Statistical Analysis of NASA CRM USM3D Data of Pressure Coefficients

Submitted by: Muhammad Usman, UoB # 12036782

Analyzed Paper: "**An Integrated CFD/Experimental Analysis of Aerodynamic Forces and Moments**" (Melton, Robertson and Moyer, 1989)

Abstract: Machine learning and data science are solving Computational Fluid Dynamics (CFD) problems by low cost simulations in flow solvers. To achieve the accurate outcomes, these flow solvers needs to be validated before critical uses. This report analyses a work which validates FLO57 code with predictions using integrated pressures. The work is found to be well-described and covered most of the aspects with some limitations regarding the choice of flow solver itself. Using the analysis of this work, an open-source data of pressure coefficients at different locations of aircraft is analyzed using Python and results indicate a promising direction to be used with machine learning techniques for predicting critical variables in an aircraft simulation.

Keywords: CFD, Unsupervised Learning, Flow Solver Validation, Statistical Analysis (2100~ words)

1- Introduction

Aerodynamics has been revolutionized by the introduction of computing techniques for the product design and dependability benchmarking. From cost effective simulation of different objects to predicting complex events, computing generally and data science specifically is being used. As correctly discussed by Morrison et al., there is an inevitable need of validating and improving the simulations of the modern flow solvers because of immense reliance of aerodynamics future on them (Morrison et al., 2014).

In this report, a NASA technical memorandum is critically analyzed against some similar works and later been used as a guide to explore an extensive open-source data of NASA CRM for transonic commercial aircrafts (Melton, Robertson and Moyer, 1989). Moreover, using the similar work as a guide, statistical data analyses and unsupervised learning techniques will be applied on the NASA computational research model (CRM) data to find insight patterns in the data.

2- Report Structure

The report consists of two main parts in such a way that the first part of the report analyses NASA memorandum followed by statistical analysis of the NASA data in second part. As there was no specific research work which have used our specific data so a similar work of NASA was critically analyzed in the first part. The first part contains summary of paper, strong points and then weak points. The second part starts with the statistical approach and data resource explanation. After that, data resource was analyzed using different techniques. Lastly, the report is concluded by summarizing the keys points.

3- Paper Critical Analysis

3.1- Paper Summary

NASA's technical memorandum with title '**An Integrated CFD/Experimental Analysis of Aerodynamic Forces and Moments**' is analyzed in this report. This work validates Euler-based flow solver FLO57 results against the real life testing scenarios (Jameson, Schmidt and Turkel, 1981). Flow solver are those computer programs which are used for Computational Fluid Dynamics (CFD) simulations in the research (Peraire, Morgan and Peiro, 1990). This work verifies the results of this solver against experimental results and found some limitations in the solver's code (Melton et al., 1989). The methodology started by using PANAIR method code to predict surface pressure and the resulting forces acting within wind tunnel and then compared the predictions results with experimental results in the same tunnel (Carmichael and Erickson, 1981). After exploring and explaining each of the possible reasons for this results delta, a specific reason was finalized and solution was proposed.

3.2- Strong Points

Strongest point of the work is its novel contribution towards overcoming the overestimating errors in the flow solver's results by using a technique without increasing the number of pressure taps. The work discusses most of the possible reasons behind overestimation of predicted drag, which was more visible in higher angles of attack, as mentioned in the work Zhu et al. later (Zhu et al., 2007). Exploring the results in detail reveal that inadequate number of integrated pressures was the reason for this delta in results. Unlike the work done by Brodersen in which it was supposed that the error in drag prediction results were due to large size of grids but was not backed by concrete results or further exploration, this work actually finalized a specific reason and moved further (Brodersen, 2002). As increasing the number of pressure taps was not possible in the experiment setup so an alternate novel method was proposed in the work without increasing pressure taps or input variables as it was done in the work of Moore et al. (Moore, Wilcox and Hymer, 1994).

Apart from this, extensive comparison of results was done with various variables. As an interesting example, effect of grid density was evaluated on surface pressure coefficient. Though, the results show minimal effect, but this kind of comparison is not covered in other similar works.

Moreover, the solution is generic and not dependent on the speed with which aircraft is moving. This is a huge positive point for such solutions as it complies with Bell's work for Mach 0.4, 0.7 and 0.8 (Bell, 2011).

3.3- Weak Points

One of the weak areas of the work was its reliance on the incomplete code of flow solver which had limitations with some parameters specifically for nose-down moment of the aircraft. To be specific, the work did not cover surface and skin pressures near tail area for nose-down moment. Therefore, selecting a better and modern flow solver for this work would have saved from this limitation as Green used USM3D slow solver of NASA for working with nose down moments in F/A-18E predictions (Green, 2008).

The work also didn't clearly talked about the geometrical coordinates for which the results were being shown or compared the results at different positions of aircraft e.g. integration pressure at tail against the front part. A similar in-depth work is done by Rogers et al. while validating OVERFLOW solver when they

analyzed pressure coefficient at different locations and spans (Rogers, Roth and Nash, 2001) (Buning, 2019).

Lastly, it was observed that the work has not thoroughly covered the background literature and related works though this could be because of the fact that there was limited related work done at that time.

3.4- Comparative Evaluation with social legal and ethical issues

The discussed work are not solving the same problem so cannot be compared for the results on the same ground but similar open source data is being used in all of them. The used data is openly available and is supposed to be used in research so all the discussed works comply with the social and ethical issues. The work produced fairly accurate results which were later improved as well but didn't thoroughly compared the impact of predictions at different Mach levels as Moore's work did.

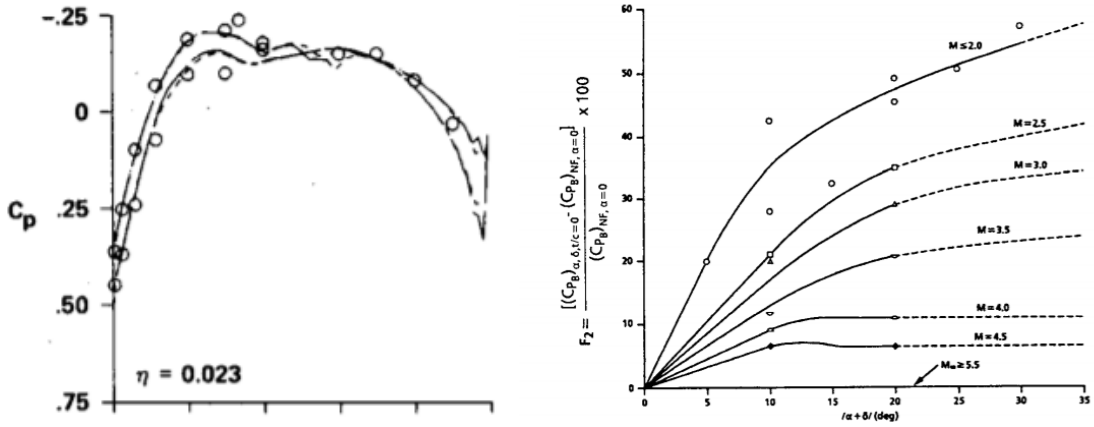


Figure 14: Melton et al. vs Moore's et al. (Melton et al., 1989) (Moore et al., 1994)

4- Statistical Analysis of NASA Data

4.1- Data resource

The data which is generated from NASA's USM3D flow solver containing X, Y, Z coordinate values and the respective coefficient of pressure against these coordinates (Rivers, 2020). This data was composed of 18 files with every file containing around 79,150 rows of data. Apart from this large amount of data, another challenge of the data was that every data file was again composed of nine parts and had some garbage values as well (Appendix figures 2 & 3). The division of data is described in more detail in section 'Data Details in appendix' and clearly shows the evidence of fulfilling big data features. Following is the explanation of these coordinates with more elaboration in figure 1 of appendix (Carter, 2019):

- X axis: Longitudinal axis of body (positive towards tail)
- Y axis: Span wise axis of body
- Z axis: Vertical direction of body (positive towards upward direction)

4.2- Data Statistics Method and Approach

Python is used for the statistical analysis of our data due to its rich libraries for data manipulation and analysis. Specifically, following packages were used:

1. **Pandas:** For seamless data import into program for processing with data frames (Pandas, 2020)

2. **Numpy:** Used for mathematical operations on N-dimensional data (NumPy, 2020)
3. **Scikit-learn:** Used for machine learning algorithms e.g. clustering (Scikit-learn, 2020)
4. **Matplotlib:** For data results visualization using plots in multidimensional space (Matplotlib, 2020)

4.3- Statistical Results of the Data in one original and one twisted file

Input Variable1, Input Variable2, Part of File	Original Data with A = 0		Twisted Data with A = 0	
	Covariance	Correlation	Covariance	Correlation
Y,Y,1	20971.78	0.08452954	20971.78	0.08452954
X,Z,1	25936.793	0.69496197	26050.377	0.69709504
Y,Z,1	6716.3887	0.28168446	6511.525	0.27273634
X,Z,2	-1.1370697	-0.46946806	21.310665	0.9019514
X,Z,3	-0.82781166	-0.48685014	23.782385	0.94440776
X,Z,4	-0.33646566	-0.31908268	25.804592	0.9714909
X,Z,5	-0.0026682455	-0.0043645003	0.06294047	0.10241257
X,Z,6	0.12307902	0.2554244	0.19460346	0.38543865
X,Z,7	0.14676991	0.377786	0.21610163	0.5149797
X,Z,8	0.1323501	0.4682612	0.18774217	0.60088134
X,Z,9	0.097555466	0.51134545	0.13154869	0.6258155
X,Z,10	0.086852655	0.6259187	0.111380816	0.7172074

Table 1: Statistical Results of data for Angle of Attack = 0

It can be seen from the results in Table 1 that correlation in twisted data is higher than that of the correlation in original data. Apart from this, all the data parts in twisted data is positively correlated but some parts in the original data file is negatively correlated for the values of X variable and Z. Especially in the highlighted cells of table 1, the correlation is very strong between X and Z which means using one of them as predictor variable could lead to the same results. These results can be further explored for formulating generalizations of such input variables for pressure coefficient as Apacoglu et al. did for similar nature work while exploring lift coefficient in their work (Apacoglu, Paksoy and Aradag, 2011). This can be achieved by computing necessary eigenvalues and eigenvectors representing the data using eigenvalue decomposition (EVD) or singular value decomposition (SVD) (Chatterjee, 2000).

4.4- Distribution of the data (Coefficient of Pressure)

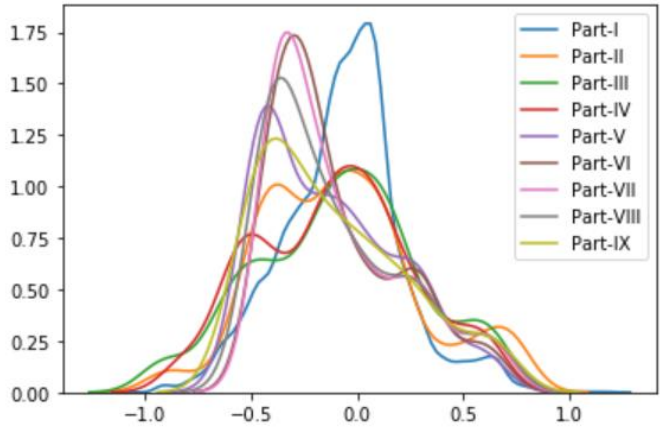


Figure 1: Coefficient of Pressure Distribution across different data parts for twisted data

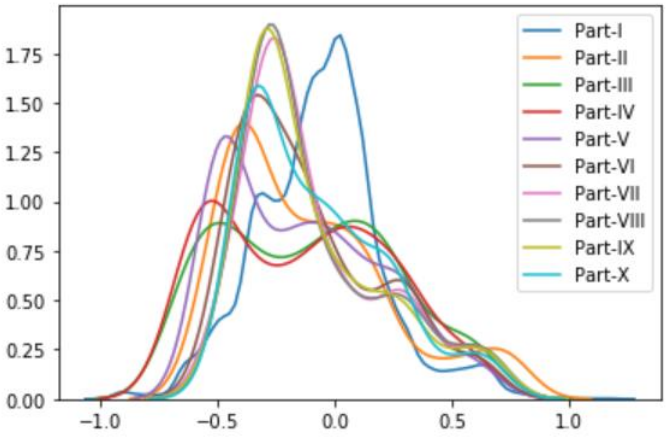


Figure 2: Coefficient of Pressure Distribution across different data parts for original data

It was observed in the graphs that original data part-I results were more uniformly distributed than the other results. Also, part-I data has most records so it was decided to compare these results for multiple values of alpha (Angle of attack). Therefore, part-I results against values of 0, 2, and 4 were compared.

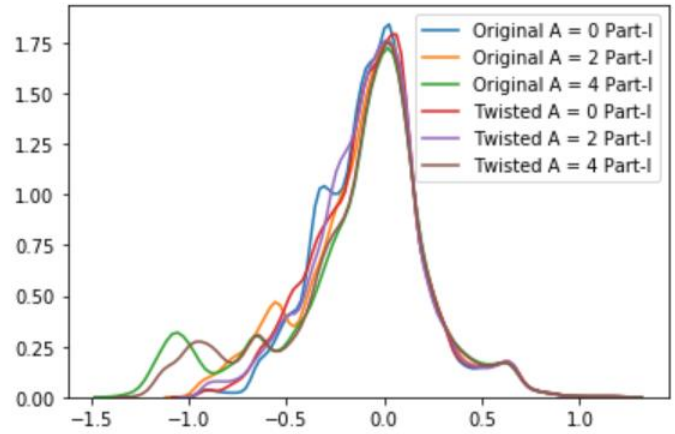


Figure 3: Coefficient of Pressure Distribution comparison across different angles in twisted and original data

The results show that distribution of data against alpha = 4 were a bit deviated from other values.

4.5- Mean of data A = 0

Part of the data	Mean of part (Twisted Data)	Mean of part (Original Data)
Part 1	-0.0669	-0.0616
Part 2	-0.0775	-0.1104
Part 3	-0.0868	-0.1047
Part 4	-0.1046	-0.1297
Part 5	-0.1255	-0.1320
Part 6	-0.1039	-0.1064
Part 7	-0.0947	-0.0873
Part 8	-0.1034	-0.0877
Part 9	-0.1116	-0.0952

Table 2: Mean of data for angle of attack = 0

Similar to the previous statistical approach, mean of pressure coefficients for part-I were calculated for different values of alpha.

	Mean of Pressure Coefficients	
Value of Alpha	Twisted Data	Original Data
0	-0.0669	-0.0616
2	-0.0687	-0.0878
4	-0.1238	-0.1364

Table 3: Mean of data for different angles of attack

4.6- Scatter Plot – Comparison of Pressure at different locations of aircraft

After data understanding, it was decided to find the relation of pressure coefficient with the coordinate values. That is, to see at which areas of the plane, the pressure would be highest. For this purpose, one of the files' data was sorted with respect to the values of pressure coefficient. Once the data was sorted, its top and bottom 50 records were separated and plotted using python.

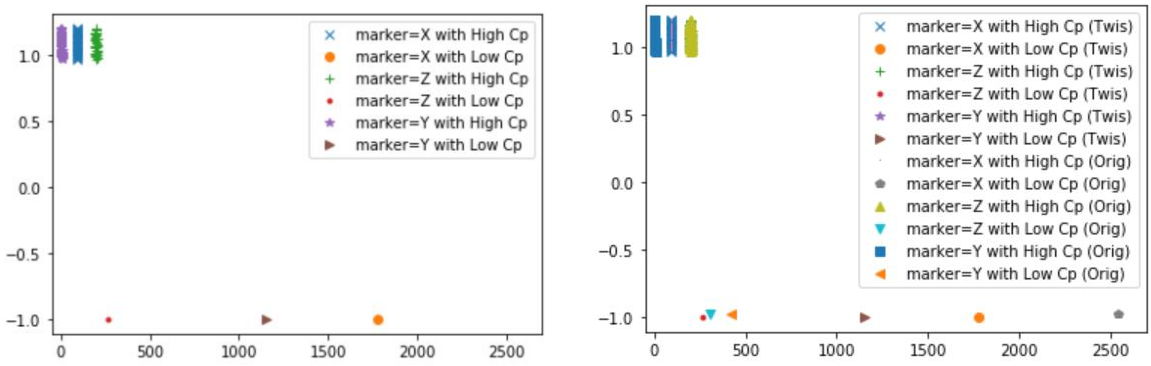


Figure 4: Scatter Plot for Twisted and Original Data file

The scatter plots reveal an interesting observation that the pressure coefficient is highest at the lowest coordinate values and low at the highest values. This observation is quite logical as the lowest coordinates are the front of plane and are more prone to the incoming wind. It can also be seen that there is a minimal difference in X,Y,Z coordinates of original and twisted data with high Cp values whereas the difference in clear for coordinates with low Cp.

4.7- Data Clustering

Once we found the covariance and correlation of the data, the next decided task to see the clustering. Using the scikit-learn library of Python (code snippet in appendix figure 4), data clustering was done to reveal patterns in the data.

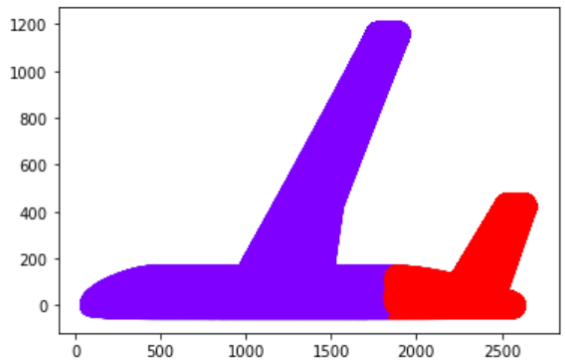


Figure 5: Two Clusters of Data

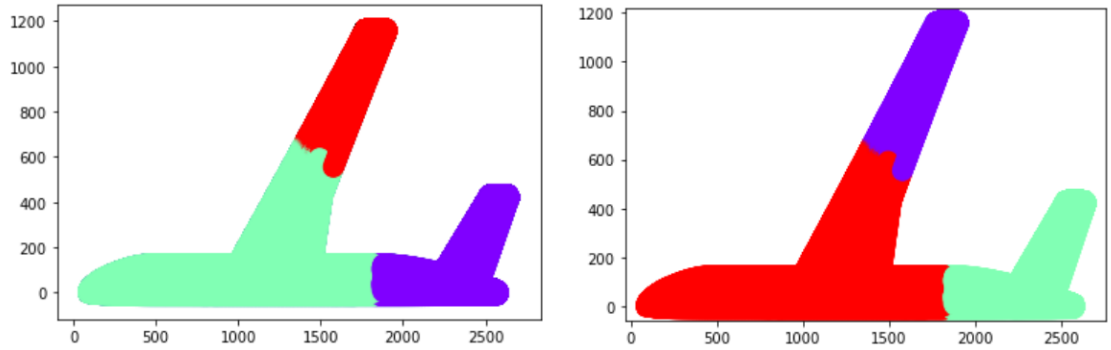


Figure 6: Three clusters of twisted and original data (Twisted Data on the left)

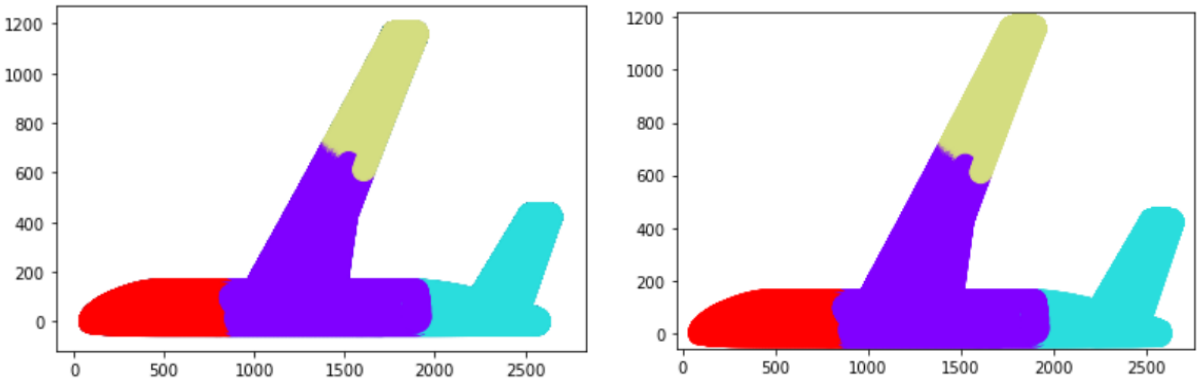


Figure 7: Four clusters of original and twisted data for a = 0 (Original Data on the left)

4.8- Scatter Plots for Frequency of Pressure Coefficients

Taking hint from the work in Hemsch 2004, it was seen that scatter plots can also reveal some insights to data if used as a tool to analyze results' frequencies.

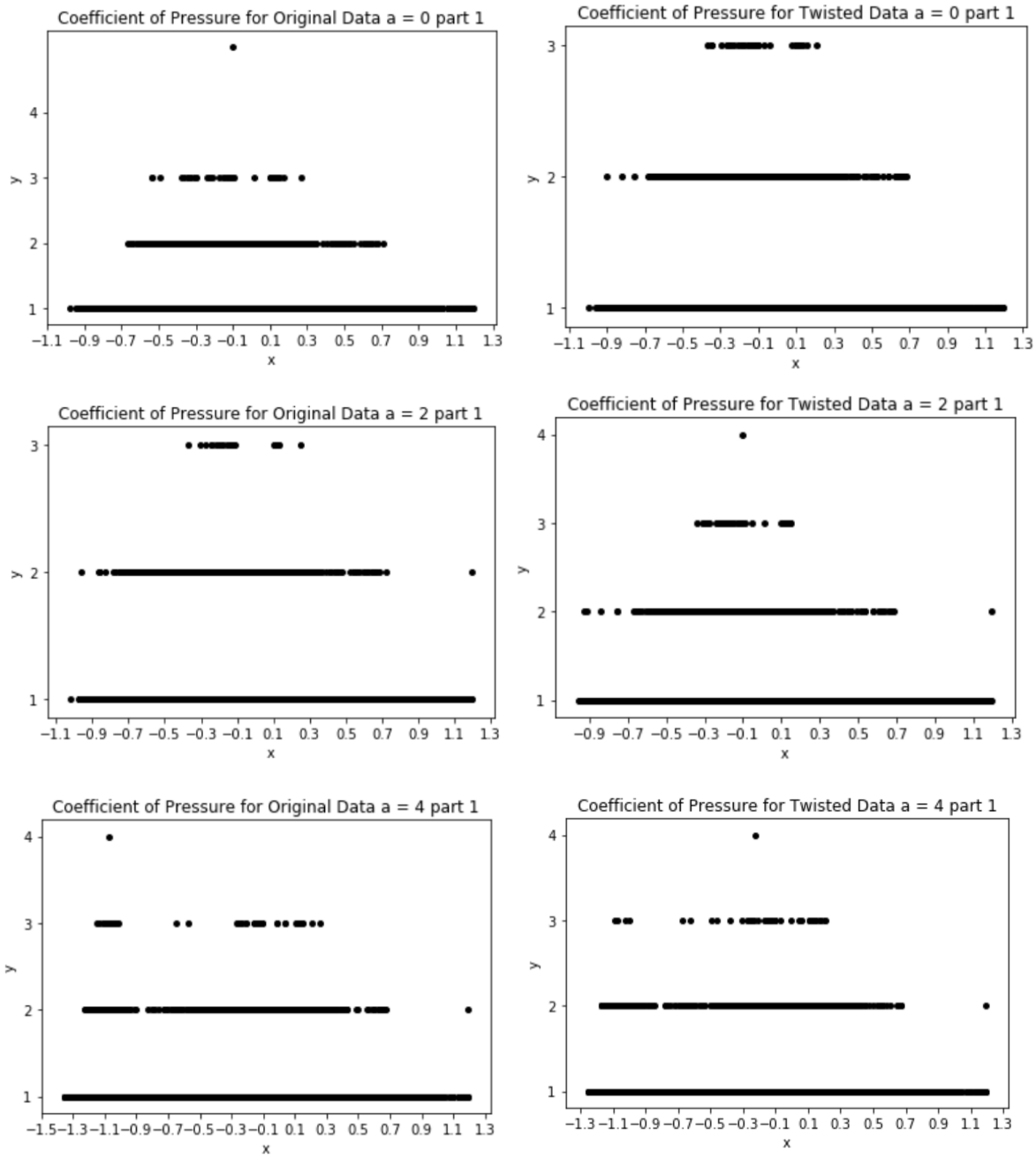


Figure 8 - 13: Comparisons of pressure coefficient frequencies in different data parts

Figures 8 – 13 further show an interesting statistics of data that for higher values of angle of attack, the pressure coefficient has more negative values than the lower angles. It also supports the overall results of minimal data differences between twisted and original results.

4.9- Association Rule Mining on the data

To further explore the data patterns, it was decided to apply association rule mining on the data. For this purpose, the data for one file with a = 0 was divided into two parts i.e. having positive and negative pressure coefficients. After that, all the input variable values were classified in four classes to see which range of input variables are producing positive or negative pressure.

Following table 4 describes the ranges for each of the attributes:

Attribute	Minimum	Maximum
X	92	2626.44
Y	-0.003	1159
Z	95	329

Table 4: Input variable ranges

As mentioned, all the attribute values were classified in four classes e.g. Very Low, Low, High, Very High. Following table describes the data division for each of the attribute:

Attribute	**Very Low Range**	**Low Range**	**High Range**	**Very High**
X	0-600	600-1200	1200-1800	1800<
Y	0-300	300-600	600-900	900<
Z	90-150	150-200	250-300	300<

Table 5: Variables classification

Following rules were extracted when Apriori Algorithm was applied on the classified data for positive values of pressure coefficients (Al-Maolegi and Arkok, 2014).

Rule	**Support**	**Confidence**	**Lift**
X Very Low ➔ Y Very Low	0.66	1.0	1.5
X Very Low ➔ Z Low	0.33	0.5	1.5
Z Low ➔ Y Very Low	0.33	0.5	1.5
X Very Low ➔ Z Low	0.33	0.5	1.5
Z Very Low ➔ X Very Low	0.33	0.5	1.5

Table 6: Rules for positive Cp

It can be seen from table 6 that low or very low values of variables are more likely to produce positive pressure coefficients which can be seen from the lift.

The same process with similar classification was applied to that data for which pressure coefficient was negative. As expected after the results mentioned earlier, the table 7 describes the rules for negative pressure coefficients for NASA data. It is clear from the following results, especially from the ones highlighted in green that very high values of Z is more probable to produce negative pressure coefficients.

Rule	Support	Confidence	Lift
Y Very High ➔ X High	0.33	1.0	1.5
Z Very High ➔ X High	0.33	1.0	3.0
X Very Low ➔ Y Very Low	0.33	0.5	1.5
Z Very Low ➔ X Very Low	0.66	1.0	1.5
Z Very High ➔ Y Very High	0.33	0.5	1.5
Z Very Low ➔ Y Very Low	0.33	1.0	1.5
Z Very High ➔ Y Very High	0.33	1.0	3.0
Z Very Low ➔ X Very Low	0.33	0.5	1.5

Table 7: Rules for negative Cp

5- Conclusion

In this report, a NASA memorandum validating FLO57 was critically analyzed against related research work and later the NASA data which was produced using USM3D was statistically analyzed with different techniques using Python. The work in memorandum is found to be novel and interesting as it proposed new solution to existing problems and discussed thorough results to support it. It could be very interesting to validate some modern CFD flow solvers using the similar approach which could possibly produce useful results. It would also be beneficial to validate the existing flow solvers against some unusual conditions as Head et al. validated solvers for non-ideal compressible flows (Head et al., 2017). In statistical analysis of NASA CRM dataset, it was revealed that high positive pressures are expected at the very low values of X, Y and Z coordinates of the airplanes and negative at high values. Depending on the statistical analysis results, it is recommended that predicting the surface pressure can be done using machine learning techniques as it is done in the work of Singh et al. for pressure coefficient and Zhang et al. for lift coefficient (Singh, Medida and Duraisamy, 2017) (Zhang, Sung and Mavris, 2018). Using such techniques can meet the future requirements of efficient simulation technologies and to validate them before using.

References

Apacoglu, B., Paksoy, A. and Aradag, S. (2011). CFD Analysis and Reduced Order Modeling of Uncontrolled and Controlled Laminar Flow Over a Circular Cylinder. *Engineering Applications of Computational Fluid Mechanics*, 5(1), pp.67-82.

Al-Maolegi, M., and Arkok, B. (2014). An improved Apriori algorithm for association rules. *arXiv preprint arXiv:1403.3948*.

Bell, J. (2011). Pressure-sensitive paint measurements on the NASA Common Research Model in the NASA 11-ft transonic wind tunnel. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition* (p. 1128).

Brodersen, O. (2002). Drag Prediction of Engine-Airframe Interference Effects Using Unstructured Navier-Stokes Calculations. *Journal of Aircraft*, 39(6), pp.927-935.

Buning, P. (2019). *NASA OVERFLOW CFD Code*. [online] Overflow.larc.nasa.gov. Available at: https://overflow.larc.nasa.gov/ [Accessed 31 Dec. 2019].

Chatterjee, A. (2000). An introduction to the proper orthogonal decomposition. *Current science*, pp.808-817.

Carter, M. (2019). *Overview of USM3D | TetrUSS: NASA Tetrahedral Unstructured Software System*. [online] Tetruss. Available at: https://tetruss.larc.nasa.gov/usm3d/overview/ [Accessed 30 Dec. 2019].

Carmichael, R. and Erickson, L. (1981). PAN AIR-A higher order panel method for predicting subsonic or supersonic linear potential flows about arbitrary configurations. In *14th Fluid and Plasma Dynamics Conference* (p. 1255).

Green, B. (2008). Computational Prediction of Nose-Down Control for F/A-18E at High Alpha. *Journal of Aircraft*, 45(5), pp.1661-1668.

Head, A., Iyer, S., de Servi, C. and Pini, M. (2017). Towards the Validation of a CFD Solver for Non-ideal Compressible Flows. *Energy Procedia*, 129, pp.240-247.

Jameson, A., Schmidt, W., and Turkel, E. (1981). Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes. In *14th fluid and plasma dynamics conference* (p. 1259).

NumPy. (2020). *NumPy — NumPy*. [online] Available at: https://numpy.org/ [Accessed 1 Jan. 2020].

Matplotlib. (2020). *Matplotlib: Python plotting*. [online] Available at: https://matplotlib.org/ [Accessed 1 Jan. 2020].

Melton, J., Robertson, D. and Moyer, S. (1989). An Integrated CFD/Experimental Analysis of Aerodynamic Forces and Moments. *National Aeronautics and Space Administration*.

Morrison, J.H., Kleb, W.L. and Vassberg, J.C. (2014). Observations on CFD Verification and Validation from the AIAA Drag prediction Workshops. In *52nd Aerospace Sciences Meeting* (p. 0202).

Moore, F., Wilcox, F. and Hymer, T. (1994). Base drag prediction on missile configurations. *Journal of Spacecraft and Rockets*, 31(5), pp.759-765.

Pandas. (2020). *Python Data Analysis Library — pandas: Python Data Analysis Library*. [online] Available at: https://pandas.pydata.org/ [Accessed 1 Jan. 2020].

Peraire, J., Morgan, K., and Peiro, J. (1990). Unstructured finite element mesh generation and adaptive procedures for CFD. *Application of Mesh Generation to complex*.

Rivers, M. (2020). *Computational Results | NASA Common Research Model*. [online] NASA Common Research Model. Available at: https://commonresearchmodel.larc.nasa.gov/computational-results/ [Accessed 3 Jan. 2020].

Rogers, S., Roth, K. and Nash, S. (2001). Validation of computed high-lift flows with significant wind-tunnel effects. *AIAA Journal*, 39, pp.1884-1892.

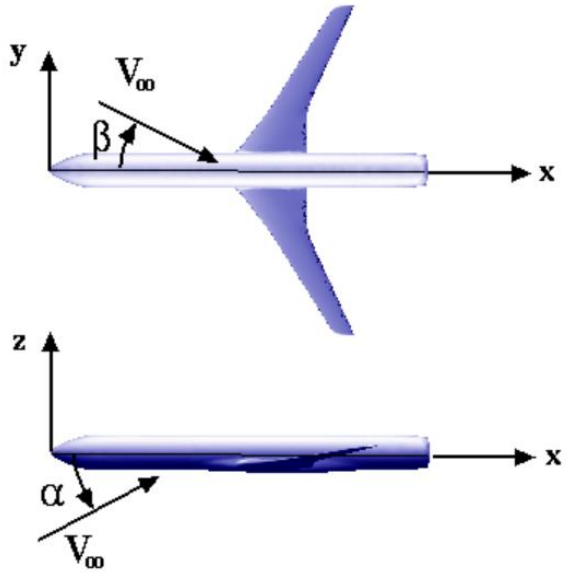
Singh, A., Medida, S. and Duraisamy, K. (2017). Machine-Learning-Augmented Predictive Modeling of Turbulent Separated Flows over Airfoils. *AIAA Journal*, 55(7), pp.2215-2227.

Scikit-learn. (2020). *Scikit-learn: machine learning in Python*. [online] Available at: https://scikit-learn.org/stable/ [Accessed 1 Jan. 2020].

Zhang, Y., Sung, W. J., and Mavris, D. N. (2018). Application of convolutional neural network to predict airfoil lift coefficient. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference* (p. 1903).

Zhu, Z., Wang, X., Liu, J. and Liu, Z. (2007). Comparison of predicting drag methods using computational fluid dynamics in 2d/3d viscous flow. *Science in China Series E: Technological Sciences*, 50(5), pp.534-549.

Appendix



Appendix Figure 1: Explanation of X,Y,Z in data

Data Details

There are three input variables X, Y, and Z whereas one output variable Cp which is coefficient of pressure. Following observations were made regarding data similarity:

1. Original Data Files
    a. Values of three input variables (X,Y,Z) in the data across all the files are same
    b. Only Cp value is changing when value 'a' is changed which denotes angle of attack (Appendix Figure 1)
2. Twisted Data Files
    a. Values of first three variables (X,Y,Z) in the data is similar in following way:
        i. a = 0 and a = 2 are same
        ii. a = 3 and a = 3.5 are same
        iii. a = 4.5 and a = 5 are same

Subparts of Data in File

76583	2625.77	419.704	302.068	0.01059
76584				
76585	Slc: Y=151.074			
76586	27.4155	4.079	4.75756	0.688476
76587	27.4156	4.079	4.75813	0.686934
76588	27.4172	4.079	4.7684	0.659955
76589	27.4188	4.079	4.77665	0.630599
76590	27.4156	4.079	4.757	0.690577
76591	27.4194	4.079	4.77959	0.620387
76592	27.4204	4.079	4.7835	0.602694

Appendix Figure 2: Subpart of data

Then after some results, the value of Y is again updated.

77280	39.5062	4.079	3.9445	0.04528
77281	67.5683	4.079	7.33774	0.126834
77282				
77283	Slc: Y=232.444			
77284	29.0753	6.27599	4.75981	0.557724
77285	29.0753	6.27599	4.75987	0.557498
77286	29.0766	6.27599	4.76977	0.520074
77287	29.078	6.27599	4.77659	0.487717
77288	29.0753	6.27599	4.75976	0.557917

Appendix Figure 3: Subpart of data (More parts like these)

```
kmeans = KMeans(n_clusters=4)
kmeans.fit(c)

plt.scatter(c.X,c.Y, c.Z, c=kmeans.labels_, cmap='rainbow')

plt.show()
```

Appendix Figure 4: Code for K-mean clustering

Association Rule Mining Implementation

```
from apyori import apriori


# X values = 92 - 2626.44
# Y values = -0.003 - 1159
# Z values = 95 - 329


######## Negative #########
# X values = 190 - 2626.44
# Y values = -0.003 - 1159
# Z values = 90 - 343


# classify the inputs in very low, low, high, very high
records = []
for row in negatives.X,negatives.Y,negatives.Z:
    item = []
```

```
curr_X = row[0]

if curr_X > 0 and curr_X < 600:
    item.append("X very low")
elif curr_X > 600 and curr_X < 1200:
    item.append("X low")
elif curr_X > 1200 and curr_X < 1800:
    item.append("X high")
elif curr_X > 1800:
    item.append("X very high")

curr_Y = row[1]
if curr_Y > 0 and curr_Y < 300:
    item.append("Y very low")
elif curr_Y > 300 and curr_Y < 600:
    item.append("Y low")
elif curr_Y > 600 and curr_Y < 900:
    item.append("Y high")
elif curr_Y > 900:
    item.append("Y very high")

curr_Z = row[2]
if curr_Z > 0 and curr_Z < 150:
    item.append("Z very low")
elif curr_Z > 150 and curr_Z < 200:
    item.append("Z low")
elif curr_Z > 250 and curr_Z < 300:
    item.append("Z high")
elif curr_Z > 300:
    item.append("Z very high")
```

```
    records.append(item)


association_rules = apriori(records, min_support=0.0045,
min_confidence=0.1, min_lift=1.5, min_length=2)
association_results = list(association_rules)
print(association_results)


print(positives[:5])


for item in association_results:
    print("Hello")
    # first index of the inner list
    # Contains base item and add item
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])

    #second index of the inner list
    print("Support: " + str(item[1]))

    #third index of the list located at 0th
    #of the third index of the inner list

    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("====================================")
```

Statistical Analysis Code

```
#!/usr/bin/env python
# coding: utf-8
```

```
# In[2]:


import pandas as pd
import io
import requests
import numpy as np

ta01=pd.read_csv(r"E:\NASA_Research\Twisted
Data\A0\twis_a0_part1.csv")
ta02=pd.read_csv(r"E:\N_Research\Twisted Data\A0\twis_a0_part2.csv")
ta03=pd.read_csv(r"E:\N_Research\Twisted Data\A0\twis_a0_part4.csv")
ta04=pd.read_csv(r"E:\N_Research\Twisted Data\A0\twis_a0_part5.csv")
ta05=pd.read_csv(r"E:\N_Research\Twisted Data\A0\twis_a0_part6.csv")
ta06=pd.read_csv(r"E:\N_Research\Twisted Data\A0\twis_a0_part7.csv")
ta07=pd.read_csv(r"E:\N_Research\Twisted Data\A0\twis_a0_part8.csv")
ta08=pd.read_csv(r"E:\N_Research\Twisted Data\A0\twis_a0_part9.csv")
ta09=pd.read_csv(r"E:\N_Research\Twisted Data\A0\twis_a0_part10.csv")
ta010=pd.read_csv(r"E:\N_Research\Twisted Data\A0\twis_a0_part11.csv")

ta21=pd.read_csv(r"E:\N_Research\Twisted Data\A2\twis_a2_part1.csv")
ta22=pd.read_csv(r"E:\N_Research\Twisted Data\A2\twis_a2_part2.csv")
ta23=pd.read_csv(r"E:\N_Research\Twisted Data\A2\twis_a2_part3.csv")
ta24=pd.read_csv(r"E:\N_Research\Twisted Data\A2\twis_a2_part4.csv")
ta25=pd.read_csv(r"E:\N_Research\Twisted Data\A2\twis_a2_part5.csv")
ta26=pd.read_csv(r"E:\N_Research\Twisted Data\A2\twis_a2_part6.csv")
ta27=pd.read_csv(r"E:\N_Research\Twisted Data\A2\twis_a2_part7.csv")
ta28=pd.read_csv(r"E:\N_Research\Twisted Data\A2\twis_a2_part8.csv")
ta29=pd.read_csv(r"E:\N_Research\Twisted Data\A2\twis_a2_part9.csv")
ta210=pd.read_csv(r"E:\N_Research\Twisted Data\A2\twis_a2_part10.csv")

ta41=pd.read_csv(r"E:\N_Research\Twisted Data\A4\twis_a4_part1.csv")
ta42=pd.read_csv(r"E:\N_Research\Twisted Data\A4\twis_a4_part2.csv")
ta43=pd.read_csv(r"E:\N_Research\Twisted Data\A4\twis_a4_part3.csv")
ta44=pd.read_csv(r"E:\N_Research\Twisted Data\A4\twis_a4_part4.csv")
ta45=pd.read_csv(r"E:\N_Research\Twisted Data\A4\twis_a4_part5.csv")
ta46=pd.read_csv(r"E:\N_Research\Twisted Data\A4\twis_a4_part6.csv")
ta47=pd.read_csv(r"E:\N_Research\Twisted Data\A4\twis_a4_part7.csv")
ta48=pd.read_csv(r"E:\N_Research\Twisted Data\A4\twis_a4_part8.csv")
ta49=pd.read_csv(r"E:\N_Research\Twisted Data\A4\twis_a4_part9.csv")
ta410=pd.read_csv(r"E:\N_Research\Twisted Data\A4\twis_a4_part10.csv")

oa01=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A0\orig_a0_part1.csv")
oa02=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A0\orig_a0_part2.csv")
oa03=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A0\orig_a0_part3.csv")
oa04=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A0\orig_a0_part4.csv")
oa05=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A0\orig_a0_part5.csv")
oa06=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A0\orig_a0_part6.csv")
oa07=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A0\orig_a0_part7.csv")
oa08=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A0\orig_a0_part8.csv")
oa09=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A0\orig_a0_part9.csv")
```

```
oa010=pd.read_csv(r"E:\N_Research\Orig
Data\Orig_A0\orig_a0_part10.csv")

oa21=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A2\orig_a2_part1.csv")
oa22=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A2\orig_a2_part2.csv")
oa23=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A2\orig_a2_part3.csv")
oa24=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A2\orig_a2_part4.csv")
oa25=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A2\orig_a2_part5.csv")
oa26=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A2\orig_a2_part6.csv")
oa27=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A2\orig_a2_part7.csv")
oa28=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A2\orig_a2_part8.csv")
oa29=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A2\orig_a2_part9.csv")
oa210=pd.read_csv(r"E:\N_Research\Orig
Data\Orig_A2\orig_a2_part10.csv")

oa41=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A4\orig_a4_part1.csv")
oa42=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A4\orig_a4_part2.csv")
oa43=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A4\orig_a4_part3.csv")
oa44=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A4\orig_a4_part4.csv")
oa45=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A4\orig_a4_part5.csv")
oa46=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A4\orig_a4_part6.csv")
oa47=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A4\orig_a4_part7.csv")
oa48=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A4\orig_a4_part8.csv")
oa49=pd.read_csv(r"E:\N_Research\Orig Data\Orig_A4\orig_a4_part9.csv")
oa410=pd.read_csv(r"E:\N_Research\Orig
Data\Orig_A4\orig_a4_part10.csv")


# In[3]:


import seaborn as sns
import matplotlib.pyplot as plt

sns.kdeplot(oa01.Cp, label="Original A = 0 Part-I")
sns.kdeplot(oa21.Cp, label="Original A = 2 Part-I")
sns.kdeplot(oa41.Cp, label="Original A = 4 Part-I")


sns.kdeplot(ta01.Cp, label="Twisted A = 0 Part-I")
sns.kdeplot(ta21.Cp, label="Twisted A = 2 Part-I")
sns.kdeplot(ta41.Cp, label="Twisted A = 4 Part-I")

plt.legend();


# In[16]:


import statistics

m1 = statistics.mean(oa01.Cp)
```

```
m2 = statistics.mean(oa21.Cp)
m3 = statistics.mean(oa41.Cp)
m4 = statistics.mean(ta01.Cp)
m5 = statistics.mean(ta21.Cp)
m6 = statistics.mean(ta41.Cp)

print(m1,m2,m3,m4,m5,m6)


# In[10]:


from sklearn.cluster import KMeans

ob_list = []
ob_list.append(oa01)
ob_list.append(oa21)
ob_list.append(oa41)
ob_list.append(ta01)
ob_list.append(ta21)
ob_list.append(ta41)

for var in ob_list:
    kmeans = KMeans(n_clusters=3)
    kmeans.fit(var)

    plt.scatter(var.X,var.Y, var.Z, c=kmeans.labels_, cmap='rainbow')

    plt.show()


# In[43]:


import matplotlib.pyplot as plt

def count_occ(df, val, my_dict):
    if val in my_dict:
        my_dict[val] += 1
    else:
        my_dict[val] = 1
#for obj in ob_list:
my_dict = dict()
for val in ta01.Cp:
        count_occ(ta01.Cp, val, my_dict)

keys = []
values = []

for key in my_dict:
    keys.append(key)
    values.append(my_dict[key])
```

```
colors = (0,0,0)
area = np.pi*5

plt.yticks(np.arange(1, 5, 1))
plt.xticks(np.arange(-1.5, 1.5, 0.2))
plt.scatter(keys, values, s=area, c=colors, alpha=1)
plt.title('Coefficient of Pressure for Twisted Data a = 0 part 1')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
#print(my_dict)


# In[43]:


sorted_frame = ta01.sort_values(by='Cp', ascending=False)
sorted_frame_orig = oa01.sort_values(by='Cp', ascending=False)
print("hello")
print(sorted_frame[0:5])
x_mean = statistics.mean(ta01.X)
y_mean = statistics.mean(ta01.Y)
z_mean = statistics.mean(ta01.Z)

x_min = min(ta01.X)
x_max = max(ta01.X)
y_min = min(ta01.Y)
y_max = max(ta01.Y)
z_min = min(ta01.Z)
z_max = max(ta01.Z)

print(x_mean, y_mean, z_mean)
print(x_min, x_max, y_min, y_max, z_min, z_max)

top = sorted_frame[0:50]
bottom = sorted_frame[len(sorted_frame.X)-1:]

orig_top = sorted_frame_orig[0:50]
orig_bottom = sorted_frame_orig[len(sorted_frame.X)-1:]

plt.plot(top.X,top.Cp, 'x',
             label="marker=X with High Cp (Twis)".format('x'))
plt.plot(bottom.X,bottom.Cp, 'o',
             label="marker=X with Low Cp (Twis)".format('o'))
plt.plot(top.Z,top.Cp, '+',
             label="marker=Z with High Cp (Twis)".format('+'))
plt.plot(bottom.Z,bottom.Cp, '.',
             label="marker=Z with Low Cp (Twis)".format('.'))
plt.plot(top.Y,top.Cp, '*',
             label="marker=Y with High Cp (Twis)".format('*'))
plt.plot(bottom.Y,bottom.Cp, '>',
```

```
             label="marker=Y with Low Cp (Twis)".format('>'))


plt.plot(orig_top.X,orig_top.Cp, ',',
             label="marker=X with High Cp (Orig)".format(','))
plt.plot(orig_bottom.X,orig_bottom.Cp, 'p',
             label="marker=X with Low Cp (Orig)".format('p'))
plt.plot(orig_top.Z,orig_top.Cp, '^',
             label="marker=Z with High Cp (Orig)".format('^'))
plt.plot(orig_bottom.Z,orig_bottom.Cp, 'v',
             label="marker=Z with Low Cp (Orig)".format('v'))
plt.plot(orig_top.Y,orig_top.Cp, 's',
             label="marker=Y with High Cp (Orig)".format('s'))
plt.plot(orig_bottom.Y,orig_bottom.Cp, '<',
             label="marker=Y with Low Cp (Orig)".format('<'))

plt.legend(numpoints=1)
plt.xlim(-50, 2700);
```