AMATH 582 Project

Lauren Jones, Elliot Jennis, Hariprasad Annamalai

March 19, 2021

Abstract

Dynamic Mode Decomposition is useful in many types of image and video analysis, ranging from facial recognition to studies of fluid flows. In this study, the fluid flow between counterflowing supersonic jets was examined in order to better understand the instabilities that have been observed in the resulting shock wave. In the dataset used for this paper, an oscillating shock was observed and its movement was characterized by the DMD. The shock front movement was tracked in order to quantify the change in the shock standoff distance and uncover the frequencies of oscillation to provide comparative results for the dominant DMD frequencies. The DMD showed that only a few modes (around 5-15) could be used in order to characterize the fluid flow of the counterflowing jets and the remaining modes were attributed to noise. The dominant DMD modes showed the oscillation of the shock front over the milliseconds of data taken. Additionally, the frequency content of the DMD modes very closely agrees with the POD frequency analysis. Discrepancies between the POD and DMD results can likely be attributed to the relative sensitivity to noise of the DMD algorithm compared to POD.

1 Introduction and Overview

Supersonic retropropulsion (SRP) is a technology that has gained traction in the last few decades as a method of decelerating spacecraft in order to establish controlled re-entry into an atmosphere. It describes the process of firing rocket engines in the direction of flight during reentry. This technology will bring humanity ever closer toward safely landing crewed spacecraft on Mars, as it can be applied to large payloads that are incapable of landing using current technology. While SRP research has advanced considerably in recent years, there are still many unanswered questions regarding the flowfield dynamics of this method of deceleration. The flow surrounding the shock can become unsteady and a better understanding of the underlying mechanisms of this unsteadiness is needed to mitigate the shock instability. The flowfeild pattern created during SRP (Figure 1) is incredibly complex, requiring detailed analysis to sus out potential instabilities within the dominant features of the flowfeild. Current statistical analysis tools (PCA, POD) provide little detail on how these unsteadiness patters evolve with time. This is why an analysis using Dynamic Mode Decomposition (DMD) is being investigated.

This project will analyze data taken in the supersonic Ludweig Tube at the University of Washington in order to obtain a better understanding of the flowfield dynamics and shock instabilities present in the SRP flow. The scale model used was a 1/4 inch diameter conical nozzle destined for Mach 2 flow. The Ludweig tube produces freestream flows at Mach 2 as well. The gas used for both was compressed nitrogen. A Proper Orthogonal Decomposition (POD) analysis was done on the dataset, as shown in Ho's Thesis on Characterizing Unsteadiness in Supersonic Retropropulsion Flows[2], in order to characterize the modes of unsteadiness present in the shock. The study detailed in this paper conducted DMD analysis of an unsteady SRP flowfeild and compared its results to the previous work done using POD.

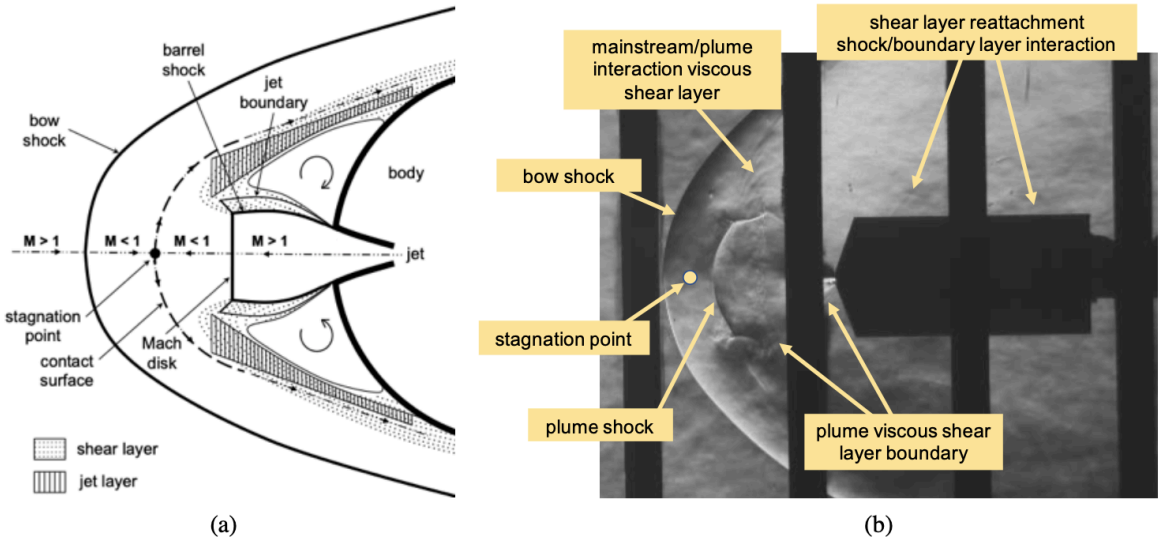


Figure 1: (a) Simplified diagram of SRP flowfeild. (b)Flow field characteristics highlighted in an actual flow field taken from Korzun and Cassel's 2020 paper on scaling SRP dynamics [3].

2 Theoretical Background

Similar to a POD analysis, (DMD) finds the dominant structures present in a a dataset using statistical methods. This is a statistical interpretation of the Singular Value Decomposition (SVD)[4].

2.1 Singular Value Decomposition

Before diving into the background of DMD, the SVD as a mathematical operation must first be investigated. The SVD is a data-driven generalization of the Fourier transform that uses simple and interpret-able linear algebra to build models on. It creates a hierarchical coordinate system based on the data set it is analyzing, which can then be used to study variations in the given dataset. As detailed in [4], the SVD is a matrix decomposition of

the data matrices, $\boldsymbol{X}$ and $\boldsymbol{Y}$, which are defined as a collection of column vectors along the order, for visualization purposes, 100000x1.

$$\boldsymbol{X} = \begin{bmatrix} x_1 & x_2 & x_3 & ... & x_m \end{bmatrix} \tag{1}$$

$$\boldsymbol{Y} = \begin{bmatrix} y_1 & y_2 & y_3 & ... & y_m \end{bmatrix} \tag{2}$$

If, as in the analysis contained in this report, a video of system motion is being analyzed, each snapshot of the video is reshaped into a column vector, where each column of $\boldsymbol{X}$ is the state of the system as it evolves in time. Therefore, x_1 represents the first video frame, x_2 represents the second video frame, and so on up to x_m: each column corresponds to an image. SVD allows us to take $\boldsymbol{X}$ and represent it in terms of three other matrices: $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, $\boldsymbol{V^T}$:

$$\boldsymbol{X} = \boldsymbol{U\Sigma V^T} \tag{3}$$

where $\boldsymbol{U}$ and $\boldsymbol{V^T}$ are unitary, or orthogonal, matrices and $\boldsymbol{\Sigma}$ is a non-negative, hierarchically ordered diagonal matrix, such that:

$$\boldsymbol{X} = \begin{bmatrix} u_1 & u_2 & u_3 & ... & x_m \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & ... & 0 \\ 0 & \sigma_2 & 0 & ... & 0 \\ 0 & 0 & \sigma_3 & ... & 0 \\ 0 & 0 & 0 & ... & \sigma_m \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 & ... & v_m \end{bmatrix} \tag{4}$$

where each column in u and v has the same shape as the columns x in $\boldsymbol{X}$. Each u column is an eigen-image hierarchically arranged in terms of its ability to describe the variance in the columns of x, meaning each u gives you a basis for which you can represent each column of x. Each σ value is a singular value of x corresponding to the singular vector u and singular vector v, representing the amount of energy that each column vector u captures of the image. The v columns are a time series for how its corresponding u evolves in the video: it is the amount of how the mode varies along its path of motion. However, when $\boldsymbol{V}$ is transposed, each column then represents the exact mixture of all the columns in $\boldsymbol{U}$ to equal x_1. σ is ordered in terms of importance, meaning we can ignore the smaller values of σ and approximate the matrix $\boldsymbol{X}$ in terms of the first few dominant columns of $\boldsymbol{U}$ and $\boldsymbol{V^T}$, creating our lower-order representation of our high-dimension data.

2.2 Exact Dynamic Mode Decomposition

Where DMD differs from POD and other techniques that use the SVD is that it's a purely data driven method requiring no governing equations to provide insight on a given data set prior to analysis. This fact makes DMD a versatile tool for data analysis and model creation particularly when dealing with data sets that describe nonlinear systems. At its most basic level, DMD creates a best fit linear operator to describe the behavior of non-linear dynamics within a data set. A requirement of the DMD algorithm is that the data is time resolved. Each 'snapshot' of data must be taken at regular intervals of time. These intervals are defined by $\Delta t = t_m - t_{m-1}$ with the entire time interval starting at t_1 and ending at

t_m[6]. To use DMD, the data matrix from equation 1 must be defined from this time resolved data where each column x_m represents a single snapshot. Two matrices are defined from this original data matrix

$$\boldsymbol{X1} = \begin{bmatrix} x_1 & x_2 & ... & x_{m-1} \end{bmatrix} \tag{5}$$

$$\boldsymbol{X2} = \begin{bmatrix} x_2 & x_3 & ... & x_m \end{bmatrix} \tag{6}$$

where $\boldsymbol{X2}$ represents $\boldsymbol{X1}$ moved into the future. The DMD algorithm seeks to approximate the Koopman operator $\boldsymbol{A}$ (equation 7) that will advance $\boldsymbol{X1}$ to $\boldsymbol{X2}$[6]. The Koopman operator is an infinite dimensional operator that can represent nonlinear infinite-dimensional dynamics without resorting to linearizatrion [4].

$$\boldsymbol{X1A} = \boldsymbol{X2} \tag{7}$$

However, the Koopman operator $\boldsymbol{A}$ is an enormous matrix requiring cost-prohibitive levels of computing power[4]. DMD approximates this operator by finding the eigenvalues and eigenvectors of $\boldsymbol{A}$ without having to calculate it[4]. To accomplish this, DMD makes use of the Singular Value Decomposition (SVD). $\boldsymbol{A}$ can be represented as a lower rank matrix defined by

$$\tilde{\boldsymbol{A}} = \boldsymbol{U^* X_2^M V \Sigma^{-1}}[4] \tag{8}$$

which approximates the eigenvalues λ of the Koopman operator [4]. The eigenvectors corresponding to $\boldsymbol{y}$ can be calculated using the eigenvalue problem shown in equation 5.

$$\tilde{\boldsymbol{A}}\boldsymbol{y} = \lambda \boldsymbol{y}[4] \tag{9}$$

The eigenvectors are then used to reconstruct the DMD modes

$$\boldsymbol{\Phi} = \boldsymbol{U}\boldsymbol{y}[4] \tag{10}$$

with the columns of $\boldsymbol{\Phi}$ representing the low rank approximation of the eigenvectors of the Koopman operator.

The eigenvectors and eigenvalues can also be used to approximate future states of the data in question, but that is beyond the scope of this paper.

3 Algorithm Implementation and Development

Before running the DMD, an SVD is done on the dataset, X, in order to get an understanding of the flowfield structures, u, and the time dynamics corresponding to each structure, v. The SVD also gives insight into which modes are dominant and how many modes will be kept when truncating the data set to rank r for the DMD. Once the SVD is done, the $X1$ and $X2$ matrices must be calculated to represent the change of X through time. The $X1$ matrix is put into an SVD, giving the U, Σ, and V matrices of the $X1$, as described by Equation 4. The line in Algorithm 1 calculating the Ur, Sr, and Vr terms is the rank truncation of the DMD. From the SVD, a number modes are seen to be dominant and therefore r truncates the dataset to the dominant modes. The $\tilde{A}$ term and eigenvalues are found, as shown in Equations 8 and 9. Last, Φ, the DMD modes, are calculated as in Equation 10.

Algorithm 1 SVD and DMD

[u,s,v]=svd(X, 'econ'); 1 = X(:,1:end-1); X2 = X(:,2:end); r=6; [U,S,V] = svd(X1,'econ'); Ur=U(:,1:r); Sr=S(1:r,1:r); Vr=V(:,1:r); Atilde = Ur'*X2*Vr/Sr; [W,D] = eig(Atilde); Phi = X2*Vr/Sr*W;

The DMD frequencies ω are calculated from the eigenvalues λ and used to produce the time dynamics of the modes, onto which the DMD modes Φ are projected to produce the DMD solution. The data of the first image is use to create initial conditions with the pseudo-inverse. The power spectrum of the DMD modes can be produced by plotting the DMD frequencies against an amplitude given by the L2 norm of the corresponding mode α, which is the projection of the original field onto Φ, as described in [5], as seen in 2. α can also be thought of as the coefficients of the dynamical modes. This allows us to view the relative energy of each DMD frequency in the data set.

Algorithm 2 Calculate DMD frequency and associated power spectrum.

lambda = diag(D); omega = log(lambda)/dt; DMDfreq = omega/(2*pi); alpha = X'*Phi; DMDpower = vecnorm(alpha);

4 Computational Results

The hypothesis of the results prior to conducting DMD were that the shock oscillation is the dominant motion represented in the data. This implies simple harmonic motion is the primary dynamic present in the data which can be represented using only two modes. The rank truncation chosen for the DMD algorithm was six. Only six of the modes plotted on the modal energy chart (Figure 2) were used in the DMD calculations with the remaining modes assumed to be representing noise.

4.1 Initial Results

Figure 2 shows the results of the SVD analysis on the dataset. It is apparent from the first graph in the figure that there are a small number of modes that make up a bulk of the energy and the remaining modes are attributed to noise. This supports the previous decision to use only the first 6 modes to reconstruct the images. The dominant three modes are plotted as a function of the frames (equivalent to time) and the oscillatory motion of the shock wave is evident. Next, the projection of the modal data is shown as a function of the frames as well for the same dominant three modes.

Figure 3 shows the mean image of the data before any SVD or DMD is done. If the test conditions were to draw out the long penetration mode of the shock, this would show that information. However, since this particular test case contains a simple harmonic motion of shock oscillation, the image simply shows the shock location at its two most common locations, where the blue shock is the longest standoff distance location and the red shock is the shortest standoff distance location. The shock oscillates between these two extremes.

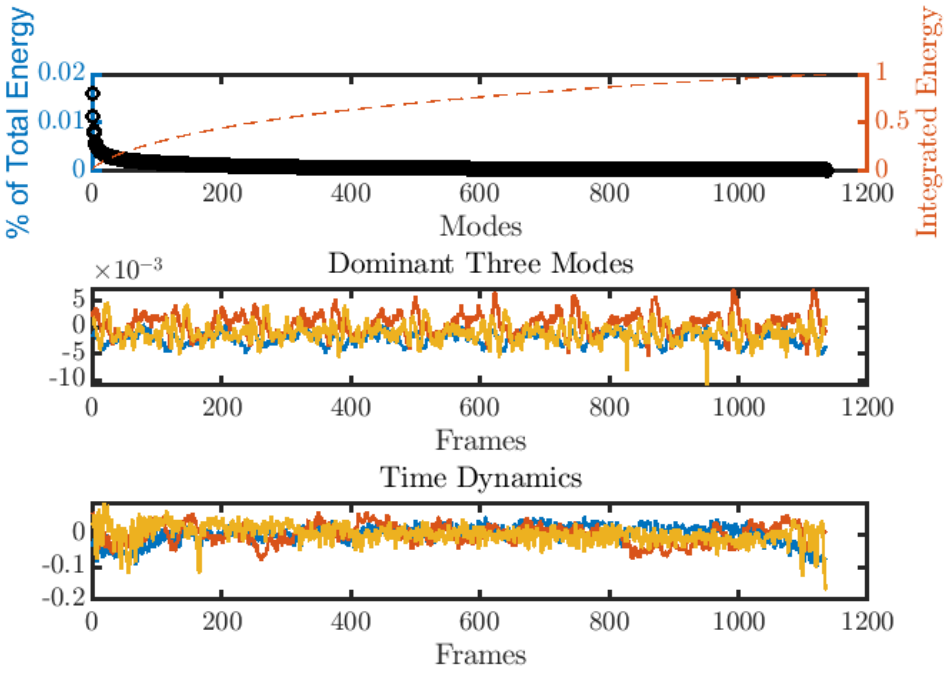


Figure 2: SVD dominant modes and associated time dynamics.

The DMD analysis should replicate this by showing the dominant patterns at the shock extremes of oscillation.

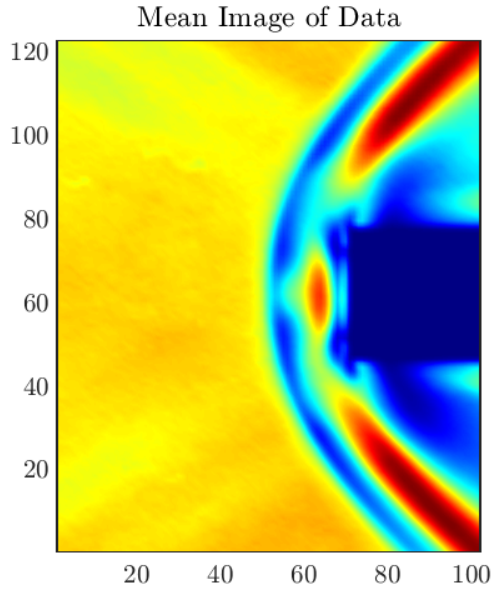


Figure 3: Mean unsteadiness characterized in the original video. This can be used as a baseline to asses DMD accuracy.

4.2 DMD Modes

Figure 4 are modes 1 through 6 projected onto the original data set. Through visual confirmation, it was determined that 6 modes best recreated the original images with low-rank reconstruction. Increasing the number of modes beyond this also increased the prevalence of noise. Areas of hotter colors represent regions of high velocity fluctuations and areas of cooler colors represent regions of low velocity fluctuations.

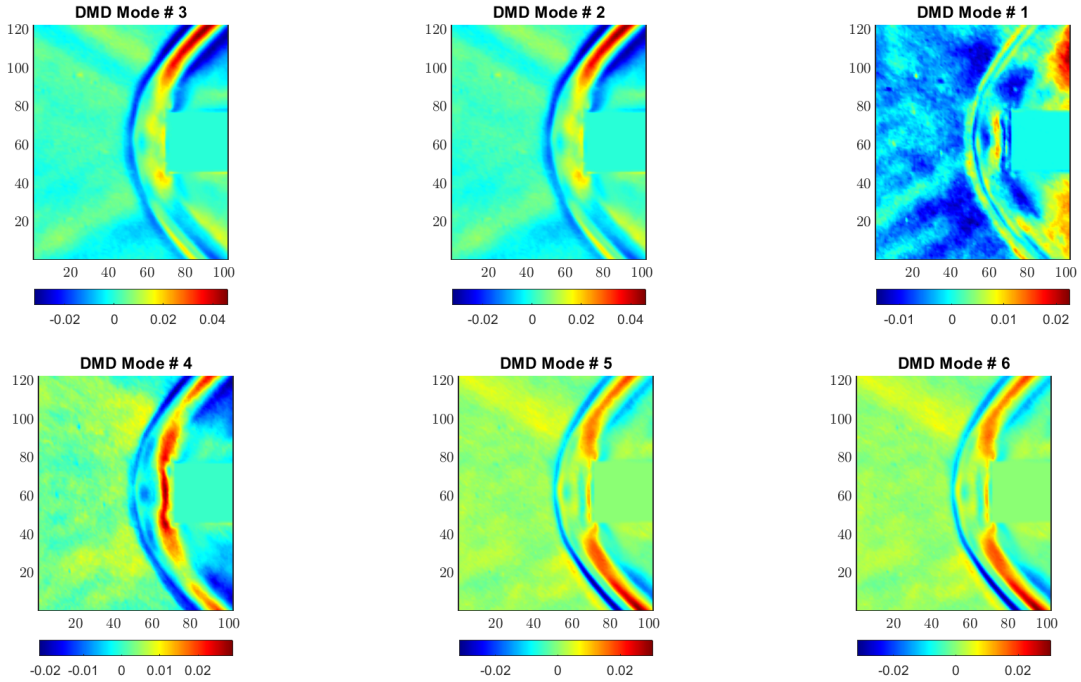
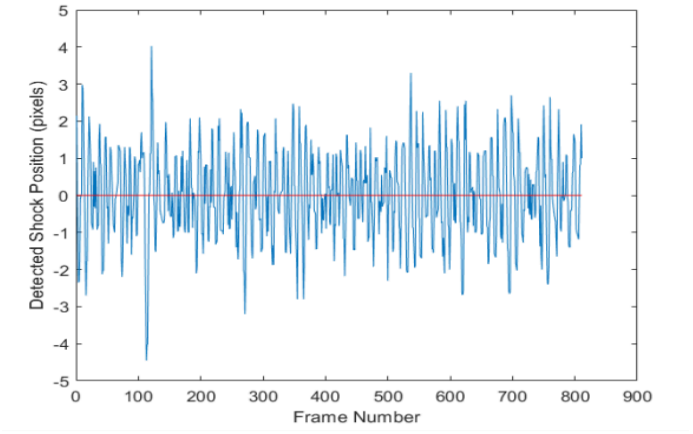


Figure 4: Modes 1-6 projected onto original data set. Hotter colors represent regions of high velocity fluctuations and cooler colors represent regions of low velocity fluctuations.

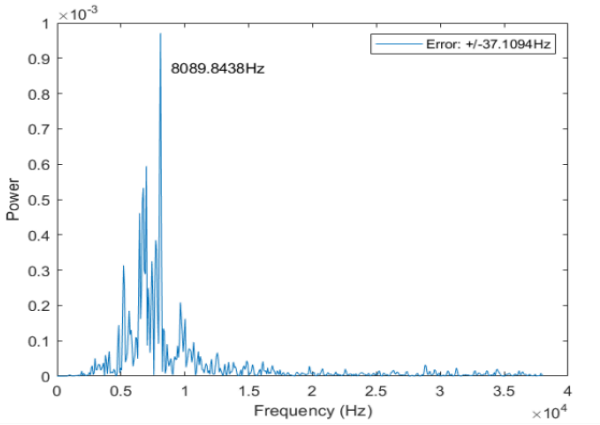
4.3 Frequency

With visualizations of the dominant modes constructed, relative frequency and power plots were created. Frequency values for POD analysis were found by tracking the shockwave front using a script developed by Ho [2]. The shockwave oscillates back and fourth when the flow is unsteady with additional flow features oscillating as well. The shock front position was plotted with time, and then that data was converted to frequency content with the dominant frequency of oscillation being 8.1 kHz[2].

These data sets provide a good baseline to test the DMD analysis against. The expectation of DMD is that the first mode, should represent some shockfront motion.



(a) Tracked position of shock front



(b) Dominant frequencies within shock motion

Figure 5: Frequency analysis of shock motion unsteadiness as seen in Ho's Thesis on Characterizing Unsteadiness in Supersonic Retropropulsion Flows, 2019.[2]

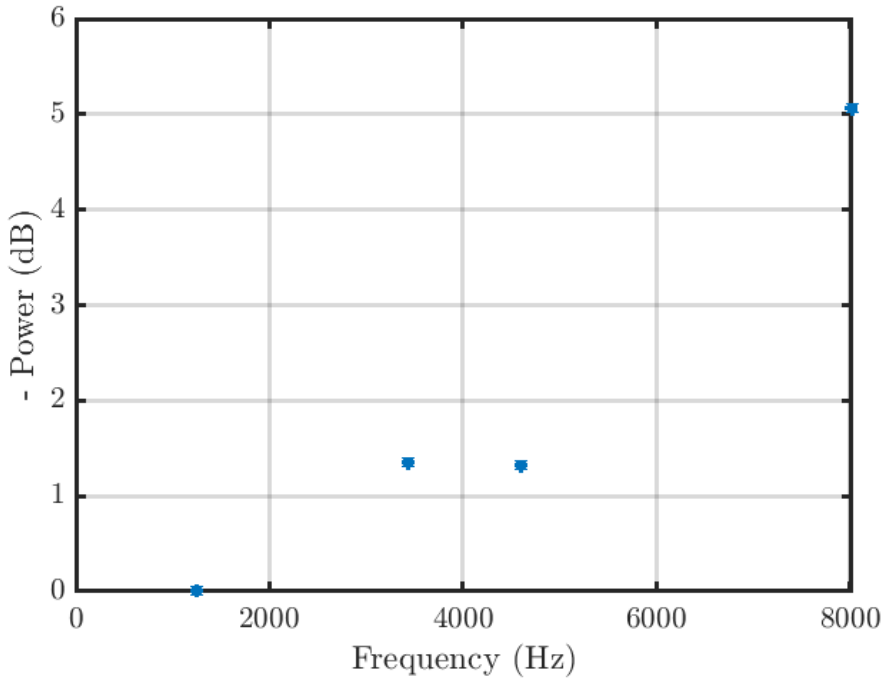


Figure 6: DMD dominant frequencies and associated amplitudes.

The modal truncation of the SVD described earlier in this section created four DMD frequencies to analyze. These frequencies are shown in Figure 6 with the dominant frequency also being around 8 kHz.

5 Summary and Conclusions

Dynamic Mode Decompositions are useful in many types of image and video analysis, ranging from facial recognition to studies of fluid flows. In this study, the fluid flow between counterflowing supersonic jets was examined in order to better understand the instabilites that have been observed in the resulting shock wave. In the dataset used for this paper, an oscillating shock was observed and its movement was characterized by the DMD. The shock front movement was tracked in order to quantify the change in the shock standoff distance and the dominant frequencies of the DMD were found and studied against the power. The DMD showed that only a few modes (around 5-15) could be used in order to characterize the fluid flow of the counterflowing jets and the remaining modes were attributed to noise. The dominant DMD modes showed the oscillation of the shock front over the milliseconds of data taken. Additionally, the frequency content of the DMD modes very closely agrees with the POD frequency analysis. The dominant frequency representing the oscillatory motion of the shock front staying around 8 kHz. The discrepancy between the power (amplitude) values of the two charts is likely due to the sensitivity to noise DMD has relative to POD.

In the future, an optimized DMD analysis will be implemented into the analysis. Because DMD analyses are so affected by noise, the optimized DMD would give a better representation of the data as it accounts for this. Several MATLAB codes implementing optimized DMD on a similar type of experiment were done in Aksham[1] and an analysis utilizing these codes was begun on the dataset, however quite a bit of troubleshooting and editing is required of the codes to be able to apply to this specific experiments. The optimized DMD is expected to be completed in the next few months.

References

[1] Travis Aksham and J Nathan Kutz. "Variable Projection Methods for an Optimized Dynamic Mode Decomposition". In: *2018 Society for Industrial and Applied Mathematics* 17.1 (2018), pp. 380–416.

[2] Brenton Ho. "Characterizing unsteadiness in supersonic retropropulsion flows". MA thesis. University of Washington, 2019.

[3] Ashley M. Korzun and Louis A. Cassel. "Scaling and Similitude in single nozzle supersonic retropropulsion aerodynamics interference". In: *AIAA 2020-0039* (2020).

[4] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[5] F Lusseyran et al. "Flow coherent structures and frequency signature: application of the dynamic modes decomposition to open cavity flow". In: *Journal of Physics: Conference Series*. Vol. 318. 4. IOP Publishing. 2011, p. 042036.

[6] Jonathan H Tu et al. "On dynamic mode decomposition: Theory and applications". In: *arXiv preprint arXiv:1312.0041* (2013).

A MATLAB Functions

- `[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in the vectors `x` and `y`. X is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`. The grid represented by the coordinates `X` and `Y` has `length(y)` rows and `length(x)` columns.
- `y = double(x)` converts the values in x to double precision.
- `y = unit8(x)` converts the values in x to type uint8.
- `y = find(x)` returns a vector containing the linear indices of each nonzero element in array x.
- `[u,s,v]=svd(X,'econ'` produces an economy-size decomposition of m-by-n matrix X. The economy-size decomposition removes extra rows or columns of zeros from the diagonal matrix of singular values, S, along with the columns in either U or V that multiply those zeros in the expression A = U*S*V'.
- `y = diag(x)` returns a square diagonal matrix with the elements of vector x on the main diagonal.
- `y = reshape(x,m,n)` returns the M-by-N matrix whose elements are taken columnwise from X.
- `y = linspace(X1,X1,N)` generates N points between X1 and X2.
- `[V,D] = eig(A)` produces a diagonal matrix D of eigenvalues and a full matrix V whose columns are the corresponding eigenvectors so that A*V = V*D.
- `N = vecnorm(A)` returns the 2-norm or Euclidean norm of A: If A is a vector, then vecnorm returns the norm of the vector. If A is a matrix, then vecnorm returns the norm of each column. If A is a multidimensional array, then vecnorm returns the norm along the first array dimension whose size does not equal 1.

B MATLAB Code

```
%function DMD_SRP
% Author: Lauren Jones, Elliot Jennis, Hari Annamalai
% d
%function DMD_SRP(File,scale,sizeX,Reconstruct,usePercent,
    numModes,2ndProjFile)

function DMD_SRP(varargin)
maxi=1;

set(0,'DefaultFigureColormap',jet())

U = double(S(round(Stab.TipYfit-1.9*NozWidth):round(Stab.TipYfit
    +1.9*NozWidth),indx:round(Stab.TipXfit+NozWidth),:));
Umean = nanmean(U,3);

%implay(framesmt)

%% Find bounds of PCA
figure
clf
pcolor(Umean)
shading interp
axis('equal', 'tight'); title('Mean Image of Data')

%PrepData

[X, Y] = meshgrid(1:size(U,2),1:size(U,1));
Uf = U-repmat(nanmean(U,3),[1,1,size(U,3)]);
Umean = nanmean(U,3);

%% create matrix will all fluctuating velocity components for
    each snapshot in a column
[uSize] = size(Uf);
Uall=reshape(Uf,uSize(1)*uSize(2),uSize(3));

%Do SVD analysis
[u,s,v]=svd(Uall.','econ');
figure
plot(diag(s)/(sum(diag(s))),'ro')
xlabel('Modes')
A = diag(s)/sum(diag(s));
energy_total = zeros(1,length(A));
```

```
energy_total(1) = A(1);
for i=2:length(A)
    energy_total(i) = A(i)+energy_total(i-1);
end
figure
subplot(3,1,1),yyaxis left, plot(diag(s)/sum(diag(s)),'ko','
    Linewidth',[2]), ylabel('% of Total Energy'),...
     yyaxis right, plot(energy_total,'--'), xlabel('Modes'),
        ylabel('Integrated Energy'),...
     ylim([0 1])
subplot(3,1,2), plot(u(1:uSize(3),1:3),'Linewidth',[2]);title('
    Dominant Three Modes');xlabel('Frames'); %modes
subplot(3,1,3), plot(v(1:uSize(3),1:3),'Linewidth',[2]); title('
    Time Dynamics');xlabel('Frames') %time dynamics
%subplot(4,1,2), plot(t,v(:,1)/max(v:,1)),t,v(:,2)/max(v(:,2)),'
    Linewidth',[2])
%subplot(4,1,3), plot(x,u(:,1)/max(u(:,1)),'Linewidth',[2])
%subplot(4,1,4), plot(x,u(:,2)/max(u(:,2)),'Linewidth',[2])

%DMD
t = linspace(0,length(goodImgs)/76000,length(goodImgs));
dt = t(2)-t(1);
X = Uall;
X1 = X(:,1:end-1);
X2 = X(:,2:end);
r=6;
[U,S,V] = svd(X1,'econ');
Ur=U(:,1:r); Sr=S(1:r,1:r); Vr=V(:,1:r);

Atilde = Ur'*X2*Vr/Sr;
[W,D] = eig(Atilde);     %eigen decomp of that matrix
Phi = X2*Vr/Sr*W;    %DMD modes

lambda = diag(D);    %DMD eigenvalues
omega = log(lambda)/dt; %DMD frequencies
%DMD modes are the purple and gold projected on old plots

x1=X(:,1);
b = Phi\x1;  % pseudo-inverse initial conditions
u_modes = zeros(r,length(t));    %time dynamics-r rows, t columns

for iter = 1:length(t)
    %outer product
     u_modes(:,iter) =(b.*exp(omega*t(iter)));
     %for each moment in time, compute this(b*e^iwt)
```

```
end
figure
plot(real(u_modes')),xlabel('Frames'),legend('Mode 1','Mode 2','
    Mode 3','Mode 4','Mode 5','Mode 6')
%xlim([0  50])
X_dmd = Phi*u_modes; %DMD solution
for i = 1:r
figure
clf
%size(Umean)
%size(Phi)
size(X)
pcolor(reshape(real(Phi(:,i)),size(RMSCrop)))
shading interp
colorbar('SouthOutside')
axis('equal', 'tight'); title(['DMD Mode #', num2str(i)])
end
DMDfreq = omega/(2*pi);
alpha = X'*Phi; %projection of u onto phi (from Lusseryan)
DMDpower = vecnorm(alpha);
figure
plot(abs(DMDfreq),-10*log10(DMDpower/max(DMDpower)),'*','
    LineWidth',2);
xlabel('Frequency (Hz)')
ylabel('- Power (dB)')
grid on;

figure
theta = (0:1:100)*2*pi/100;
plot(cos(theta),sin(theta),'k--');
hold on, grid on
scatter(real(lambda),imag(lambda),'ok');
xlabel('Real')
ylabel('Imaginary')
```