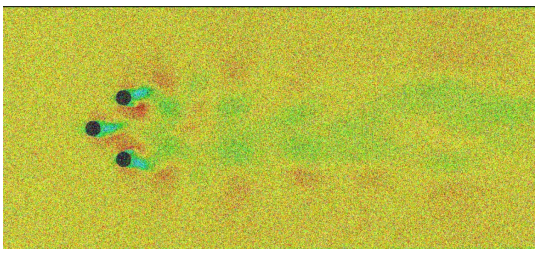
In fluid dynamics, when we're using Proper Orthogonal Decomposition (POD) to analyze flow data, the type of noise we choose depends on what we're trying to achieve and the kind of flow we're dealing with.



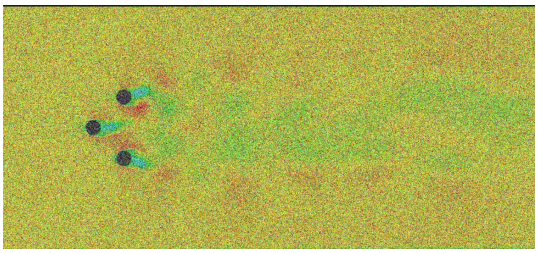
(a) 0.2 noise



(b) 0.4 noise

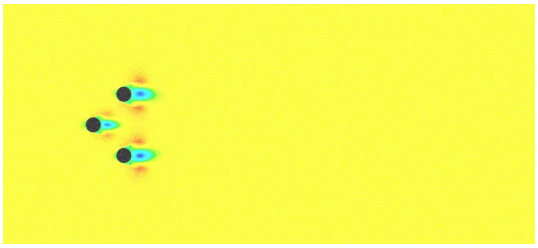


(c) 0.6 noise

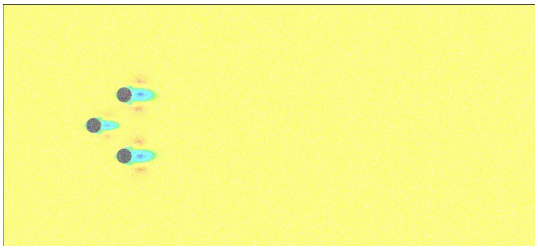


(d) 0.8 noise

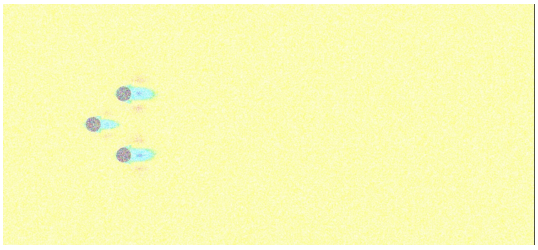
Figure 4: Comparison of Gaussian noise images



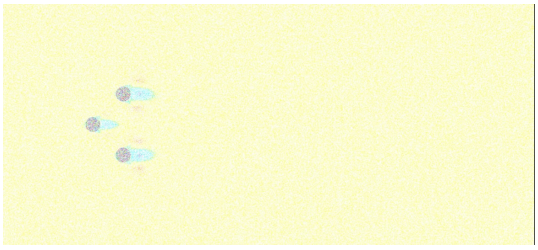
(a) 0.2 noise



(b) 0.4 noise

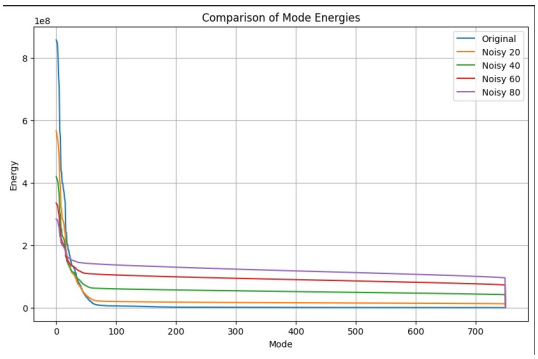


(c) 0.6 noise

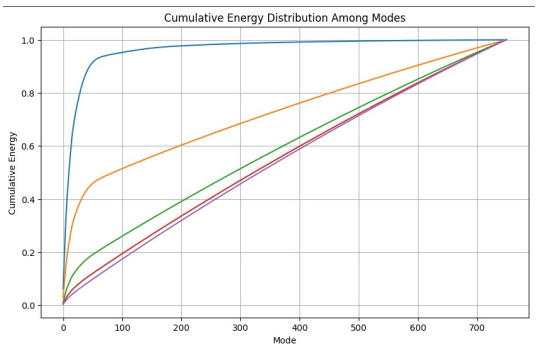


(d) 0.8 noise

Figure 5: Comparison of Rayleigh noise images

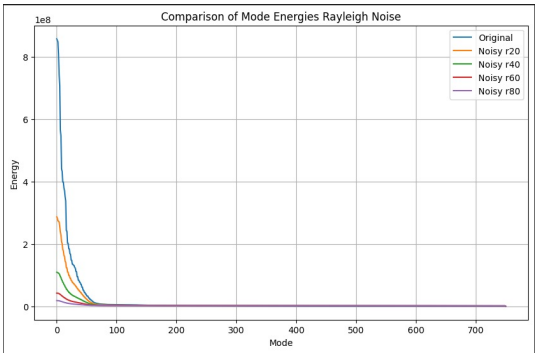


(a)

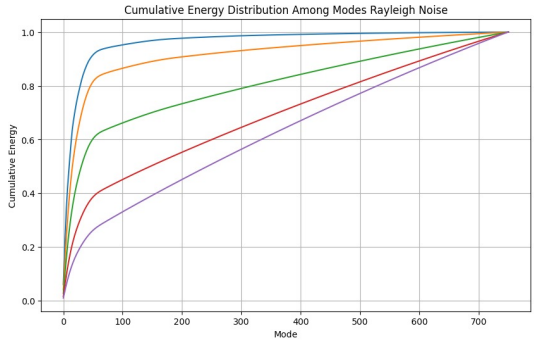


(b)

Figure 6: Plots for Gaussian Blur noise



(a)



(b)

Figure 7: Plots for Rayleigh noise

POD helps us pick out the main patterns and movements in a flow from a bunch of data, like from experiments or computer simulations. The main goal is to spot the organized flow patterns and separate them from random noise and messy fluctuations.

We often go with Gaussian noise:

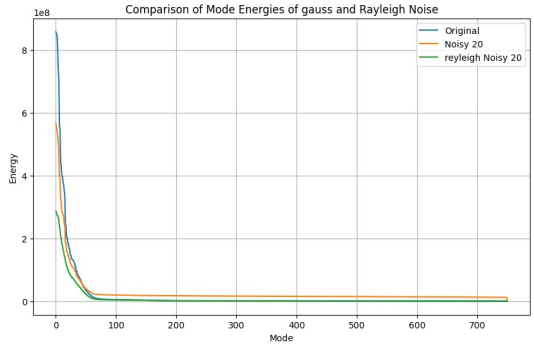
1. Statistical Properties: Gaussian noise is nice because we can describe it well with just its average (mean) and how spread out it is (standard deviation). This makes it easier to work with and tell apart from the actual flow patterns.

2. Independence: Gaussian noise usually behaves independently and uniformly across the data, which makes our number-crunching easier. We can use standard techniques like PCA without much fuss.

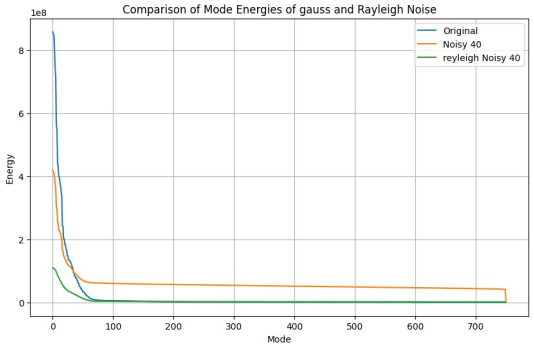
3. Similarity to Flow Features: Many times, the flow patterns we're interested in sort of resemble Gaussian distributions, especially when we're talking about big, turbulent movements.

4. Robustness: Gaussian noise can handle small changes and random bits in the data pretty well. This makes it handy for dealing with experimental data that might not be perfect.

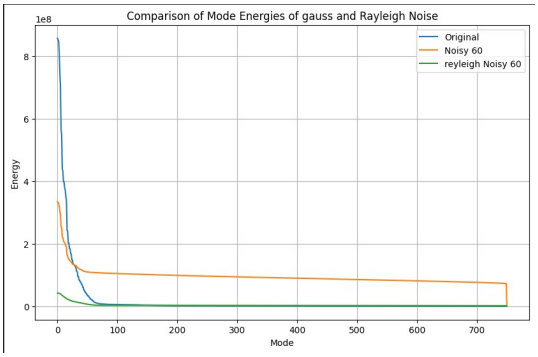
But the right noise choice depends on the specific flow and data. Sometimes, other types of noise might make more sense, like if there are weird blips now and then in the flow, or if there are optical quirks messing with measurements. So, it's all about understanding what's going on in the data and picking the best fit for the job.



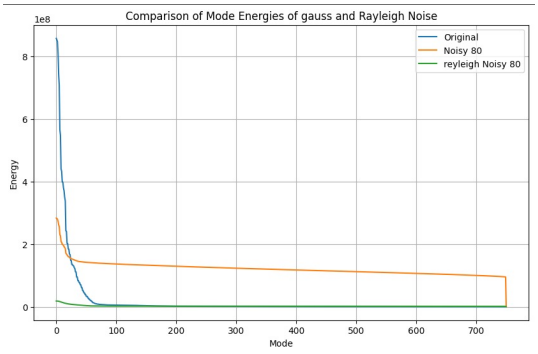
(a) 0.2 noise



(b) 0.4 noise



(c) 0.6 noise



(d) 0.8 noise

Figure 8: Comparison of Gaussian and Rayleigh noise images

5 Super-Resolving

there are different methods of super resolving 1. Gaussian Blur*: This method, as used in the provided program, applies a Gaussian blur filter to the image. It convolves the image with a Gaussian kernel to reduce high-frequency noise and blur the image. The cv2.GaussianBlur() function is used for this purpose.

2. *Median Filtering*: Median filtering replaces each pixel's value with the median value of the intensity levels in the neighborhood of that pixel. It is effective in removing salt-and-pepper noise

while preserving edges in the image.

3. *Bilateral Filtering*: Bilateral filtering is a non-linear filtering technique that smooths images while preserving edges. It considers both spatial closeness and intensity similarity when filtering, which makes it effective for denoising while preserving fine details.

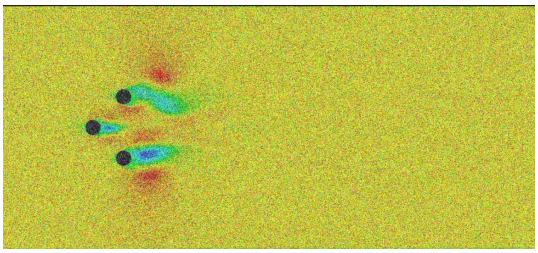
4. *Non-local Means Denoising*: Non-local Means Denoising (NLMeans) is a denoising algorithm that estimates the noise in an image by averaging similar patches from different areas of the image. It removes noise while preserving image details and textures.

5. *Wavelet Transform*: Wavelet-based denoising techniques decompose the image into different frequency bands using wavelet transforms. Noise is usually present in high-frequency components, so filtering out these components can effectively reduce noise while preserving image details.

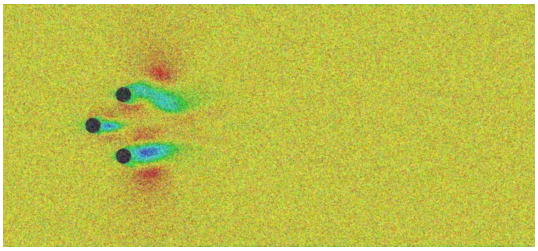
6. *Deep Learning-based Methods*: Recent advancements in deep learning have led to the development of denoising algorithms based on convolutional neural networks (CNNs). These methods learn to denoise images directly from data and can achieve state-of-the-art performance in image denoising tasks.

Each denoising method has its advantages and limitations, and the choice of method depends on factors such as the type and level of noise in the image, computational resources, and desired trade-offs between denoising effectiveness and preservation of image details.

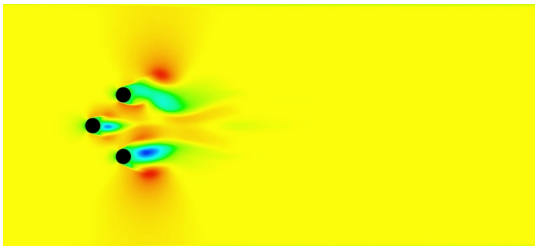
We try to use CNN and guassian blur denoiceing techniques. But we CNN was computationally very costly and took so much time for single image so we used Gaussian Blur for denoicing.



(a) Noised image

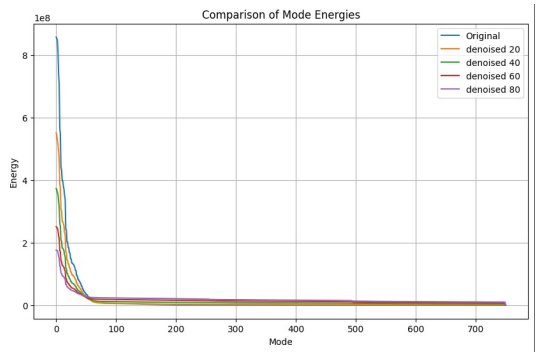


(b) Denoise image

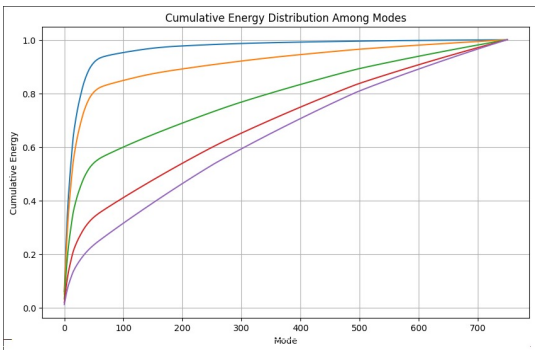


(c) True Image

Figure 9: comparison between noise and denoise



(a)



(b)

Figure 10: Comparison of True vs Denoise energy distribution

6 References

Internet helped us a lot in this research. We see multiple research papers to gain some valuable insight from it.

Research paper
https://www.annualreviews.org/docserver/fulltext/fluid/52/1/annurev-fluid-010719-060214.pdf?expires=1713174809i
https://www.sciencedirect.com/science/article/pii/S0045793021001407
https://drive.google.com/drive/folders/15qem91lS5XanKg$_{j}Jc0EKf9xVdg7H2mB$
YOutube source https://www.youtube.com/playlist?list=PLnO2sW0-XPrd$_{D}6lV5YBCaUTa-NsMUibJ$
ChatGPt - ChatGpt helped us to code as well as to resolve many syntax errors.