Data Analysis Methods for the Study of Plasma Structures in Pulsed Inductive Thrusters

Arvindh Sharma *

March 19, 2020

Abstract

Data analysis techniques were developed to characterize the plasma current sheets in a pulsed inductive thruster based on high-speed images and LC circuit signal data. Test cases involved propellant gas injection using two different schemes: only through the pre-ionizer, and through both the pre-ionizer and coil face. Spoke-like structures in the plasma sheet were observed in all test cases. Fast Fourier Transform (FFT) was used to study and compare the frequency components in the images and LC signals, and showed that an artificial noise from high-speed camera was dominating the frequency space. Singular Value Decomposition (SVD) revealed major modes in the plasma structures. A novel binning method was developed to track the spoke patterns and estimate their rotation speeds. A metric for the uniformity of the plasma sheet was developed and the different test cases were compared using this metric.

1 Introduction and Motivation

Pulsed Plasma Thrusters, or Pulsed Inductive Thrusters (PIT), are a type of electrode-less electric propulsion devices that generate thrust by accelerating a plasma sheet. The plasma sheet is formed inductively in a gas that is injected across a coil face from the current flowing along the filaments in the coil. The coil current also generates a radial magnetic field, and the resulting Lorentz force between the plasma and the magnetic field accelerates the plasma sheet and trapped neutrals away from the coil face. Consequently, formation of a uniform thin plasma sheet is a major requirement for the efficient operation of the thruster [1, 2, 3, 4, 5, 6].

Space Lab at the University of Washington is developing a high-pulse rate PIT. The current in the coil is driven by a voltage fluctuation from a capacitor discharge that oscillates at an LC frequency given by $f_{LC} = 1/(2\pi\sqrt{L_{coil}C_{mb}}$, where L_{coil} is the coil inductance and C_{mb} is the capacitance of the main capacitor bank. We also employ an auxilary pre-ionizer (PI) that ionizes the gas before it is injected through a central aperture; in other cases, the gas is pumped partly through the pre-ionizer and partly across the coil face. Fig. 1 and 2 are montages of images from the first few micro-seconds of operation of the former and latter cases, respectively, acquired with a high-speed camera. It can be seen from the images that the plasma sheet is composed of spoke-like patterns in both cases initially, and the patterns diffuse to form a rather uniform sheet with time. The key-hole like pattern is due to the use of a light-blocker over the central PI aperture. While the spokes are concentrated closer to the central PI in the first case, they extend throughout the coil face in the second case. Characterizing the spoke patterns and determining the uniformity of the current sheet is of major importance in designing the thruster operation.

The focus of this work is to develop data analysis techniques to study the plasma structures from the image data, and correlate their dynamics with the LC signal (voltage and current) that drives the plasma sheet, measured using an oscilloscope. The images in this work were acquired at a frame rate of 5 MHz while the oscilloscope measured the LC signals at 200 MHz. Analysis techniques like FFT, SVD, and signal filtering are extensively used in this work in order to develop an algorithm that can be used to characterize and study the data.

2 Theoretical Background

2.1 Fourier series and FFT

Any periodic function, $f(x)$, can be represented using an infinite sum of sines and cosines, provided $f(x)$ is piecewise smooth. This representation is called a *Fourier series* and is shown in Eq. (1) [7].

*GitHub : `https://github.com/arvindhsharma`

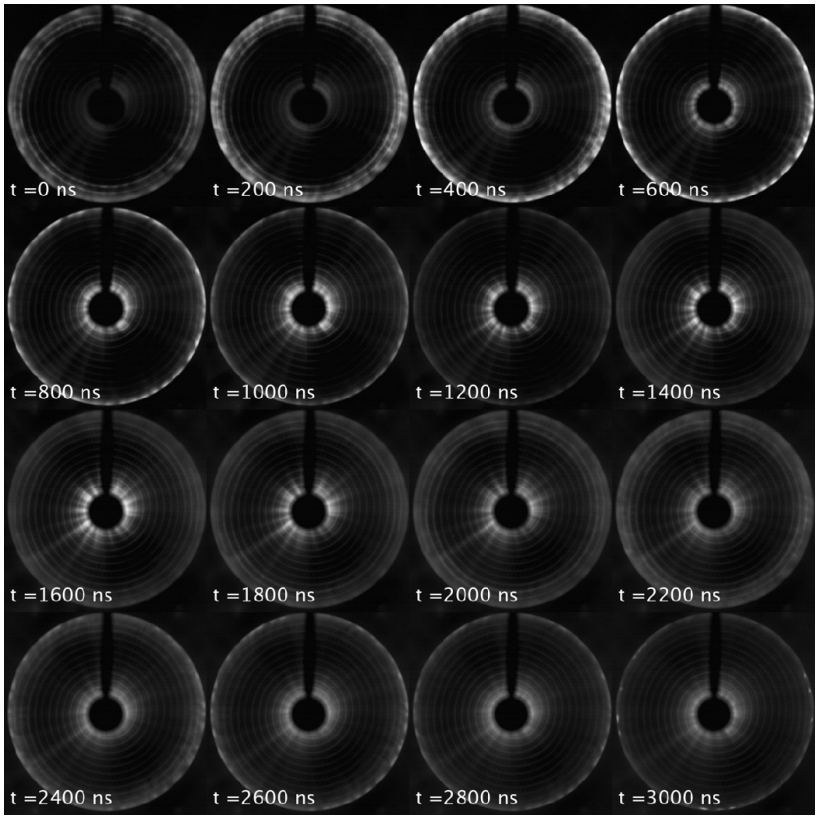

Figure 1: Evolution of plasma structures when propellant is injected only through pre-ionizer. Time, t, is relative to the first image in the montage.

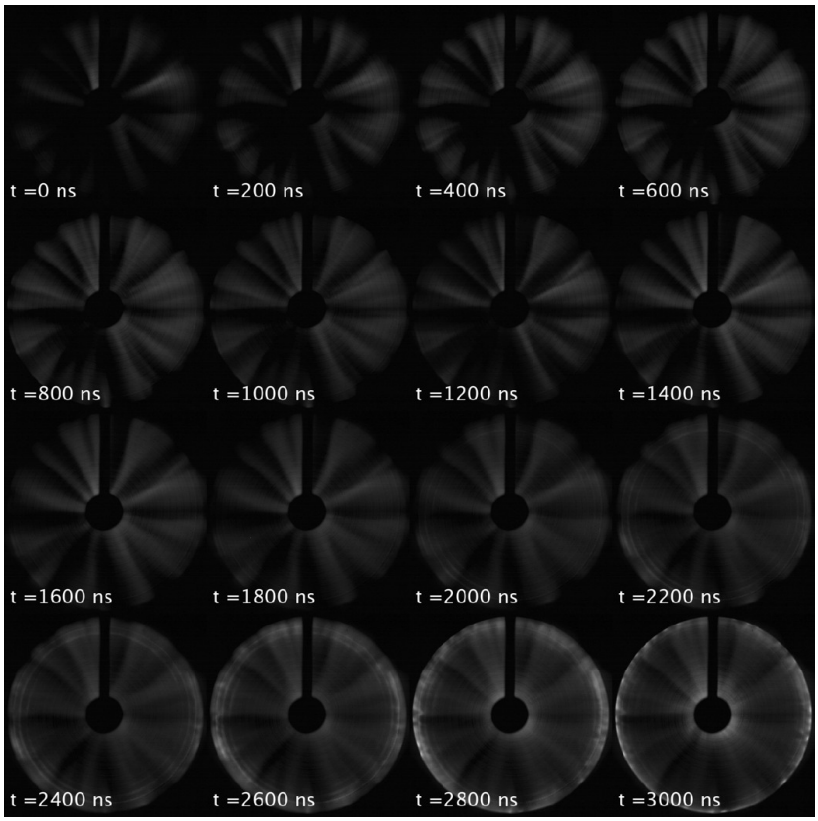

Figure 2: Evolution of plasma structures when propellant is injected through both the pre-ionizer and the coil face. Time, t, is relative to the first image in the montage.

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos kx + b_n \sin kx) \quad x \in (-\pi, \pi] \quad . \tag{1}$$

Since $f(x)$ is periodic, it is expected to repeat itself outside of the domain $(-\pi, \pi]$. It can be recognized that a function defined in an arbitrarily big domain $[-L, L]$ could be scaled to the Fourier domain, when the positional variable x is scaled to $\frac{\pi x}{L}$. The Fourier series for this domain is then given by Eq. (2).

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos \frac{k\pi x}{L} + b_n \sin \frac{k\pi x}{L} \right) \quad x \in [-L, L] \quad . \tag{2}$$

If $L \to \infty$, then the signal $f(x)$ does not necessarily need to be periodic since it is assumed to be periodic over an infinite domain for the purpose of representing it as a Fourier series. At the same time, we can replace the discrete Fourier sum with infinite continuous frequencies and write the series as an integral. This allows us to define the Fourier transform pair given in Eq. (3), which contain the transformations to and back from the Fourier domain for $f(x)$.

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \quad , \tag{3a}$$

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k) dk \quad . \tag{3b}$$

The FFT is an algorithm developed to accurately and quickly evaluate the Fourier transforms (and inverses). For a signal of length N, the number of operations using FFT is $O(N \log N)$. As N grows large, the number of operations grows almost linearly. In order to implement the algorithm, there is a requirement that the length of the input signal be a power of 2. Thus, any signal $f(x)$, periodic or non-periodic, can be transformed to the Fourier domain and back to its original domain (spatial, temporal, etc.) efficiently using FFT as long as its length is a power of 2. This allows us to study the signal in the domain of our choosing where trends can be identified more easily. FFT operates on a domain of length 2π, so if the input signal is of a different domain size, it will have to be scaled to 2π.

Another important concept is that when $f(x)$ is transformed to Fourier domain, the spectral signal corresponding to wave numbers does not change regardless of the order or location of the spectral content in the spatial domain. In other words, a single pulse of given frequency and amplitude in a signal $f(t) \quad t \in [0, 10]$, will have the same Fourier transform regardless of whether the pulse is centered at $t = 1$ or $t = 9$. So even though the signal generated by the marble will have different spatial response due to its changing location, its transform in the frequency domain can be expected to be unchanged.

2.2 Singular Value Decomposition (SVD)

SVD transforms a matrix $\boldsymbol{A}$ into a series of matrix operations: first, a rotation by a unitary matrix $\boldsymbol{U}$, then a stretching action by a diagonal matrix $\boldsymbol{\Sigma}$, and then a rotation by a unitary matrix $\boldsymbol{V^*}$. For $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, the SVD is given by Eq. (4), where $\boldsymbol{U} \in \mathbb{C}^{m \times m}$ (row-space), $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$, and $\boldsymbol{V} \in \mathbb{C}^{n \times n}$ [7].

$$\boldsymbol{A} = \boldsymbol{U\Sigma V}^* \quad . \tag{4}$$

The matrix operations could also be understood to be linear operations — the columns of $\boldsymbol{U}$, which represent the principal modes in the data $\boldsymbol{A}$, are first scaled by the diagonal element in the corresponding columns of $\boldsymbol{\Sigma}$; then the resulting columns are linearly combined with the elements in each column of $\boldsymbol{V^*}$ as coefficients (or weights), thus reproducing the original matrix $\boldsymbol{A}$ [8].

Since the singular values in $\boldsymbol{\Sigma}$ are arranged in a descending order of magnitude, it is easy to reduce a higher-dimensional system to a lower-dimensional representation by picking the first few modes (k) that comprise almost the entire energy contained in the original system. For the matrix $\boldsymbol{A}$ of rank r, Eq. (5) represents the matrix as a sum of its SVD modes, where $k \in [0, r]$, σ_j are the diagonal values of $\boldsymbol{\Sigma}$, $\boldsymbol{u}_j$ are columns of $\boldsymbol{U}$, and $\boldsymbol{v}_j^*$ are columns of $\boldsymbol{V}^*$.

$$\boldsymbol{A}_k = \sum_{j=1}^{k} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^* \quad . \tag{5}$$

The most important property of SVD is that the lower-rank approximations constructed using Eq. (5) are also the *best* approximations, as determined by l^2 norm and Frobenius norm. These norms are given by Eq. (6) and Eq. (7).

$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_2 = \sigma_{k+1} \quad . \tag{6}$$

$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_F = \sqrt{\sigma_{k+1}^2 + \sigma_{k+2}^2 + \cdots + \sigma_r^2} \quad . \tag{7}$$

2.3 Gaussian filter

Gaussian filters are useful in preserving the signal strength in the spectral domain around the frequency of interest, k_0, while reducing the intensity everywhere else. With the variance σ^2 acting as the factor defining the width of the filter, a simple Gaussian with unit maximum amplitude can be constructed as shown in Eq. (8) in 1-dimension.

$$G(k;\sigma) = \exp(-\frac{(k-k_0)^2}{2\sigma^2}) \quad . \tag{8}$$

3 Algorithm Implementation and Development

Algorithm 1 provides an overview of the analysis methods used to study each test case. Here is a brief description of the process.

3.1 Data setup

The image data and the voltage and current signals from the test case under study are loaded into the workspace. Since the images are acquired at a slower rate (5 MHz) compared to the voltage signals (200 MHz), the voltage and current signals are resampled at times corresponding to the image acquisition times. Then the images, which are in RGB format, are converted to grayscale and cropped so that the area outside of the coil face is removed.

3.2 Frequency analysis

In order to understand the dominant frequencies at which the structures in the image evolve, we sum up the pixel intensities in each image and store it in a vector called *intensity*. This allows us to perform FFT on the overall intensity changes in the images and compare them to the FFT of the current signal that drives the plasma.

3.3 SVD analysis

Noting that the images display spoke patterns that change in time, SVD analysis holds the potential to reveal dominant modes in the patterns and their variation in time. The pixel intensities from each image is reshaped into a column vector and a matrix is built with such vectors corresponding to images from roughly 1 LC cycle during which the spokes form. SVD is performed on the resulting matrix, and the dominant modes and their time-evolution are visualized.

3.4 Binning analysis

The tracking of the spoke patterns and estimating their speeds requires novel computation techniques. A "binning" method is developed, where the azimuthal direction around the center of the coil face is converted into overlapping sectors, or "bins."

The first step in the process is to identify the center of the coil. An image with good contrast is obtained by averaging some of the brightest images in the image stack. Next, a circle is fitted around the inner edge of the coil face (around the light block), which is a circle with a notch. The center of the fitted circle is taken to be the center of the coil, and the radius of the circle is taken to be the *inner_radius*. Similarly, a larger circle is fitted around the outer edge of the coil face, with its center restricted to be very close to the center of the inner circle. The radius of the larger circle is then designated *outer_radius*. The result of this process is shown in Fig. 9 in Appendix C.

The rectangular pixel co-ordinates (x, y) can be transformed to polar co-ordinates (r, θ), with the center of the image as origin, as shown in Fig. 10 and 11 in Appendix C. This is useful in representing the 360° around the center as 600 overlapping bins, with each bin having a width of 1.2°, and overlaps with half of the adjacent bin on either

Algorithm 1 Algorithm to analyze image and signal data.

Load images and voltage/current signal data
Resample voltage/current signals at image acquisition times
Convert RGB images to grayscale and crop them to area of interest
procedure FREQUENCY ANALYSIS(images)
    for $n = 1:1:length(images)$ **do**
        $intensity(n) \leftarrow$ Sum of all pixel intensities in n^{th} image
    end for
    Analyze the FFTs of *intensity* and current signals
end procedure
procedure SVD ANALYSIS(Images from 1 LC cycle)
    $Image\_Composite \leftarrow$ Image pixel data arranged as column vectors
    Perform SVD on $Image\_Composite$ and analyze the modes
end procedure
procedure BINNING ANALYSIS(images)
    $bright\_avg\_img \leftarrow$ Average of brightest images
    Fit a circle to the edge around the aperture at the center of coil face
    $[image\_center, inner\_radius \leftarrow$ [Center of the circle, radius of the circle]
    $outer\_radius \leftarrow$ Radius of circle fit around the outer edge of coil face
    Convert rectangular pixel co-ordinates (x, y) to polar co-ordinates (r, θ) with $image\_center$ as origin
    Divide the 360° around the origin into 600 sectors (bins) with overlap
    Normalize the images to have same sum of intensities of pixels in $r \in [inner\_radius, 0.4 * outer\_radius]$
    $BinSums \leftarrow$ Sum of pixels in each bin in the normalized images
    $BinSums\_tr \leftarrow$ Truncate BinSum vectors to remove the bins corresponding to central light blocker
    $BinSums\_tr\_filt \leftarrow$ Apply a Gaussian filter to $BinSums\_tr$ and filter out high-frequency noise
    Calculate spoke speeds by tracking the peaks in $BinSums\_tr\_filt$ across images
    $uniformity \leftarrow$ Power in the DC component of $BinSums\_tr$ normalized by the total power in $BinSums\_tr$
end procedure

side. To normalize all the images, the sum of the pixels in the region from the inner circle to 40% of the outer radius from the center are calculated for all images, and the mean is determined. The pixel intensities of the images are then scaled so that this sum in each image is the same as the mean. The $BinSums$ are calculated by summing up the pixels that fall within each of the 600 bins, for each of the 180 images, thus forming a matrix of size 600×180. Since the bins covering the light block in the image do not vary much from image to image, those bins are removed from the $BinSums$ to form a truncated matrix $BinSums\_tr$. Fig. 12 in Appendix C shows the first bin (27) and last bin (574) used to construct the truncated matrix. Finally, the high-frequency noise in $BinSums\_tr$ is filtered out using a Gaussian filter centered around the 0 wave number. The filtered signal is called $BinSums\_tr\_filt$.

The spokes can be tracked by identifying the maxima in the $BinSums\_tr\_filt$ signal corresponding to each image. By measuring the bins, and thus the angle, across which the maxima corresponding to the spokes move in successive image frames, the rotation speed of the spokes can be estimated. In order to measure the uniformity of the current sheet, the power around the DC component of the FFT of $BinSums\_tr\_filt$ is normalized by the total band power in the FFT, for each image. If the current sheet is uniform, $BinSums\_tr\_filt$ would not have high-frequency components corresponding to the rise and fall of the signal when spokes are present. Thus, a higher concentration of the power in the DC component of the FFT indicates a more uniform current sheet, and this metric is used to track how the uniformity changes during the operation of the thruster.

4 Computational Results

Four different cases are analyzed using the code given in Appendix B, and the test conditions are listing in Table 1. The major difference to note is that Case 1 and 3 have gas injection only through the central PI, while the other cases have most of the gas injection through the coil face in addition to the injection through the PI.

Fig. 3 shows the results of the FFT on image and current signal data for Case 1 and 2. It can be seen that the LC oscillation that drives the plasma formation occurs at 108 kHz, as seen in the current signal data in Fig. 3(a) and (c). The signal pertaining to the sum of the intensities, however, is dominated by fluctuations at 500 kHz and higher harmonics (Fig. 3(b) and (d)). This is possibly an artifact of the high-speed camera, as suggested by the fact that the

Case	Chamber pressure [mtorr]	PI pressure [mTorr]	Coil pressure [mTorr]	Capacitance [μF]	Voltage [V]
1	3	3	0	3	1
2	3	1	2	3	1
3	2	2	0	2.5	1
4	3	1	2	2	1

Table 1: Test cases and operating conditions.

dominant frequency of 500 kHz is exactly $\frac{1}{10}^{th}$ of the image acquisition rate of 5 MHz. Thus, the FFT analysis does not reveal any frequency domain correlation between the plasma patterns and the driving LC signal, but enables us to recognize the large artifact introduced by the camera. Attempts to filter out the noise from the intensity signals did not provide benefit in identifying new patterns while morphing the data to a large degree, and so those efforts were not pursued further in this work.

The next step in the analysis process was performing SVD on the images corresponding to about 1 LC cycle during which the spoke patterns were present. The image range of interest was manually selected for each case. The first 6 dominant modes for Case 2 are shown in Fig. 4, and their corresponding time evolution, given by the columns of $\boldsymbol{V}$ from the SVD, are plotted in Fig. 5. It can be seen that the modes attempt to capture the prominent features from the spoke patterns, with Mode 1 showing the spokes near the center. However, the time dependency of the Mode 1 mirrors the change in overall intensity previously seen in Fig. 3. It can also be seen that all the mode signals seem to be dominated by fluctuations every 10 frames, or at 500 kHz, which is the previously identified noise from the camera.

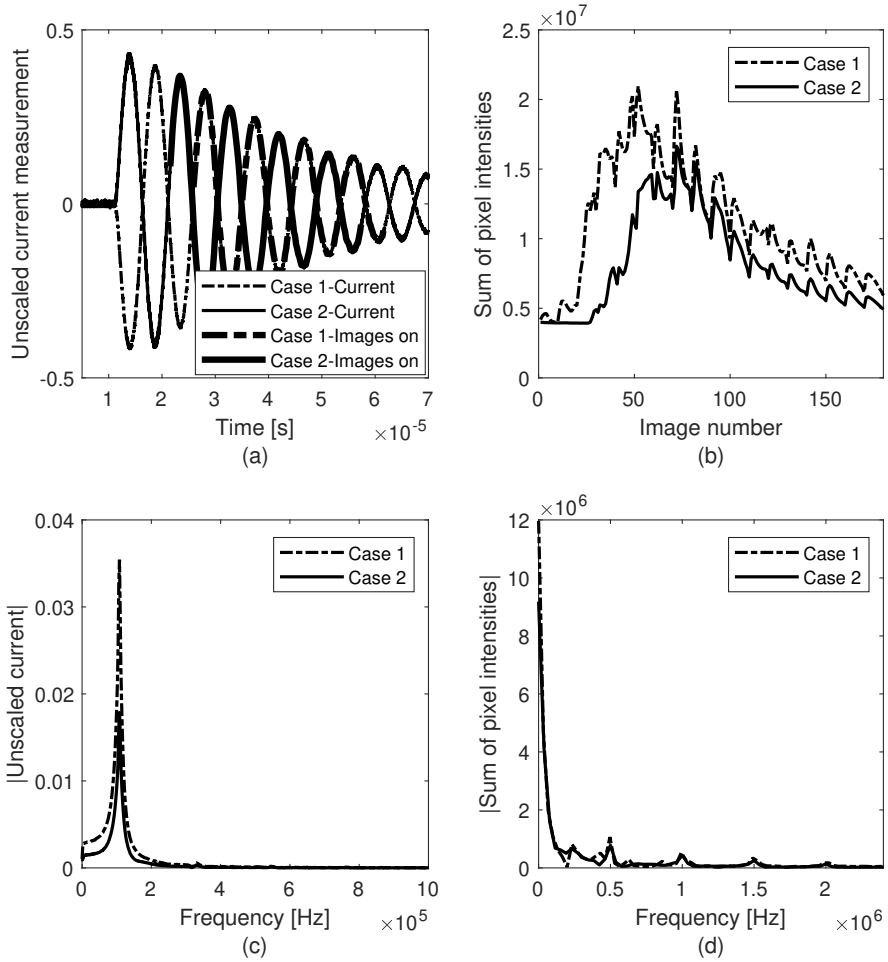

Figure 3: For Case 1 and Case 2: (a) Time trace of coil current, with heavier line-weights indicating when the image acquisition is active; (b) The total sum of intensities in the images; (c) FFT of the coil currents; (d) FFT of the sum of intensities in the images.

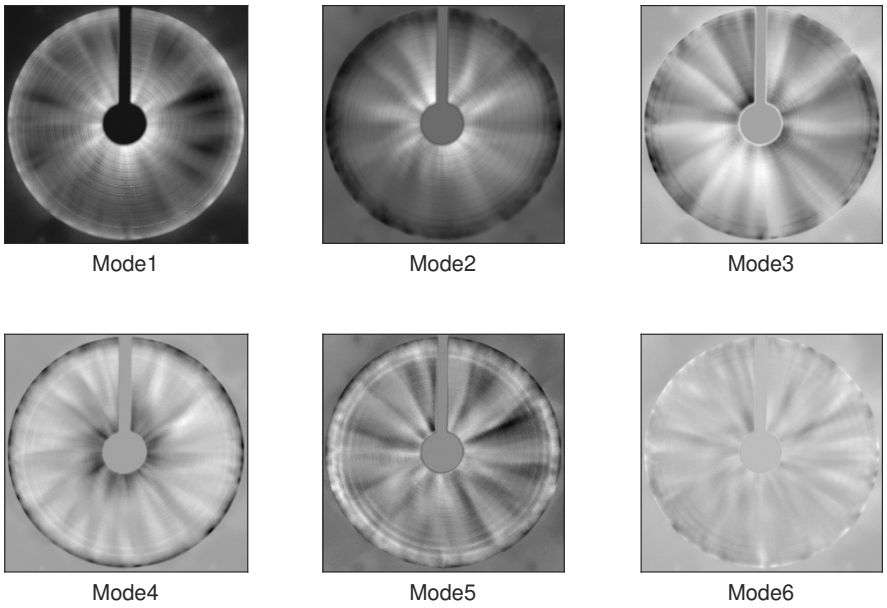

Figure 4: First 6 modes of SVD for Case 2.

Added to this complexity in modeling the spokes is the fact that the SVD suffers when trying to capture rotating structures and traveling waves [7]. Thus, the SVD does not help provide additional information as to the movement of the spokes, and we have to rely on other techniques.

The binning analysis is well suited in this case to track the spoke patterns. Fig. 6 shows the process by which spokes can be identified in two different images by locating the maxima in $Binsums\_tr\_filt$ signals. By measuring the number of bins across which the maxima corresponding to the spokes move from one image to the next, the angular movements of the spokes can be approximated. This helps us in calculating angular speed of the spokes in narrow ranges of images when the spokes are prominent. It was seen that the spokes rotate at speeds varying between 8 kHZ and 50 kHz in the different cases under study. It was also noticed that the spokes seem to reverse the direction of rotation while retaining their speed. More work is planned in the future to correlate the spoke movements with underlying plasma physics that drive the phenomenon. Fig. 7(a) and (b) show how four major spoke patterns for Case 2 are tracked between two images, and Fig. 7(d) and (d) show that the corresponding images are captured when the current signal approaches a local maximum.

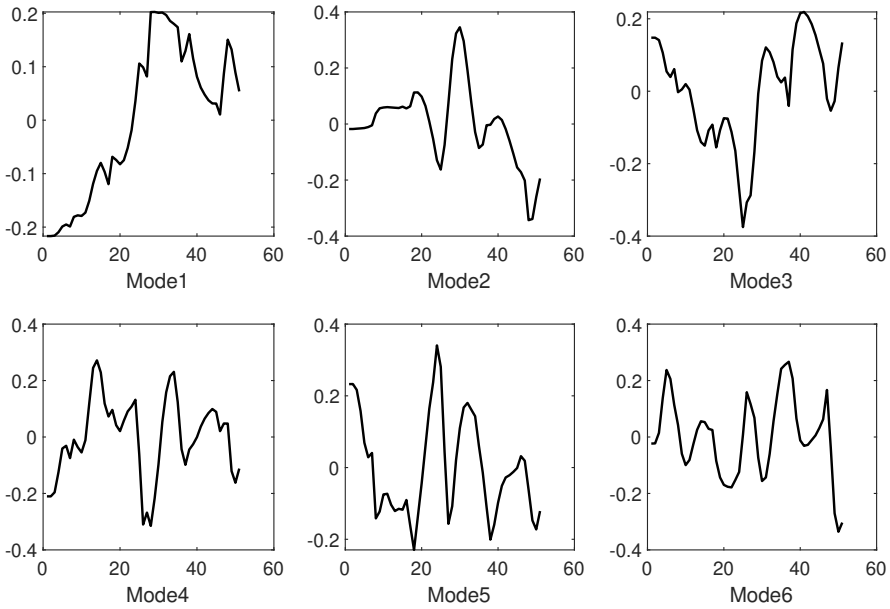

Figure 5: Time variation of the first 6 modes of SVD for Case 2.

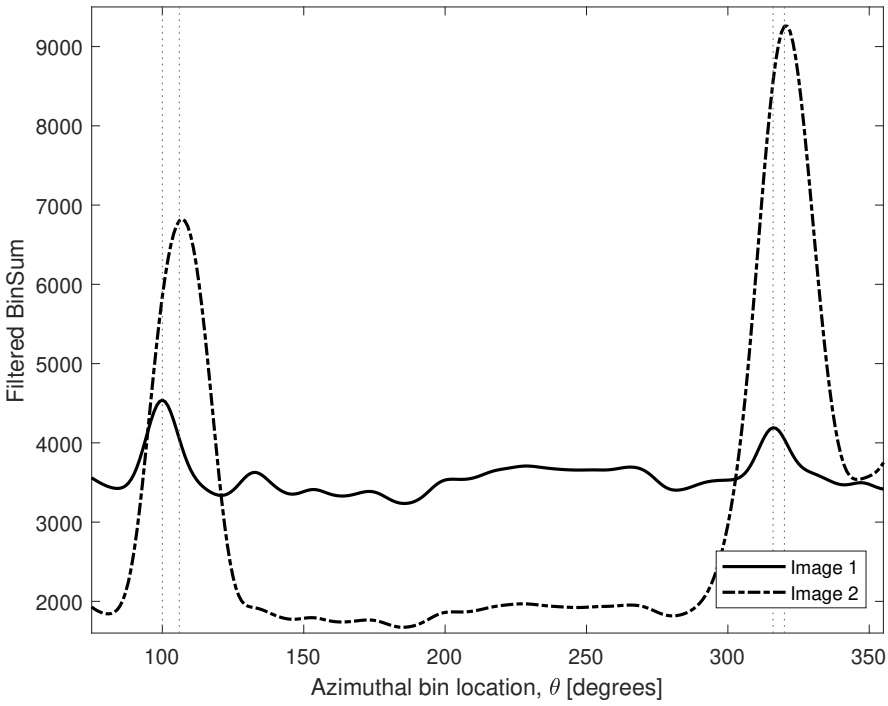

Figure 6: Plot showing the azimuthal movement of the spokes (indicated by the peak locations), from $BinSums\_tr\_filt$ traces of two images acquired 600 ns apart for Case 2.

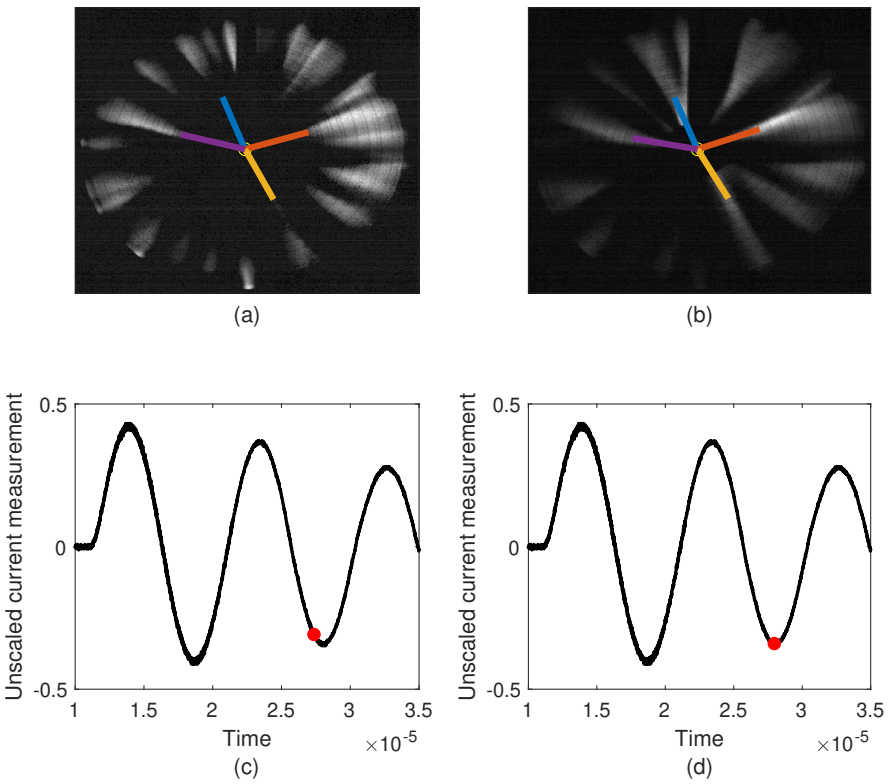

Figure 7: For case 2: (a) Frame showing the location of prominent spokes; (b) Frame showing the changed location of the tracked spokes in the image taken after 600 ns; (c) Coil current time trace with the red dot corresponding to the time when image in (a) was acquired; (d) Coil current time trace with the red dot corresponding to the time when image in (b) was acquired.

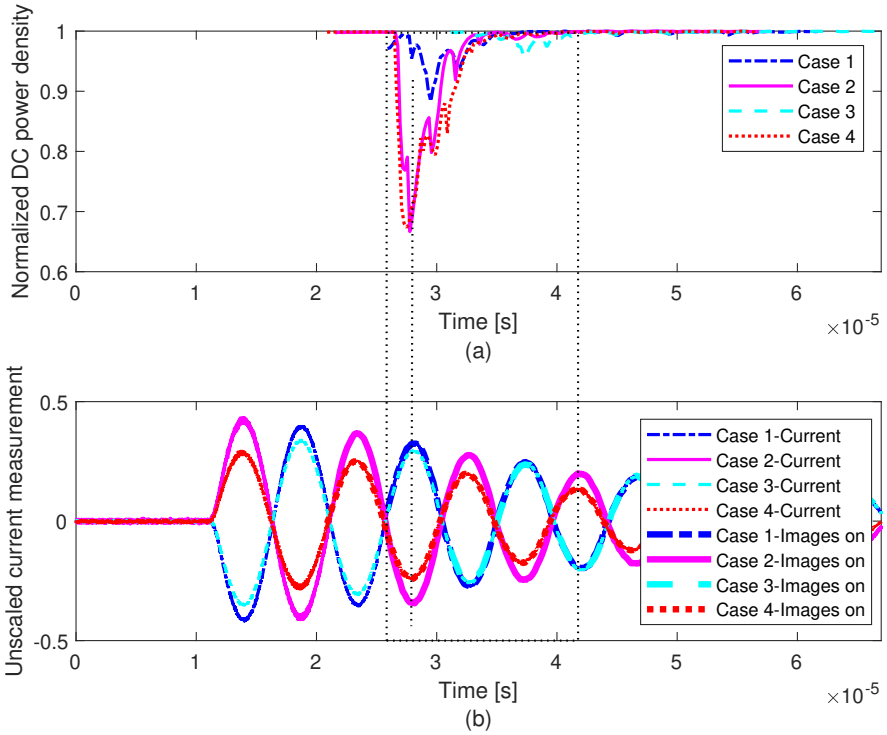

Figure 8: For Case 1—4: (a) Plot of the power density of $BinSums\_tr$ at zero frequency normalized by the maximum power density at 0 frequency in all the images; (b) Time trace of coil current, with heavier line-weights indicating when the image acquisition is active.

Fig. 8(a) shows how uniform the current sheet is in the first few LC cycles after voltage discharge based on image analysis of the four cases under study. Fig. 8(b) helps visualize the corresponding direction of the current signal, before and during the image acquisition. It was observed by correlating the uniformity measurement with the images that the first dip, after approximately 1.5 LC oscillations when the current starts increasing, corresponds to when the visible plasma formation begins. This is when the spoke-like structures are visible, and the uniformity drops to a minimum in all four cases. Subsequently, the transient spoke patterns disappear and the plasma sheet acquires a uniform sheet-like appearance, and this corresponds to the uniformity measure in Fig. 8(a) approaching a value of 1. The time it takes for the different cases to form a uniform sheet appear to be roughly equal to 1.75 LC cycles from when the plasma starts forming, and about 3.25 LC cycles from when the capacitor initially discharges. The minima in the uniformity metric appears to correspond to the point when large scale plasma structures form, and this aligns with when the current approaches a local maxima. It also shows that for Case 1 and 3, the uniformity measure is much higher throughout compared to Case 2 and 4, even though the spoke-like patterns are more prominently observed in the former. This could be due to the fact that the spoke patterns are more intense around the central aperture in Case 1 and 3, whereas the entire sheet seems to move and form structures in Case 2 and 4. These measures help in identifying operating regimes in which the thruster can achieve good efficiency, since the formation of a uniform current sheet is one of the most important considerations in thruster design.

5 Summary and Conclusions

This work explored and developed data analysis techniques to characterize the operation of pulsed plasma thrusters based on image and signal analysis. Understanding the cause and dynamics of spoke-like structures in the plasma sheet, and measuring the uniformity of the sheet, are important considerations in the thruster design. FFT of the image and signal data showed that the image data has a large noise component that is likely fed by the high-speed camera. SVD analysis of the images further confirmed the presence of significant noise, while also helping to visualize the spoke-like modes. However, since the spokes appear to rotate, the SVD does not completely reveal the time-dependent behavior of the rotating patterns. A novel binning technique was developed to track the spokes, and the spokes are estimated to rotate at speeds between 8 kHz and 50 kHz. A metric for the uniformity of the current sheet was calculated from the FFT of signals from the binning method, and its analysis reveals that the spoke patterns

transiently form and morph into uniform current sheets within 1 to 1.75 LC cycles, and the plasma starts forming after the first 1.5 LC cycles in all the cases under study. The analysis techniques offer the potential to study new datasets from different test cases, and thus help understand the plasma physics driving the transient thruster operation.

References

[1] C. L. Dailey and R. H. Lovberg. *The PIT MkV pulsed inductive thruster*. Tech. rep. NASA Technical Report, 1993.

[2] K. A. Polzin. "Faraday accelerator with radio-frequency assisted discharge (FARAD)". PhD thesis. Princeton University, 2006.

[3] A. K. Martin et al. *Design and Testing of a Small Inductive Pulsed Plasma Thruster*. Tech. rep. Tech. rep., 2015.

[4] T. E. Weber. "The electrodeless Lorentz force thruster experiment". PhD thesis. University of Washington, 2010.

[5] Kurt Polzin. "Comprehensive Review of Planar Pulsed Inductive Plasma Thruster Research and Technology". In: *Journal of Propulsion and Power* 27 (May 2011), pp. 513–531. DOI: 10.2514/1.54140.

[6] Kurt Polzin and Edgar Choueiri. "Performance Optimization Criteria for Pulsed Inductive Plasma Acceleration". In: *Plasma Science, IEEE Transactions on* 34 (July 2006), pp. 945–953. DOI: 10.1109/TPS.2006.875732.

[7] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. DOI: 10.1017/9781108380690.

[8] Robert A. Beezer. *A First Course in Linear Algebra*. URL: http://linear.ups.edu/html/fcla.html. (accessed: 03/08/2020).

Appendix A MATLAB Functions

Here is an overview of the major functions used in the programs in Appendix B.

- `Y = fft(X)` returns the FFT of the vector `X`, or if `X` is a matrix, the FFTs of the columns of `X`. `Y` is of the same size as `X`.
- `Y = ifft(X)` returns the discrete inverse Fourier transform of the vector `X`, or if `X` is a matrix, then the discrete inverse Fourier transform of the columns of `X`.
- `Y = fftshift(X)` rearranges Fourier transform `X` so that the zero frequency component is at the center of the array. `X` can be a vector, a matrix, or higher-dimensional array.
- `X = ifftshift(Y)` performs the inverse of `fftshift()` and rearranges `Y` so that the frequency components are in the same order as the original FFT output.
- `[U, S, V] = svd(A)` returns the singular value decomposition of the matrix `A`, such that `A = U * S * V'`.
- `B = reshape(A, sz)` reshapes the matrix `A` to the size specified by the size vector `sz`, and outputs the result in `B`.
- `y = resample(x, p, q)` resamples the data in column vector `x` (or columns of `x` if it is a matrix) at a new rate `(p / q)` times the original sampling rate. It also applies an anti-aliasing FIR lowpass filter and compensates for the time delay introduced by the filtering process.
- `[centers, radii] = imfindcircles(A, radius, options)` returns a list of potential circles (or arcs) of radius '`radius`' that are present in the image `A`. Additional options can be specified to modify the sensitivity of the detection, and to indicate whether the circles of interest are darker or lighter compared to the background.
- `I = rgb2gray(RGB)` returns a grayscale equivalent of the color image `RGB`, where the equivalence is governed only by the luminance of pixels.

Appendix B MATLAB Code

```
clear
close all

%% Adding file path

addpath('D:\research_UW\PIT\AVI_Files\')
addpath(genpath('D:\research_UW\lib'))

%% Getting info about the file from log

prompt = '\n Enter filename: ';
filename = input(prompt, 's');
% filename = 'pit_0451';
temp = strsplit(filename, '_');
case_id = temp{2};
log_data = readtable('PIT_log.csv');
scopedata_filename = log_data.Shot_(strcmp(filename, log_data.KiranaFile));
scopedata_filename = scopedata_filename{:};
[~, case_idx] = whosi(log_data.KiranaFile, case_id);
Fs_img = log_data.KiranaFrameRate_fps_(case_idx);

%% Opening AVI and log files

[frames, num_of_frames, frame_size] = readavi(strcat(filename, '.avi'));
DPO_data = load(strcat('D:\research_UW\PIT\Scope_data\DPO_5054\mat\setup_test_',...
    case_id));
TDS_data = load(strcat('D:\research_UW\PIT\Scope_data\TDS_3034B\TDS_3034Bmat\setup_test_',...
    case_id));

%% Resampling data at each image time point

Fs_TDS = 1 / mode(diff(TDS_data.time));
% timeDelta_btw_img_in_TDS = Fs_TDS / Fs_img;
mod_TDS_timing_signal = TDS_data.CH1 - 1.5; % Because 1.5 V is where an image is triggered
% Next step: Use the diff function to find the points that are "rising",
% and use logical AND with a product of adjacent points. The product of
% adjacent points would be negative in this case.
img_pts_filt = find((diff(TDS_data.CH1) > 0) & ((mod_TDS_timing_signal(1:end - 1) .*...
    mod_TDS_timing_signal(2:end)) < 0));
if length(img_pts_filt) ~= 181
    error('Check the trigger signal.')
end
img_pts_TDS = img_pts_filt(1:180) + 1;

oldTDSfieldnames = fieldnames(TDS_data);
newTDSfieldnames = cellfun(@(x) strcat('img_', x), oldTDSfieldnames, 'UniformOutput', 0);
for ii = 1: 1: length(oldTDSfieldnames)
    TDS_data.(newTDSfieldnames{ii}) = TDS_data.(oldTDSfieldnames{ii})(img_pts_TDS);
end

oldDPOfieldnames = fieldnames(DPO_data);
newDPOfielnames = cellfun(@(x) strcat('img_', x), oldDPOfieldnames, 'UniformOutput', 0);

[~, ~, img_pts_DPO] = intersect(TDS_data.img_time, DPO_data.time);
for ii = 1: 1: length(oldDPOfieldnames)
```

```
    DPO_data.(newDPOfielnames{ii}) = DPO_data.(oldDPOfieldnames{ii})(img_pts_DPO);
end

Fs_DPO = 1 / mode(diff(DPO_data.time));

%% Convert to grayscale

for ii = 1:1:num_of_frames
    frames(ii).gray_pxdata = rgb2gray(frames(ii).pxdata);
    % Cropping
    frames(ii).cr_gray_pxdata = imcrop(frames(ii).gray_pxdata, [183, 116, 553, 553]);
end

%% Overall Intensity Analysis and Signal Analysis

intensity = nan(num_of_frames, 1);

for ii = 1:1:num_of_frames
    intensity(ii, 1) = sum(sum(frames(ii).gray_pxdata));
end

% Finding peaks

[pks1, locs1] = findpeaks(intensity, 'MinPeakHeight', 0.75 * max(intensity));
[pks2, locs2] = findpeaks(-intensity(1:locs1(1)));
start_of_int_phnmn = locs2(end);

% FFT

% Limages = num_of_frames - start_of_int_phnmn + 1;
Limages = num_of_frames;
% Fs_img = 5000000;
% Y = fft(intensity(start_of_int_phnmn:end));
Y = fft(intensity);
Timages = (0: 1 / Fs_img: (Limages - 1) / Fs_img)';
Freqimages = ((0: Limages-1) * (Fs_img / Limages))';
ImageFFT = abs(Y) .* (2 / Limages);
ImageFFT(1, 1) = ImageFFT(1, 1) / 2;

figure, plot(intensity)
hold on
plot(start_of_int_phnmn, -pks2(end), 'ro')

figure, plot(Freqimages, ImageFFT)

L_sig = length(DPO_data.time);
Y = fft(DPO_data.CH4);
freq_sig = ((0: L_sig - 1) * (Fs_DPO / L_sig));
SigFFT = abs(Y) .* (2 / L_sig);
SigFFT(1, 1) = SigFFT(1, 1) / 2;

figure
subplot(2, 1, 1)
plot(DPO_data.time, DPO_data.CH4)
subplot(2, 1, 2)
plot(freq_sig, SigFFT)
xlim([0 1e6])
```

```
%% SVD/PCA on images

Image_composite = nan(554 * 554, length([25:50]));
c1 = 0;
for frame_num = 25:75
    c1 = c1 + 1;
    frame = frames(frame_num).cr_gray_pxdata;
    Image_composite(:, c1) = frame(:);
end

avg_img = mean(Image_composite, 2);
Image_composite_pca = Image_composite - avg_img * ones(1, length([25:50]));
[U, S, V] = svd(Image_composite_pca, 'econ');

figure, imagesc(reshape(avg_img, 554, 554))
figure;
for ii = 1: 1: 6
    subplot(2, 3, ii)
    imagesc(reshape(U(:, ii), 554, 554))
end
colormap(gray)

%% Getting a bright image for finding boundaries

cr_frame_size = size(frames(1).cr_gray_pxdata);
bright_sum_img = zeros(cr_frame_size);
for jj = locs1'
    bright_sum_img = bright_sum_img + double(frames(jj).cr_gray_pxdata);
end
bright_avg_img = uint8(bright_sum_img / length(locs1));

%% Finding center of everything

% Assumption - The center is the same for all images in a video

fig = figure;
imshow(bright_avg_img)
I = bright_avg_img;
[centers_i, radii_i, metric_i] = imfindcircles(I, [40, 60], 'ObjectPolarity',...
    'dark', 'Sensitivity', 0.95);
if length(radii_i) == 1
    inner_radius = radii_i;
    image_center = centers_i;
    hold on
    plot(image_center(1), image_center(2), 'r*')
    viscircles(image_center, inner_radius, 'Color', 'm');
else
    error('Inner circle detection filed. Troubleshoot manually!')
end
[centers_o, radii_o, metric_o] = imfindcircles(I, [250, 280], 'ObjectPolarity',...
    'bright', 'Sensitivity', 0.98);
if length(radii_o) == 1
    outer_radius = radii_o;
    viscircles(image_center, outer_radius, 'Color', 'b');
else
    error('Outer circle detection filed. Troubleshoot manually!')
```

```
end

% Find if the edge detection is acceptable
figure(fig)
fprintf('\nDistance between centers of the two circles in image = %.1f pixels\n',...
    pdist([image_center; centers_o], 'euclidean'))
prompt1 = 'Enter 1 to continue, else abort: ';
if input(prompt1) ~= 1
    error('Program terminated.')
end

% T = adaptthresh(I, 'NeighborhoodSize', 4 * floor(size(I) / 20) + 1);
% % T = adaptthresh(I, 'NeighborhoodSize', 4 * floor(size(I) / 16) + 1);  %
% % for pit 0615
% I1 = imbinarize(I, T);
% figure, imshow(I1)
% [B, L] = bwboundaries(I1, 'noholes');
% boundL = cellfun(@length, B, 'UniformOutput', 0); % Lengths of extracted boundaries
% boundLvec = cell2mat(boundL);
% [~, boundLvecsort] = sort(boundLvec, 'descend');
%
% % Outer boundary - For estimate of center
% for iii = 1: 1: length(boundLvecsort)
%     temp = B{boundLvecsort(iii)};
%     xmin = min(temp(:,2));
%     xmax = max(temp(:,2));
%     x0est = mean([xmin xmax]);
%     ymax = max(temp(:,1));
%     ymin = min(temp(:,1));
%     y0est = mean([ymin, ymax]);
%     if (abs(x0est - frame_size(2) / 2) < 0.05 * frame_size(2) / 2) && ...
%             (abs(y0est - frame_size(1) / 2) < 0.05 * frame_size(1) / 2) ...
%             && (xmin > 170 && xmin < 210)
%         ob_id = boundLvecsort(iii);
%         break
%     end
% end
%
% outer_bound = B{ob_id};  % since the longest boundary is probably the outer boundary
% hold on
% plot(outer_bound(:, 2), outer_bound(:, 1), 'b-')
%
% % ymax = max(outer_bound(:,1));
% % ymin = min(outer_bound(:,1));
% % y0est = mean([ymin, ymax]);
% %
% % xmin = min(outer_bound(:,2));
% % xmax = max(outer_bound(:,2));
% % x0est = mean([xmin xmax]);
%
% rsquare = sqrt((outer_bound(:,1) - y0est).^2 + (outer_bound(:,2) - x0est).^2);
% rest = max(rsquare);
% outer_pts_filt = rsquare > (0.98 * rest);
%
% Outer_Bound = [outer_bound(outer_pts_filt, 1) outer_bound(outer_pts_filt,2)];
% hold on
% plot(Outer_Bound(:,2), Outer_Bound(:,1), 'r--')
```

```
%
% % Fitting a non-linear least square fit to get center
% xdat = Outer_Bound(:,2);
% ydat = Outer_Bound(:,1);
% ytar = ydat.^2;
% lsfun = @(x) x(1).^2 - (xdat-x(2)).^2 + x(3).*(2*ydat - x(3)) - ytar;
% From equation of a circle
% X0 = [rest, x0est, 0.7 * y0est];
% lb = [0.95*rest, .4*frame_size(2), .4*frame_size(1)];
% ub = [1.05*rest, .6*frame_size(2), .6*frame_size(1)];
%
% X = lsqnonlin(lsfun, X0, lb, ub);
% outer_radius = X(1);
% % hold on
% % plot(X(2), X(3), 'r*')
%
% % Inner boundary
%
% for jjj = iii + 1: 1: length(boundLvecsort)
%     temp = B{boundLvecsort(jjj)};
%     ri = sqrt((temp(:,1) - y0est).^2 + (temp(:,2) - x0est).^2);
%     if all(ri < outer_radius)
%         ib_id = boundLvecsort(jjj);
%         break
%     end
% end
% % inner_bound = B{boundLvecsort(2)};  % since the 2nd longest boundary is
% probably the inner boundary
% inner_bound = B{ib_id};
% hold on
% plot(inner_bound(:, 2), inner_bound(:, 1), 'b-')
% % yimax = max(inner_bound(:,1));
% % yimin = min(inner_bound(:,1));
% % y0iest = mean([yimin, yimax]);
% %
% % ximin = min(inner_bound(:,2));
% % ximax = max(inner_bound(:,2));
% % x0iest = mean([ximin ximax]);
%
% % risquare = sqrt((inner_bound(:,1) - y0iest).^2 + (inner_bound(:,2) - x0iest).^2);
% riest = min(ri);
% inner_pts_filt = ri < (1.4 * riest);
%
% Inner_Bound = [inner_bound(inner_pts_filt, 1) inner_bound(inner_pts_filt,2)];
% hold on
% plot(Inner_Bound(:,2), Inner_Bound(:,1), 'm-')
%
% yimax = max(Inner_Bound(:,1));
% yimin = min(Inner_Bound(:,1));
% y0iest = mean([yimin, yimax]);
%
% ximin = min(Inner_Bound(:,2));
% ximax = max(Inner_Bound(:,2));
% x0iest = mean([ximin ximax]);
%
% % Fitting a non-linear least square fit to get center
% xidat = Inner_Bound(:,2);
```

```
% yidat = Inner_Bound(:,1);
% yitar = yidat.^2;
% lsfun = @(x) x(1).^2 - (xidat-x(2)).^2 + x(3).*(2*yidat - x(3)) - yitar;
% From equation of a circle
% Xi0 = [riest, x0iest, 0.7 * y0iest];
% lb = [0.95*riest, .4*frame_size(2), .4*frame_size(1)];
% ub = [1.05*riest, .6*frame_size(2), .6*frame_size(1)];
%
% Xi = lsqnonlin(lsfun, Xi0, lb, ub);
% hold on
% plot(Xi(2), Xi(3), 'mx')
%
% inner_radius = Xi(1);
% image_center = Xi(2:3);

%% Sectorizing the images


I = frames(100).cr_gray_pxdata;
figure, imshow(I)
hold on
[X_px, Y_px] = meshgrid((1:1:cr_frame_size(2)), (1:1:cr_frame_size(1)));
X_diff_px = X_px - image_center(1);
Y_diff_px = -(Y_px - image_center(2));   % Flipping here because y = 1 is
% actually the top row in image, not bottom
R_px = sqrt(X_diff_px .^ 2 + Y_diff_px .^ 2);
Theta_unadj_px = atan2(Y_diff_px, X_diff_px);
Theta_px = Theta_unadj_px;
Theta_px(Theta_unadj_px < 0) = Theta_px(Theta_unadj_px < 0) + (2 * pi);
Theta_deg_px = rad2deg(Theta_px);
contour(X_px', Y_px', Theta_deg_px', 24, 'ShowText', 'on')
figure, contourf(X_px', Y_px', R_px', 'ShowText', 'on')
figure, contourf(X_px', Y_px', Theta_deg_px', 12, 'ShowText', 'on')

% Sector half-width
sector_hw = 360 / 300;
theta_deg = 0 : 360 / 600 : 360 - 360 / 600;

% To see what's happening
% k2 = I;
figure;
for ii = 1:1:25
    k2 = bright_avg_img;
    k2(Theta_deg_px >= (theta_deg(ii) - sector_hw) &...
        Theta_deg_px < (theta_deg(ii) + sector_hw) & R_px > inner_radius &...
        R_px < 0.4 * outer_radius) = 255;
    imshow(k2)
    pause(0.1)
end

%% Normalization of images by mean intensity in interested area

intensity_a = nan(num_of_frames, 1);

for ii = 1:1:num_of_frames
    intensity_a(ii, 1) = sum(sum(frames(ii).cr_gray_pxdata(R_px > inner_radius...
        & R_px < 0.4 * outer_radius)));
```

```
end

for ii = 1:1:num_of_frames
    frames(ii).norm_pxdata = frames(ii).cr_gray_pxdata * (mean(intensity_a) /...
        intensity_a(ii));
end

%% BinSums and Filtering

% Visualizing bins to rearrange BinSum vectors

Theta_deg_px_adj = Theta_deg_px - 90;
Theta_deg_px_adj(Theta_deg_px_adj < 0) = Theta_deg_px_adj(Theta_deg_px_adj < 0) + 360;

figure;
k2 = bright_avg_img;
k2(Theta_deg_px_adj >= (theta_deg(27) - sector_hw) & Theta_deg_px_adj <...
    (theta_deg(27) + sector_hw) & R_px > inner_radius & R_px < 0.4 * outer_radius) = 255;
k2(Theta_deg_px_adj >= (theta_deg(574) - sector_hw) & Theta_deg_px_adj <...
    (theta_deg(574) + sector_hw) & R_px > inner_radius & R_px < 0.4 * outer_radius) = 255;
imshow(k2)

BinSums = zeros(length(theta_deg), num_of_frames);
BinSums_full = zeros(length(theta_deg), num_of_frames); % These are bin sums of bigger sections

for mm = 1:1:num_of_frames
    temp_img = frames(mm).norm_pxdata;
    for nn = 1:1:length(theta_deg)
        BinSums(nn, mm) = sum(temp_img(Theta_deg_px_adj >= (theta_deg(nn) -...
            sector_hw) & Theta_deg_px_adj < (theta_deg(nn) +...
            sector_hw) & R_px > inner_radius & R_px < 0.4 * outer_radius));
        BinSums_full(nn, mm) = sum(temp_img(Theta_deg_px_adj >=...
            (theta_deg(nn) - sector_hw) & Theta_deg_px_adj < (theta_deg(nn)...
            + sector_hw) & R_px > inner_radius & R_px < outer_radius));
    end
end

% Noticing that bins 1 to 26 and 575 to 600 cover the light blocker
BinSums_tr = BinSums(27:574, :);
BinSums_full_tr = BinSums_full(27:574, :);

% Quick visualization of BinSums_r
figure
hold on
for kk = 30: 1: 35
    plot(BinSums_tr(:, kk))
end

BinSums_tr_t = fft(BinSums_tr);
BinSums_full_tr_t = fft(BinSums_full_tr);
N = size(BinSums_tr_t, 1);
k = (2 * pi / (N)) * [0:(N / 2 - 1) (-N / 2):-1];
ks = fftshift(k);

filt = (exp(- 9 * k .^2))';

BinSums_tr_tf = BinSums_tr_t .* repmat(filt, [1 180]);
```

```
BinSums_tr_f = ifft(BinSums_tr_tf);

% Quick visualization of BinSums_tr_f
colorvec = (linspace(0, 0.8, 6))' .* [1, 1, 1];
figure
hold on
for kk = 30: 1: 35
    plot(BinSums_tr_f(:, kk) / max(BinSums_tr_f(:, kk)), 'Color', colorvec(kk - 29, :))
end
ylabel('BinSums normalized by max')
xlabel('Azimuthal bin location')
title('BinSums')

% Quick visualization of BinSums_tr_f
colorvec = (linspace(0, 0.8, 6))' .* [1, 1, 1];
figure
hold on
for kk = 30: 1: 35
    plot(ks, fftshift(abs(BinSums_full_tr_t(:, kk))) /...
        max(abs(BinSums_full_tr_t(:, kk))), 'Color', colorvec(kk - 29, :))
end
xlim([0 3.14])

%% Speed

prompt2 = 'Enter frame range of interest: ';
frames_of_int = input(prompt2);  % Change this from case to case
top_pks = nan(10, length(frames_of_int));
top_locs = nan(10, length(frames_of_int));
spk_pks = nan(2, length(frames_of_int));
spk_locs = nan(2, length(frames_of_int));
for kk = frames_of_int
    [allpks, alllocs] = findpeaks(BinSums_tr_f(:, kk), 'MinPeakDistance', 30);
    [temppks, tempids] = sort(allpks, 'descend');
    top_pks(:, kk - (frames_of_int(1) - 1)) = temppks(1:10);
    top_locs(:, kk - (frames_of_int(1) - 1)) = alllocs(tempids(1:10));
end

% Locating sectors where max intensity occurs across frames

c1 = 0;
temp_top_pks = top_pks;
temp_top_locs = top_locs;
for nn = 1: 1: 10

    [~, top_ids_descend] = sort(temp_top_pks(:), 'descend');
    temp_sel_sector = temp_top_locs(top_ids_descend(1));
    temp_ids = find((temp_top_locs > temp_sel_sector - 9) &...
        (temp_top_locs < temp_sel_sector + 9));
    if length(temp_ids) == length(frames_of_int)
        c1 = c1 + 1;
        spk_locs(c1, :) = temp_top_locs(temp_ids);
        spk_pks(c1, :) = temp_top_pks(temp_ids);
        temp_top_pks(temp_ids) = [];
        temp_top_locs(temp_ids) = [];
    else
```

```
        temp_top_pks(temp_ids) = [];
        temp_top_locs(temp_ids) = [];
        continue
    end
end

% Calculating angular speed

ang_spd = deg2rad(diff(spk_locs, 1, 2) * 360 / 600) * Fs_img; % in rad/s
ang_freq = ang_spd / (2 * pi);

max_ang_freq = zeros(size(ang_spd, 1), 1);

for ii = 1: 1: (length(frames_of_int) - 1)
    for jj = ii + 1: 1: length(frames_of_int)
        temp = deg2rad((spk_locs(:, jj) - spk_locs(:, ii)) * 360 / 600) *...
            (Fs_img / (jj - ii)) / (2 * pi);
        max_ang_freq = max(max_ang_freq, abs(temp));
    end
end

% Preparation for movie frames

spk_locs_theta_rd = deg2rad(theta_deg(spk_locs) + 90 + 26 * 0.6);
spk_rho = 0.4 * outer_radius;
% spk_rho = inner_radius + rescale(spk_pks, inner_radius, outer_radius);
spk_x = image_center(1) + spk_rho .* cos(spk_locs_theta_rd);
spk_y = image_center(2) - spk_rho .* sin(spk_locs_theta_rd);


%% Getting a video of the spoke movement
% colorvec = parula(4);
% c2 = 0;
% imagecount = length(frames_of_int);
% M(imagecount) = struct('cdata', [], 'colormap', []);
% writerObj = VideoWriter(strcat(filename, '.avi'));
% giffilename = strcat(filename, '.gif');
% writerObj.FrameRate = 2;
% open(writerObj);
% for kk = frames_of_int
%     c2 = c2 + 1;
%     fig = figure;
%     fig.WindowState = 'maximized';
%     s1 = subplot(4, 1, 1:3);
%     imshow(frames(kk).cr_gray_pxdata)
%     hold on
%     plot(image_center(1), image_center(2), 'ro')
%     for mm = 1: 1: 3
%         plot([image_center(1) spk_x(mm, kk - (frames_of_int(1) - 1))],...
%             [image_center(2) spk_y(mm, kk - (frames_of_int(1) - 1))],...
%             'Color', colorvec(mm, :), 'LineWidth', 3)
%     end
%     s2 = subplot(4, 1, 4);
%     s2.Box = 'on';
%     h1 = plot(DPO_data.img_time, DPO_data.img_CH4);
%     hold on
%     h2 = plot(DPO_data.img_time(kk), DPO_data.img_CH4(kk), 'r.');
```

```
%     h2.MarkerSize = 16;
%
%     M(c2) = getframe(fig);
%     writeVideo(writerObj, M(c2));
%
%     im = frame2im(M(c2));
%     [N1, map] = rgb2ind(im, 256);
%     if c2 == 1
%         imwrite(N1, map, giffilename, 'gif', 'LoopCount', Inf, 'DelayTime', 0.5);
%         else
%                 imwrite(N1, map, giffilename, 'gif', 'WriteMode', 'append', 'DelayTime', 0.5);
%     end
%
%     close(fig)
% end
% close(writerObj)

%% Uniformity Evolution

% Binsums_tr_t are the FFT of Binsums_tr

samp_freq = 2 * pi / 600;
xdft = BinSums_full_tr_t(1:(N / 2 + 1), :);
psdx = (1 / (samp_freq * N)) * abs(xdft) .^ 2;
psdx(2:end - 1, :) = 2 * psdx(2:end - 1, :);
freq1 = 0: samp_freq / N: samp_freq - samp_freq / N;

band_p = bandpower(BinSums_full_tr);
DC_power_dens = psdx(1, :) ./ band_p;
norm_DC_power_dens = DC_power_dens / max(DC_power_dens);

figure;
subplot(2, 1, 1)
plot(DPO_data.img_time, norm_DC_power_dens)
xlim([0 DPO_data.img_time(end)])
subplot(2, 1, 2)
plot(DPO_data.img_time, DPO_data.img_CH4, 'LineWidth', 2)
hold on
plot(DPO_data.time, DPO_data.CH4, 'LineWidth', 0.5)
xlim([0 DPO_data.img_time(end)])

% Spectrogram

fig3 = figure(3);
pcolor(1:180, freq1, abs(BinSums_full_tr_t)),
shading interp
ylim([0 samp_freq / 2])
xlabel('Frame')
ylabel('Frequency')
% set(gca, 'Ylim', [-50 50], 'Fontsize', [14])
colormap(pink)
fig3.Units = 'inches';
fig3.Position = [0 4 6.75 5.0625];
fig3.PaperUnits = 'inches';
fig3.PaperSize = [6.75 5.0625];
ax = gca;
ax.FontSize = 12;
```

Appendix C Additional figures

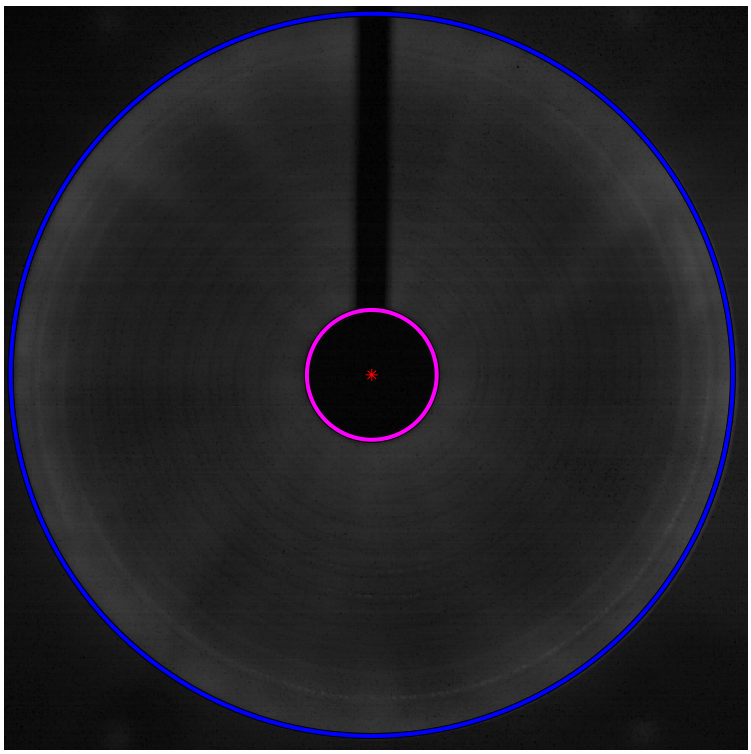

Figure 9: Representative image with center of the coil indicated by red *, the circle around the light block in magenta, and the circle around the outer coil face in blue.

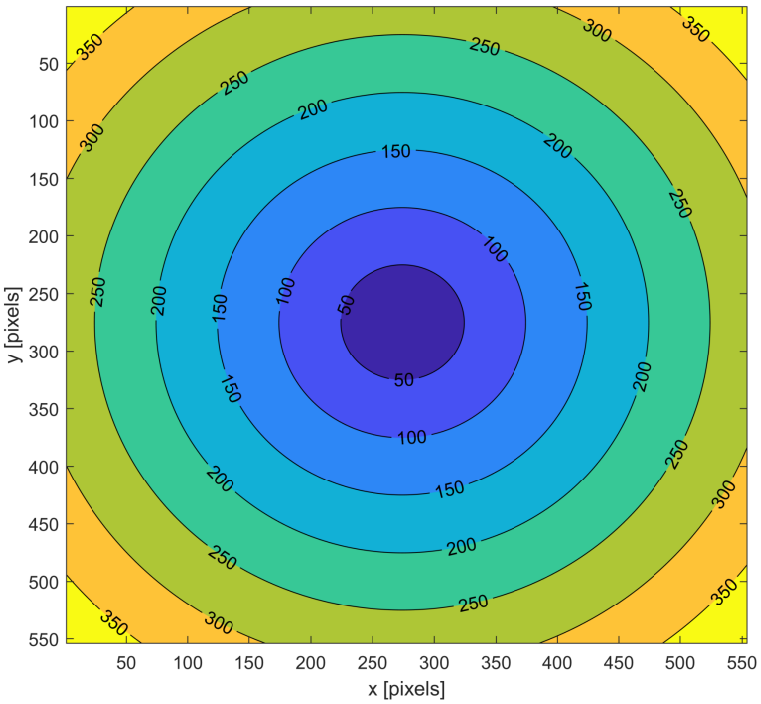

Figure 10: Plot showing the contours of radial distance from the center of the image.

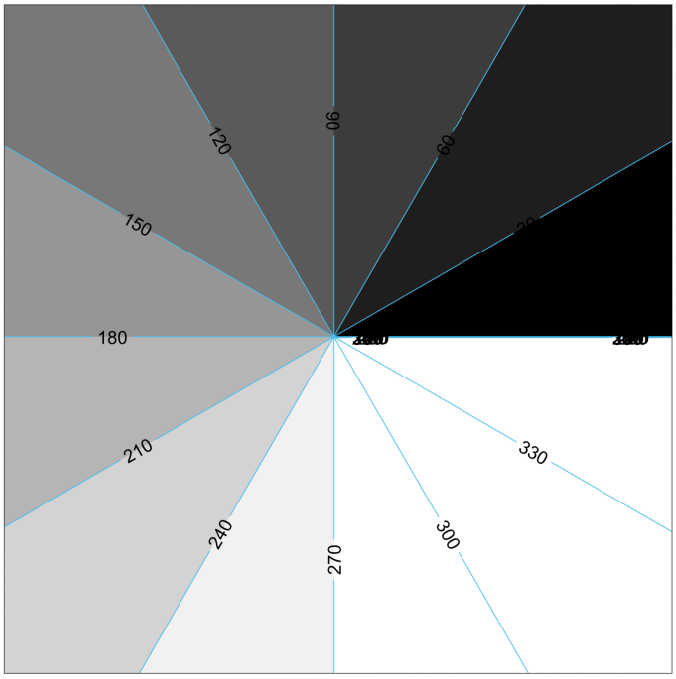

Figure 11: Plot showing the contours of polar angle around the center of the image.

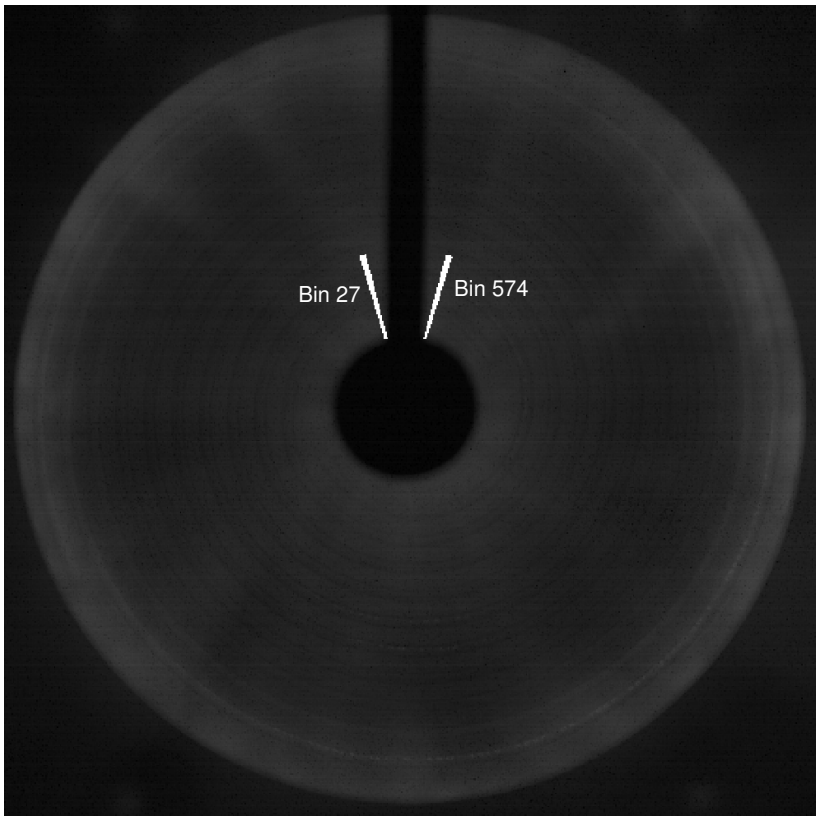

Figure 12: Representative image showing two bin locations. The truncated *BinSums_tr* matrix is built by assembling *BinSums* corresponding to the bins from 27 to 574, thus avoiding the bins covering the light block.