EMPIRICAL MODE DECOMPOSITION APPLIED TO PLANAR AND VOLUMETRIC VELOCITY FIELD MEASUREMENTS OF A SUPERSONIC SEPARATED FLOW

BY

MATTHEW DAVID KOLL

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Aerospace Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Advisers:

Professor J. Craig Dutton
Professor Gregory S. Elliott
ABSTRACT

The supersonic separated flowfield aft of a blunt-faced cylinder aligned with the freestream is highly complex such that a technique able to identify instantaneous turbulent structure within it is valuable. In this study, multi-dimensional extensions of fast and adaptive empirical mode decomposition (FAEMD) are implemented on both three-component planar and volumetric velocity fields of a Mach 2.5 supersonic base flow which were obtained using particle image velocimetry. The resulting two-dimensional intrinsic mode functions reveal the various length scales associated with different regions of the flowfield. Coherent streamwise-oriented structures of different scales were detected throughout the flowfield that indicate the presences of quasi-streamwise vortices. The presence of sharply angled structures, at about 45° to the local flow direction, suggests that both conventional- and counter-hairpin vortices are present within the flowfield, especially in the recompression zone and trailing wake. An autocorrelation analysis of the two-dimensional modes revealed the average size, orientation and shape of these different structures. The autocorrelation revealed that the largest flow structures reside in the shear layer due to the elongated nature of these structures in this region. The three-dimensional spatial analysis of this flowfield resulted in the identification of small-scale and large-scale instantaneous turbulent structures. Quasi-streamwise vortices and hairpin vortices were found during the three-dimensional analysis, in both the shear layer and the trailing wake. Linear stochastic estimation of these three-dimensional results revealed the presence of conventional hairpin and counter-hairpin vortices within the shear layer.
ACKNOWLEDGMENTS

I would like to thank my family for all of their support during my college career. I would not be here without them.

I must also thank both of my advisors, Dr. Dutton and Dr. Elliott. I have learned a great deal from the both of you, and I am greatly appreciative of your support throughout this project.

I would also like thank Dr. Ansell for sharing his knowledge of EMD at the beginning of this project and for answering all my random questions in the hallway of Talbot.

A special thanks to all my friends and coworkers. Thank you for welcoming me into the group when I joined two years ago.

The author would like to thank the U.S. Army Research Office for their support of this work with DURIP Equipment Grant No. W911NF-14-1-0404 and Research Grant No. W911NF-15-1-0194. Dr. Matthew Munson is the program monitor for both grants. The author would also like to thank James Favale, former graduate research assistant, UIUC Aerospace Engineering Department, and Branden Kirchner, Ph.D. candidate, UIUC Aerospace Engineering Department, for the acquisition and processing of the PIV measurement data used during this analysis.
# TABLE OF CONTENTS

NOMENCLATURE.................................................................................................................. v

Chapter 1: Introduction ........................................................................................................ 1

Chapter 2: Methodology ...................................................................................................... 4
  Empirical Mode Decomposition .......................................................................................... 4
  Ensemble Empirical Mode Decomposition ........................................................................ 9
  Fast and Adaptive Empirical Mode Decomposition ......................................................... 10
  Multi-Dimensional Fast and Adaptive Empirical Mode Decomposition ....................... 13
    Combining EEMD and FAEMD ....................................................................................... 18
  Code Verification and Parameter Discussion ................................................................... 18
    Code Verification ......................................................................................................... 18
    Lena Parameter Study ................................................................................................. 23

Chapter 3. Results and Discussion ...................................................................................... 27
  Introduction ..................................................................................................................... 27
  FABEEMD Results of S-PIV Data ................................................................................... 28
    Autocorrelation Analysis ............................................................................................. 37
    FABEEMD Computational Time ................................................................................... 41
  FATEEMD Results of Tomo-PIV Data .......................................................................... 41
    Linear Stochastic Estimation of TIMFs ....................................................................... 46

Chapter 4: Conclusions ....................................................................................................... 50

References .......................................................................................................................... 52

Appendix A: Multi-dimensional Ensemble Empirical Mode Decomposition .................... 56

Appendix B: FABEEMD Code (MATLAB) ....................................................................... 59

Appendix C: FATEEMD Code (MATLAB) ....................................................................... 65
**NOMENCLATURE**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEMD</td>
<td>bi-dimensional empirical mode decomposition</td>
</tr>
<tr>
<td>(FA)(E)EMD</td>
<td>(fast and adaptive) (ensemble) empirical mode decomposition</td>
</tr>
<tr>
<td>(M)(E)EMD</td>
<td>(multi-dimensional)(ensemble) empirical mode decomposition</td>
</tr>
<tr>
<td>IMF</td>
<td>intrinsic mode function</td>
</tr>
<tr>
<td>LSE</td>
<td>linear stochastic estimation</td>
</tr>
<tr>
<td>PIV</td>
<td>particle image velocimetry</td>
</tr>
<tr>
<td>S-PIV</td>
<td>stereoscopic (planar) particle image velocimetry</td>
</tr>
<tr>
<td>TEMD</td>
<td>tri-dimensional empirical mode decomposition</td>
</tr>
<tr>
<td>Tomo-PIV</td>
<td>tomographic (volumetric) particle image velocimetry</td>
</tr>
<tr>
<td>$c$</td>
<td>intrinsic mode function</td>
</tr>
<tr>
<td>$d$</td>
<td>nearest neighbor distances</td>
</tr>
<tr>
<td>$E$</td>
<td>stochastically estimated event</td>
</tr>
<tr>
<td>$g$</td>
<td>original signal</td>
</tr>
<tr>
<td>$IO$</td>
<td>index of orthogonality</td>
</tr>
<tr>
<td>$m$</td>
<td>mean envelope</td>
</tr>
<tr>
<td>$N_{mode}$</td>
<td>number of intrinsic mode functions extracted</td>
</tr>
<tr>
<td>$N_x$</td>
<td>total number of indices in the $x$ direction</td>
</tr>
<tr>
<td>$N_y$</td>
<td>total number of indices in the $y$ direction</td>
</tr>
<tr>
<td>$Q_{cr}$</td>
<td>Q-criterion</td>
</tr>
<tr>
<td>$r$</td>
<td>residual</td>
</tr>
<tr>
<td>$s$</td>
<td>sifting function</td>
</tr>
<tr>
<td>$u$</td>
<td>streamwise velocity component (notation for S-PIV)</td>
</tr>
<tr>
<td>$v$</td>
<td>transverse velocity component (notation for S-PIV)</td>
</tr>
<tr>
<td>$V_r$</td>
<td>radial velocity component (notation for Tomo-PIV)</td>
</tr>
<tr>
<td>$V_\theta$</td>
<td>tangential velocity component (notation for Tomo-PIV)</td>
</tr>
<tr>
<td>$V_a$</td>
<td>axial velocity component (notation for Tomo-PIV)</td>
</tr>
<tr>
<td>$w$</td>
<td>window size</td>
</tr>
<tr>
<td>$x$</td>
<td>axial coordinate, measured from base</td>
</tr>
<tr>
<td>$y$</td>
<td>transverse coordinate, measured from geometric centerline</td>
</tr>
<tr>
<td>$z$</td>
<td>spanwise coordinate</td>
</tr>
</tbody>
</table>
\( \varepsilon \quad = \quad \text{convergence tolerance} \)

\( \zeta_z \quad = \quad \text{out-of-plane vorticity component} \)

\( \lambda_{ci} \quad = \quad \text{three-dimensional swirling strength criterion} \)
Chapter 1: Introduction

The region of separated flow behind supersonic, axisymmetric blunt bodies is a research area of great interest because of its direct application to real-world problems. The wake aft of blunt bodies is turbulent in most applications, resulting in a highly complex, three-dimensional flow field. The region directly aft of the trailing edge of these bodies is also separated and causes a low-pressure recirculation region that results in an increase in aerodynamic drag, or base drag. The base drag of these objects can account for up to 35-50% of the total vehicle drag [Rollstin, 1987].

A simplified sketch of the near-wake base flow is shown in Figure 1.1, displaying the main features of this flowfield. It is important to note that this schematic represents a mean flowfield, and that instantaneous snapshots can vary greatly from the mean. A turbulent boundary layer forms along the cylindrical afterbody. The boundary layer separates at the shoulder of the base, forming a free shear layer that is bounded by the high-speed freestream on its outer edge and a low-speed recirculation region on its inner edge. A series of expansions waves also form at the shoulder of the base. This expansion fan acts to turn the high-speed freestream and free shear layer inwards towards the centerline of the base. The free shear layer eventually converges downstream of the rear stagnation point, forming the compressible trailing wake region. The high-speed freestream also converges and turns back towards the freestream direction, creating a recompression shock system.

![Figure 1.1. Simplified sketch of the mean flowfield [Reedy, 2013].](image-url)
Velocity measurements made of this flow field include two-component LDV [Herrin and Dutton, 1994], two-component planar PIV [Reedy, 2013], three-component stereoscopic planar PIV (S-PIV) [Favale et al., 2017], and three-component tomographic (tomo) volumetric PIV [Kirchner et al., 2017, 2018]. Of these four different measurement methods, the stereo and tomo-PIV techniques have proven the most useful for identifying key features within the flow field. LDV is limited to ensemble-averaged measurements of velocity at single points in space, whereas conventional PIV can obtain velocity measurements within a plane, simultaneously offering snapshots of the instantaneous velocities. S-PIV, an extension of planar-PIV, can measure three-component velocities, but the measurements are limited to a plane. Tomo-PIV can measure three-component velocities within a volume, albeit usually with a smaller field of view than for planar PIV.

Several computational methods have also been used to analyze this flow, including RANS [Papp and Ghia, 2001; Sahu 1986, 1994; Sahu and Nietubicz, 1994; Sahu and Heavey, 1995], LES [Fureby et al., 1999], DES [Forsythe et al, 2002; Subbareddy and Candler, 2005; Kawai and Fujii, 2007], and DNS [Sandberg and Fasel, 2006a and 2006b]. However, the results of critical parameters from these numerical investigations do not match with the experimental studies of this flow in many cases. For example, RANS studies of this flowfield predicted a radially decreasing centerline pressure along the base [Papp and Ghia, 2001; Sahu 1994]. However, experimental studies resulted in a pressure that was relatively constant across the base with a slight increase radially [Herrin and Dutton, 1994]. DNS studies; overall, have shown good agreement with experimental quantities, but are performed at much lower Reynolds number than typical applications to decrease the computational cost of the analysis [Sandberg and Fasel, 2006a, 2006b].

Recently, several useful flow structure/modal analysis techniques have emerged to help understand the nature of complex flowfields. These techniques include proper orthogonal decomposition (POD), dynamic mode decomposition (DMD), and empirical mode decomposition (EMD). POD [Favale et al., 2017] and DMD [Marié et al., 2013; Horchler et al., 2015; Statnikov et al., 2015] have both been used to study the low-order modes of both computational and experimental flowfields to extract dominant flow features. Recently, POD has been implemented to study the same flowfield being discussed in this thesis. The POD analysis indicated that a global axial pulsing motion is present, but presented little information about the turbulent instantaneous
flow structure [Favale et al., 2017]. Therefore, another approach is studied in this thesis, empirical mode decomposition, in order to identify the turbulent instantaneous flow structure.

Empirical mode decomposition was first introduced in 1998 by Huang et al. EMD is a nonparametric, adaptive data analysis technique used to identify simple oscillatory modes called intrinsic mode functions (IMFs), potentially even for nonlinear and nonstationary processes. Originally, EMD was developed as a one-dimensional signal analysis technique; however, several multi-dimensional EMD variants have been developed in recent years. Two techniques of interest are multi-dimensional ensemble empirical mode decomposition (MEEMD) and fast and adaptive bi/tri-dimensional ensemble empirical mode decomposition (FABEEMD/FATEEMD). MEEMD was first introduced by Wu et al. in 2009, and it has recently emerged as a useful analysis technique in fluid mechanics. Ansell and Balajewicz [2016] used MEEMD on velocity field measurements of an unsteady mixing layer obtained with time-resolved PIV. The analysis showed that small-scale turbulent structures could be separated from the large-scale vortical structures produced by the Kelvin-Helmholtz instability. Koll et al. [2017] analyzed velocity field information for a supersonic base flow obtained from both two and three-dimensional PIV measurements. This analysis showed the average size of the length scales seen in the IMFs and introduced the idea of reconstructing the image without the lowest-order IMFs to create a low-order approximation. However, because of the large number of vectors present in volumetric measurements, Koll et al. [2017] found that the MEEMD technique was too computationally intensive for any statistical analysis and was limited to only an instantaneous analysis of a few images. The second multidimensional technique, FAEMD, was introduced in 2008 by Bhuiyan et al. and will be the main topic of this thesis as applied to analyze a supersonic base flow. This method and results from it will be discussed in detail in the following chapters.
Chapter 2: Methodology

Empirical Mode Decomposition

Empirical mode decomposition is a technique that decomposes a signal into intrinsic mode functions (IMFs) and was first introduced by Huang et al. in 1998; thus, it is a relatively new technique for signal processing. A full description of an IMF can be found in the Appendix. In short, for a signal to be classified as an IMF, it must satisfy three criteria. First, the number of extrema and the number of zero-crossings in the signal must be equal or differ by at most one. Second, the upper and lower envelope of the signal must be symmetric about zero; in other words, the mean envelope of the signal is zero. Third, the set of IMFs obtained must form a complete and orthogonal basis [Sharply et al., 2006].

EMD implements a process known as sifting to decompose the signal into its IMFs. A visual representation of the EMD sifting process for a one-dimensional signal is shown in Figure 2.1. The sifting process starts with a temporally or spatially dependent signal $g(x)$. The local maxima and minima of $g(x)$ are first identified. Cubic splines are then fit across each set of extrema to define the upper and lower envelopes of $g(x)$. The upper and lower envelopes are then used to generate a mean envelope $m_1(x)$. The mean envelope is subtracted from the signal to produce the sifting function, $s_1(x)$:

$$s_1(x) = g(x) - m_1(x)$$  \hspace{1cm} (2.1)

The subscript denotes the current sifting iteration. Once a sifting function has been obtained, it is used in place of the original signal:

$$s_l(x) = s_{l-1}(x) - m_l(x)$$  \hspace{1cm} (2.2)

The subscript, $l$, denotes the current sifting iteration. In this case, $s_0(x)$ would be the original signal $g(x)$. The procedure of repeatedly calculating and subtracting the mean is known as the sifting process. The sifting process is repeated until, in most applications, a mean-squared tolerance stopping criterion is met, such that:

$$\frac{\sum_{l=1}^{N_x}(s_l - s_{l-1})^2}{\sum_{l=1}^{N_x}(s_{l-1})^2} < \varepsilon$$  \hspace{1cm} (2.3)
Figure 2.1. Example of the sifting process for a one-dimensional signal showing the envelopes for three sifting iterations, a-c, the first IMF after sifting, d, and the residual component remaining after the first IMF is extracted, e.
where \(i\) is the index of the discretized signal, \(l\) is the current sifting iteration, and \(\varepsilon\) is a prescribed convergence tolerance. The tolerance is often on the order of \(10^{-3}\) to \(10^{-6}\), but its value varies based on the type of data. For example, if multi-component velocity field data are available, then each velocity component that is processed with EMD, will have a unique sifting tolerance value.

Notice that the sifting stopping criterion says nothing about the mean envelope or the number of extrema and zero crossings, two of the three criteria that a signal must satisfy to be an IMF. This criterion is known as a “steady-state” criterion because it computes the change between the old sifting function and new sifting function [Ansell et al., 2016]. In other words, the sifting process is stopped when the sifting functions no longer experience dramatic changes from one iteration to another, hence “steady-state.” The effects of varying sifting tolerance levels will be discussed in the two-dimensional EMD section because of the focus on multi-dimensional analysis in this discussion.

Once the stopping criterion has been met, the final sifting function \(s_l(x)\) is extracted as the first IMF, \(c_1(x)\). Once the first IMF has been obtained, this IMF is used to compute a residual signal. The residual signal is computed by subtracting the IMF from the original signal:

\[
r_1(x) = g(x) - c_1(x)
\]

To obtain higher-order IMFs, the sifting process is repeated, but the newly acquired residual signal is used in place of the original signal. The process of sifting and subtracting IMFs can be repeated until either the number of IMFs desired have been found or no more oscillatory components are present in the residual signal, i.e., the residual signal is monotonic. In general, the frequency, either spatial or temporal, of the fluctuations within each IMF will decrease as the IMF number increases, unless mode mixing occurs. Mode mixing will be discussed further in the Ensemble Empirical Mode Decomposition section.

An advantage of this sifting process is that the original signal can be reconstructed by superimposing the IMFs onto one another. This creates the reconstructed signal, \(g'(x)\), and introduces an important quantity, the Reconstruction Error, \(RE\):

\[
RE = \left[ \frac{\sum_{i=1}^{N_x} (g'_i - g_i)^2}{\sum_{i=1}^{N_x} g_i^2} \right]^{1/2}
\]
Ideally, the reconstruction error should be as small as possible, to show that the signal was not altered during the sifting process. In other words, small \( RE \) proves that the set of IMFs is complete. The ability to superimpose the IMFs onto one another opens a few more possibilities when deciphering the results obtained from EMD. The signal could be reconstructed without the first few modes or the last modes to create a lower-order or higher-order approximation of the signal, respectively. Also, IMFs could be combined to create new components, such as the second and third IMFs could be combined to create a new signal that highlights different aspects of the signal that may not have been previously noticed with the IMFs separately.

It is not enough for the IMFs to form a complete basis; the IMFs must also form an orthogonal basis as well. Thus, a definition of the Index of Orthogonality, \( IO \), is made as follows:

\[
IO = \frac{\sum_{i=1}^{N_x} \sum_{l=1}^{N_{mode}} \sum_{k=l+1}^{N_{mode}} (c_l(i)c_k(i))}{\sum_{i=1}^{N_x} g^T(i)}
\]

\( N_{mode} \) is the number of IMFs extracted during the sifting process. For simplicity, the equation for the \( IO \) can be thought of as a dot product between two vectors consisting of the IMFs normalized by the square of the reconstructed signal. Thus, a low \( IO \) indicates a more orthogonal set of IMFs. This value can vary depending on the type of EMD being implemented, but in general an \( IO \) value less than about 0.1 is considered acceptable [Bhuiyan et al., 2008].

A synthetic signal was generated, as an example, to display the IMFs obtained and the separation of scales that occurs during the EMD process. The synthetic signal was generated using two different sine waves of varying frequency. The equations for the synthetic signal are as follows:

\[
q_1(x) = \sin(20\pi x) \\
q_2(x) = \sin(5\pi x) \\
q_3(x) = x^2 \\
q_{syn} = q_1 + q_2 + q_3
\]

The synthetic signal, and the three components that comprise it, are shown in Figure 2.2. Processing the signal with EMD results in two IMFs and a residual component. A comparison of the IMFs and synthetic components is shown in Figure 2.3. Examining Figure 2.3, EMD was
successfully able to extract the different length scales. The
edge effects should be noted, as the signal elsewhere
matches with the original synthetic components almost
exactly. However, this is a very well posed problem, as the
ratio of the frequency of the two sine components is
relatively high, 4. If the ratio of the frequencies was lower,
i.e., nearer-unity, then the original EMD technique may
have problems extracting the IMFs correctly. Possible
results include the mixing of the two frequencies into
different IMFs; this phenomenon is known as mode mixing
and is a potential weakness of EMD.

**Figure 2.2.** Synthetic signal to be processed with EMD.

**Figure 2.3.** Components of the synthetic signal compared to their IMF counterparts after EMD has been performed.
Ensemble Empirical Mode Decomposition

A drawback of the original EMD process is that mode mixing can occur between the IMFs obtained. Mode mixing is an effect of how EMD was designed to extract the IMFs. Only IMFs that clearly contribute to the signal maxima and minima can be identified and extracted in the sifting process. IMFs that are not able to clearly contribute extrema will not be able to be separated in the sifting process and therefore will remain mixed in another IMF. The amplitudes and frequencies of the signal components determine whether or not EMD is able to separate them into individual IMFs or mixed IMFs. While there are many sources that cause mode mixing, they can be broadly attributed to two causes: closely spaced spectral components or intermittency. Multiple methods have been developed to help combat the problem of mode mixing [Wu et al., 2009; and Yunchao et al., 2008] One of the more popular and effective methods for reducing/eliminating mode mixing is Ensemble Empirical Mode Decomposition (EEMD) introduced by Wu and Huang in 2009.

The core process of EMD is unchanged in EEMD, but additional steps are added before and after the sifting process. Before the sifting process begins, white Gaussian noise is added to the signal. The amplitude of the noise added is a fraction of the RMS of the signal, usually on the order of 20% of the signal RMS [Wu and Huang, 2009]. The purpose of the noise addition is to help slightly alter the different scales in the signal. The alteration of the length scales may help extract modes that were once mixed into others during the sifting process.

However, the noise is not just added once, but many times to generate an ensemble of noise-altered signals. The ensemble usually consists of on the order of 100 noise-altered signals [Wu and Huang, 2009]. EMD is applied to the entire ensemble of noise-altered signals and each signal produces a set of IMFs. For example, if five modes are required for extraction and an ensemble size of 100 is chosen, then there will be 500 IMFs for the ensemble: 100 first IMFs, 100 second IMFs, etc. There will also be 100 residual components. Each ensemble of IMFs is averaged to form the true IMFs and residual. A flow chart is shown in Figure 2.4, displaying the EEMD process.

White Gaussian noise is chosen because it contains a large variety of frequencies that can alter a signal on multiple scales, hence providing a one-fits-all solution. Additionally, white Gaussian noise can easily be eliminated/reduced through averaging, if one has a large enough ensemble size. The reconstruction error mentioned above becomes even more important in this
case because the signal is purposely being altered, and the true IMFs generated must still be able to reproduce the original signal.

**Figure 2.4.** Flow chart displaying the order of noise addition, sifting, and averaging required in the EEMD process.

**Fast and Adaptive Empirical Mode Decomposition**

A drawback to the original EMD techniques is computational time. The most computationally intensive step is finding the envelopes for sifting. The original method implements cubic splining across the minima and maxima. In one-dimensional space, this does not cause a significant computational expense, but when extended to multiple dimensions, the cost becomes quite significant. One such two-dimensional technique is Multi-Dimensional Ensemble Empirical Mode Decomposition (MEEMD), which is discussed more in the Appendix. MEEMD can be extremely slow because of the ensemble nature of the method. For multi-dimensional analysis, a faster method of finding envelopes is needed. To meet this need, Bhuiyan et al.
introduced the Fast and Adaptive Empirical Mode Decomposition (FAEMD) technique [Bhuiyan et al., 2008].

The sifting process in FAEMD is identical to that in the original EMD technique, but the generation of the mean envelope uses order-statistics filtering (OSF) instead of cubic splining. However, OSF requires a pre-determined window size to operate. Therefore, the first step in FAEMD must be to determine this window size. The window size is determined by finding the nearest neighbor distance between minima and maxima separately. The nearest neighbor distance is calculated using the Euclidean distance, and each extrema set will result in one distance creating two vectors of distances, $d_{min}$ and $d_{max}$, for the distances between minima and maxima, respectively. Using these distance vectors, four different unique cases can be calculated to use as the window size implemented in the filtering, known as types:

\[
\begin{align*}
    w_1 &= \min\{\min(d_{max}), \min(d_{min})\} \\
    w_2 &= \max\{\min(d_{max}), \min(d_{min})\} \\
    w_3 &= \min\{\max(d_{max}), \max(d_{min})\} \\
    w_4 &= \max\{\max(d_{max}), \max(d_{min})\}
\end{align*}
\]

(2.8)

Once the window size is determined, the value is rounded to the nearest odd integer for the filtering process. With the window sized determined, the original signal is processed using a minimum-statistics filter and a maximum-statistics filter to generate the minimum and maximum envelopes, respectively. However, just using the filters as is results in envelopes that resemble piece-wise constant step-graphs and are not suitable for the sifting process. Thus, a smoothing technique is implemented to create envelopes that resemble those that would be observed from cubic splining. The usually smoothing technique is simply a sliding average using the same window size as from the filtering. Since a sliding average is used for smoothing, the edges of the signal must be padded the appropriate amount based on the window size. The values for padding are simply the edge values repeated as many times as necessary. Once the envelopes are smoothed, the mean envelope is calculated and the sifting process begins as it would normally.

Since the development of FAEMD, more window types have been developed than the original four. A fifth type was developed and implemented as the mean of the first four types [He and Liu, 2016]:
\[
    w_5 = \frac{w_1 + w_2 + w_3 + w_4}{4}
\]  
(2.9)

This was created because the window sizes do not always grow with IMF order as expected. For example, the minimum window size possible is three, and if the scale corresponding to a window size of three is subtracted from the signal, then the next IMF should have a larger window size. Instead, the window size could remain constant and generate oscillatory effects around any sharp changes in the signal. On the other hand, sometimes the window size grows too large too quickly because of an outlier extremum, especially if type four is chosen. By averaging the different types, this helps to prevent these types of problems from happening during the sifting process. Lastly, a sixth type simply computes an average distance between extrema by dividing the length of the signal by the number of extrema [Du et al., 2018]:

\[
    w_6 = \frac{\text{length of signal}}{\# \text{ of extrema}}
\]  
(2.10)

The definition of type six is modified when working in more than one dimension, and the equations will be shown in the corresponding multi-dimensional sections.

![Figure 2.5](image)

**Figure 2.5.** Comparison between mean generation with cubic splining (top) and order-statistics filtering (bottom); modified from Byuihan et al., 2008.
Even though there are more steps involved, overall FAEMD is faster than the original EMD technique, primarily due to the simplicity of the order-statistics filtering compared to cubic splining. Furthermore, FAEMD helps minimize the edge effects that normally occur in the original EMD technique, such as the upper and lower envelopes crossing at the edges. However, the FAEMD technique does not track the extrema as closely as what would be seen in cubic splining. In some cases, however, the mean envelope generated by the FAEMD process better tracks the trends observed in the original signal than the mean envelope generated from cubic splining. Bhuiyan et al. discuss this, and a simple example is shown in Figure 2.5. One can observe from Figure 2.5 that the mean envelopes closely resemble one another even though the minimum and maximum envelopes are quite different from each other in places. The upper and lower envelopes for the cubic splining case intersect one another at the right edge of the signal, which is problematic. On the other hand, the upper and lower envelopes for the FAEMD case do not intersect and are able to more accurately preserve the trends one might expect if the signal was extended at the edges.

Multi-Dimensional Fast and Adaptive Empirical Mode Decomposition

There are several different types of Multi-Dimensional Empirical Mode Decomposition techniques. Two of the more popular ones are Bi-Dimensional Empirical Mode Decomposition (BEMD) and Multi-Dimensional Ensemble Empirical Mode Decomposition (MEEMD). BEMD relies on very computationally intensive splining techniques to form the upper and lower envelopes, such as thin-plate splining, and is limited to two-dimensional signals. MEEMD decomposes multi-dimensional signals by making a series of one-dimensional passes in each direction. Depending on the size of the signal and the number of dimensions, MEEMD can take an extraordinary amount of time to compute. Even when MEEMD is performed, the IMFs obtained are not truly two-dimensional, but rather are only quasi-two-dimensional because the signal is decomposed in each dimension separately. Koll et al. studied MEEMD for the same velocimetry data that will be discussed in the results sections herein, and a more robust explanation of the MEEMD process and computational timing can be found in the Appendix.

Fast and Adaptive Bi-Dimensional Empirical Mode Decomposition (FABEMD) and Fast and Adaptive Tri-Dimensional Empirical Mode Decomposition (FATEMD) were developed to overcome the computational cost of other multi-dimensional techniques. The overall concept of
FABEMD and FATEMD is the same as described for FAEMD, but now the windows used for filtering and smoothing are squares and cubes, respectively. A detailed discussion of FABEMD will follow using visual guides to show how each step works. A seven by seven array of integer values is used to show the process. This array is shown in Figure 2.6. As mentioned before, the local maxima and minima need to be identified first. This is done by using the eight nearest neighbors surrounding each point. For a point to be considered a maximum or minimum, it must be strictly greater than or less than, respectively, all of its neighbors. Only the nearest three and five neighbors are used to find the extrema of corner and side points, respectively, in the array. The maxima and minima for this example can be found in Figure 2.7.

<table>
<thead>
<tr>
<th>8</th>
<th>10</th>
<th>7</th>
<th>8</th>
<th>3</th>
<th>9</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>6</td>
<td>9</td>
<td>8</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 2.6. Original signal for the FABEMD example.

Figure 2.7. Maxima (left) and minima (right) maps for the original signal.

The next step is to find all of the nearest-neighbor distances for each extremum. This can be accomplished in a variety of different ways. One of the more popular methods is to use
Delaunay triangulation to quickly find the two nearest extrema for each point that form the Delaunay triangle. This reduces the number of extrema that need to be checked and is faster than using a brute force method. Since this is a simple example, it is easy to infer the nearest neighbor for each extremum.

The next step is to define the window size that is needed. Table 2.1 displays the different types available and their corresponding window sizes. For Type 6, a new definition is needed for the multi-dimensional data; therefore, two- and three-dimensional definitions are also give below:

<table>
<thead>
<tr>
<th>Window Size</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Type 3</th>
<th>Type 4</th>
<th>Type 5</th>
<th>Type 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_6 = \sqrt{N_xN_y}$ (two-dimensional)</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$w_6 = \sqrt[3]{N_xN_yN_z}$ (three-dimensional)</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

where $N$ denotes the number of points in the subscripted dimension and $n$ is the total number of extrema in the array. Type 6 provides a quick way to calculate and average neighbor distances without having to introduce Delaunay triangulation. Choosing Type 6, the window size for the filtering process is three. For this case, the center of a $3 \times 3$ square window is moved across each point of the array. The maximum and minimum are identified inside of the windowed domain. All points inside of the windowed domain are changed to the maximum and minimum to generate the maximum and minimum envelopes, respectively. For the sides and corners, the window is changed to a $3 \times 2$ rectangle and a $2 \times 2$ square, respectively. The results of the filtering step can be seen in Figure 2.8.
At this stage, the upper and lower envelopes are starting to take shape, but there are a few more steps before the calculation of the mean envelope. The next step is to pad the edges of the matrix so that it can be smoothed. The amount of padding on the edges is determined by the window size. The number of rows and columns added to the matrix is the window size minus one. So, if the window size is three, then two rows and two columns will be added. One column will be added to the right and left sides of the envelopes. One row will also be added to the top and bottom sides of the envelopes. The values of these added elements will simply be repeating values of the edges. The padded envelopes can be seen in Figure 2.9 for this example. It is important to note that the envelopes are now nine by nine in size.

Figure 2.8. Filtered upper (left) and lower (right) envelopes for the example problem.

Figure 2.9. Padded upper and lower envelopes for the simple example.
The final upper and lower envelopes can now be found by smoothing the padded matrices. This is the last step before the mean envelope is calculated. A simple sliding window average is used to smooth the envelopes. The window used in this averaging method is the same size as found in the filtering process. The center of the window slides through each point of the original matrix and calculates the mean of this window. Thus, the padded rows and columns are only used when the window is centered on what would be the edges or corners of the original matrix. The smoothed upper and lower envelopes, as well as the mean envelope are shown in Figure 2.10. Different forms of smoothing could be applied to generate the upper and lower envelopes, but for the sake of simplicity and computational time, only the sliding window average is used in this discussion.

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>10</th>
<th>9.78</th>
<th>9.44</th>
<th>9.11</th>
<th>9</th>
<th>9</th>
<th></th>
<th>2</th>
<th>2</th>
<th>2.33</th>
<th>2.67</th>
<th>2.33</th>
<th>1.67</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>9.33</td>
<td>9.67</td>
<td>9.89</td>
<td>9.44</td>
<td>9.11</td>
<td>8.89</td>
<td>9</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2.22</td>
<td>2.44</td>
<td>2.33</td>
<td>2</td>
<td>1.67</td>
</tr>
<tr>
<td>9.33</td>
<td>9.67</td>
<td>9.89</td>
<td>9.44</td>
<td>9.11</td>
<td>8.89</td>
<td>9</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2.22</td>
<td>2.44</td>
<td>2.33</td>
<td>2</td>
<td>1.67</td>
<td></td>
</tr>
<tr>
<td>8.89</td>
<td>9.44</td>
<td>10</td>
<td>9.56</td>
<td>9.22</td>
<td>8.89</td>
<td>9</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2.11</td>
<td>2.22</td>
<td>2.11</td>
<td>1.89</td>
<td>1.67</td>
<td></td>
</tr>
<tr>
<td>8.22</td>
<td>8.89</td>
<td>9.56</td>
<td>9.33</td>
<td>9.11</td>
<td>7.89</td>
<td>9</td>
<td></td>
<td>1.67</td>
<td>1.67</td>
<td>1.67</td>
<td>1.78</td>
<td>1.78</td>
<td>1.78</td>
<td>1.67</td>
<td></td>
</tr>
<tr>
<td>8.22</td>
<td>8.67</td>
<td>9.11</td>
<td>9.11</td>
<td>8.78</td>
<td>8.33</td>
<td>8</td>
<td></td>
<td>1.33</td>
<td>1.33</td>
<td>1.33</td>
<td>1.44</td>
<td>1.33</td>
<td>1.22</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8.22</td>
<td>8.22</td>
<td>8.22</td>
<td>7.67</td>
<td>7.33</td>
<td>7</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8.11</td>
<td>7.78</td>
<td>7.44</td>
<td>6.67</td>
<td>6.33</td>
<td>6</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2.10.** The smoothed upper and lower envelopes, top right and top left, and the mean envelope, middle bottom, for the example.
The convergence sifting criterion above must be expanded for two and three dimensions, and the tolerance level must be set accordingly. Once the sifting is completed, the remaining sifting function becomes the first Bi- or Tri-dimensional IMF (BIMF or TIMF, respectively). The first BIMF or TIMF is then subtracted from the signal, and the residual is used as the starting point for finding a new IMF. This process repeats itself until either a specified number of IMFs have been extracted or the residual is monotonic in nature and no more IMFs can be extracted. Once again, it is important to verify the completeness and orthogonality of the IMFs by quantifying the reconstruction error and the index of orthogonality. Like the convergence criterion, the \( RE \) and \( IO \) definitions must be extended into multiple dimensions.

**Combining EEMD and FAEMD**

A code, developed in-house, combines the fast and adaptive method with the noise-assisted ensemble method. This results in a hybrid Fast and Adaptive Bi-/Tri-Dimensional Ensemble Empirical Mode Decomposition (FAB/TEEMD) method. Combining these two methods provides a fast EMD method that is less susceptible to mode mixing. The processes are the same as discussed previously. The flow chart in Figure 2.4 can still be followed, but the signals are now two- or three-dimensional and the original EMD method is replaced with the fast and adaptive variants. The codes were developed in MATLAB and can be found in the Appendix.

**Code Verification and Parameter Discussion**

**Code Verification**

Since both a FABEEMD and FATEEMD code were developed in-house, the first step was to verify the results of the code. Lena, seen in Figure 2.11, is an extremely well-known image for image processing. Another well-known set of images are the Brodatz textures, an example of which is shown in Figure 2.11. Lena and one of the Brodatz textures, D11, are used to validate the FABEEMD code integrity by comparing to results found in the literature. Due to the lack of standard images for three-dimensional cases, the FATEEMD code could not be tested. So, all lessons from the FABEEMD study apply to the FATEEMD code, as well.
Figure 2.11. Lena (left) and Brodatz Texture, D11 (right).

It is important to quantify all the different input parameters before running the FABEEMD code. These parameters include the number of modes desired, $N_{\text{mode}}$, the sifting convergence criterion, $\epsilon$, the window type for the filtering process, Type, the amplitude of the noise, $N_{\text{amp}}$, and finally the number of noise-assisted signals in the ensemble, $N_{\text{esb}}$. Lena is the first case to be run for verification. The input parameters for Lena can be found in Table 2.2. Choosing the parameters is not straightforward and requires multiple trial and error runs before settling on the final values. A parameter study will be discussed specifically for Lena in the next sub-section to show the process of choosing the parameters.

<table>
<thead>
<tr>
<th>$N_{\text{mode}}$</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>0.002</td>
</tr>
<tr>
<td>$N_{\text{esb}}$</td>
<td>100</td>
</tr>
<tr>
<td>$N_{\text{amp}}$</td>
<td>0.2</td>
</tr>
<tr>
<td>Type</td>
<td>6</td>
</tr>
</tbody>
</table>

The results for Lena using these parameters can be found in Figure 2.12. These results are similar to results found in the literature [e.g., Bhuiyan et al., 2008; Wu et al., 2009]. Bhuiyan uses a picture of Elaine instead of Lena, but the overall trends and features can still be observed between the two. Wu et al., uses Multi-Dimensional Ensemble Empirical Mode Decomposition (MEEMD). This provides a larger basis to compare to already well-known techniques and similar images.
Looking at the BIMFS obtained from FABEEMD in Figure 2.12, it is clear to see the trend of growing length scales with BIMF order. The first BIMF contains the smallest scales in an image. For Lena, the impressively sharp smallest-scale oscillations are at the borders of her hat and facial features. The second and third BIMFs extract larger-scale features, such as the slight change in contrast in the feathers of Lena’s hat. The fourth and fifth BIMFs continue this trend of growing scale, such as the contrasts within Lena’s face and even the dark edge of the mirror compared to the rest of the background. Upon inspecting the residual, clearly there are still some large-scale fluctuations left in the signal. This means that more BIMFS could be extracted if desired, until the signal becomes strictly monotonic in nature. Even with some large fluctuations in the residual, a mean trend can still be seen. This trend includes the overall light contrasts between the dark regions, the feathers and Lena’s hair, and the light regions, Lena’s face and shoulder.

Figure 2.12. First five BIMFs, a-e, and the residual, f, of Lena obtained from FABEEMD.

The orthogonality and completeness of the BIMF set should also be reported to validate the decomposition. The $IO$, $RE$, and computational time are shown in Table 2.3. The low
RE value indicates that the BIMFs can reconstruct the original signal accurately; thus, the BIMF set for Lena is complete. The low IO value indicates that the set forms a nearly orthogonal basis. The last value, computation time, is important because it is the main advantage of FAEMD and its multi-dimensional variants. The total time from start to finish of the FABEEMD technique is 35 seconds parallelized on four cores operating at an average of 50% efficiency. The image of Lena is 256 x 256 pixels. Using another in-house code for MEEMD, the computational time for Lena is several minutes. MEEMD is discussed more in the Appendix, and more comparisons between FAEMD and MEEMD are made there.

Table 2.3. IO, RE, and computational time for the decomposition of Lena shown in Fig. 2.12.

<table>
<thead>
<tr>
<th>RE</th>
<th>0.007%</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO</td>
<td>0.07</td>
</tr>
<tr>
<td>Time [sec.]</td>
<td>35</td>
</tr>
</tbody>
</table>

Next, the D11 Brodatz texture image (640 x 640 pixels) is decomposed using the in-house FABEEMD code. This image poses a different type of problem than Lena; see Figure 2.11. D11 contains several varying scales of oscillatory fluctuations. There is the large stripe pattern across the images, but there are also light threads woven between the darker thread in the dark stripes. There are also a few randomly scattered regions of lighter and darker contrast in the image. The input parameters for this decomposition are shown in Table 2.4. The first eight BIMFs and residual of the D11 decomposition are displayed in Figure 2.13. The RE, IO, and computational time are shown in Table 2.5. The BIMFs obtained in this analysis are fairly similar to the ones decomposed by He et al. [2016]; the scales grow at a similar rate and the same relative features are extracted.

Table 2.4. Input parameters for D11.

<table>
<thead>
<tr>
<th>N\text{mode}</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>\epsilon</td>
<td>0.002</td>
</tr>
<tr>
<td>N_{esb}</td>
<td>100</td>
</tr>
<tr>
<td>N_{amp}</td>
<td>0.2</td>
</tr>
<tr>
<td>Type</td>
<td>6</td>
</tr>
</tbody>
</table>
Figure 2.13. First eight BIMFs and residual of the D11 Brodatz texture from the FABEEMD analysis performed.

Table 2.5. $IO$, $RE$, and computational time for the decomposition of D11 shown in Fig. 2.13.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$RE$</td>
<td>0.01%</td>
<td></td>
</tr>
<tr>
<td>$IO$</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>Time [sec.]</td>
<td>342</td>
<td></td>
</tr>
</tbody>
</table>
Lena Parameter Study

As mentioned in the previous section, selecting the values for the input parameters is not particularly straightforward. The main idea is to lower the index of orthogonality and reconstruction error, while extracting IMFs that intuitively make sense. The $IO$ and $RE$ can easily be measured to find the best combination of parameters that lowers their values. The last point, however, is more vague and comes down to a simple question: “What are you looking for?” This has a variety of answers, all of which can lead to different solutions. For example, in the decomposition of Lena we are looking for fluctuations in the light contrast. In the image, there are a few obvious contrast changes that can be observed and searched for in the computed IMFs, such as the contrasts changes around her hat and facial features and the shadow of her nose, which lead to larger-scale fluctuations around these features. This vague criterion suggests that there is some amount of a priori knowledge needed when deciding on the parameters for extracting IMFs, which will be discussed more in the Results and Discussion section. Due to the vagueness of the last criterion, the focus of this parameter study will be on the quantifiable values: $IO$, $RE$, and computational time.

The first parameters to be studied are the ensemble size, $N_{esb}$, and the amplitude of the noise, $N_{amp}$. Although these parameters could be studied separately, it is more intuitive to study them together, since they are directly correlated. In this study, $N_{amp}$ was varied from 0.025 to 0.3 using steps of 0.025. $N_{esb}$ was varied from 10 to 200 using steps of 10. The results were used to create contour maps for $IO$, $RE$, and computational time as shown in Figure 2.14.

Figure 2.14 contains a few interesting trends that can be used when implementing FABEEMD. Firstly, it states that the orthogonality of the BIMFs is not dependent on the size of the ensemble. The $IO$ graph shows that the amplitude of the added noise has a more direct influence on the orthogonality. This makes sense, given that mode mixing decreases the orthogonality of the IMFs, and a strong noise amplitude helps more individual fluctuations be found. However, the $RE$ graph shows that a larger noise amplitude also leads to more reconstruction error. One must also be careful that the noise does not dominate the entire signal on all scales, since this can lead to problems. The large reconstruction error is easily countered by increasing the ensemble size, but at the cost of more computational time. The ideal combination lies between the extremes of every case and can be determined on an as-needed basis.
Figure 2.14. The IO (left), RE (right), and computational time (center) for the $N_{amp}$ and $N_{esb}$ parameter sweep performed on Lena.

The next parameter to be studied is the sifting convergence tolerance, $\varepsilon$. This study was done by sweeping through different tolerance levels and studying the IO and RE. FABEEMD is very sensitive to the number of sifting iterations performed. The signal can very easily be over-sifted and under-sifted; thus, computational time is not measured because of the strict need for the

Figure 2.15. IO and RE for varying sift convergence criteria.
correct number of sifting iterations. Thus, the computational time mainly varies with the ensemble size and number of modes being extracted. Figure 2.15 displays the IO and RE for varying tolerance levels. Figure 2.16 displays the third BIMF of Lena for a few different tolerance levels.

Figure 2.16. Six variants of the third BIMF of Lena. The sifting criterion is the largest at the top left and smallest at the bottom right.

In general, as the sifting criterion decreases, the number of sifts increases. Figure 2.15 shows that the index of orthogonality decreases with more strict tolerance levels. Thus, more sifting iterations create more orthogonal modes. The figure also shows that the reconstruction error is independent of the sifting criterion. However, the decreasing orthogonality is not necessarily a good thing. Figure 2.16 shows that BIMF3 of the Lena decomposition does not change much until over-sifting occurs. The over-sifting is a result of the window sizes no longer growing as the BIMF number increases, thus creating an abnormal series of fluctuations around Lena’s border features. These figures indicate that the sifting convergence criterion is a give-take system for the type of modes desired. Obviously, a low IO is desired, but in most cases one must pick a moderate IO value to avoid over-sifting effects.
The last parameter studied is window size, Type. This determines which of the six methods is used to find the window size for the entire sifting process. Since all the other parameters are set, the decomposition of Lena was performed for all six different methods. Figure 2.17 displays how the window size grows for each mode and each Type. It can be noted from Figure 2.17, that Types 1&2 lead to a near constant window size. This is not ideal, because the window size should be growing as the smaller lengths scale subtract out for finding the higher-order BIMFs. Types 3&4 become too large, too quickly, and the window size becomes on the order of the size of the original image. Types 5&6 follow an expected trend. The windows sizes grow as the BIMF number increases, plus the window size grows smoothly. There is no sudden increase or decrease in the size. Therefore, Types 5 and 6 are the recommended methods, at least for this case.

Figure 2.17. Window size of each BIMF for the different types of window methods.
Chapter 3. Results and Discussion

Introduction

The S-PIV data obtained by Favale et al. [2017] are studied during this analysis. The data sets were obtained at a University of Illinois at Urbana-Champaign base flow facility, using a blow-down supersonic, axisymmetric wind tunnel. A four-camera setup was used to obtain instantaneous S-PIV measurements of the entire flow field. Two cameras were used to measure the upstream region, and two additional cameras were used to image the downstream region. The resulting vector fields were then stitched together to form a single instantaneous velocity snapshot of the flowfield. The resulting velocity vector fields contain information for all three velocity components within a plane. Figure 3.1 displays a typical instantaneous planar velocity magnitude field. Further description of the experiments can be found in Favale et al. [2017] and Favale [2017].

![Figure 3.1. Typical instantaneous planar velocity magnitude field](image1.jpg)

The second data set comes from tomo-PIV data obtained by Kirchner et al. [2017, 2018]. The tomo-PIV measurements were obtained in the same base flow facility mentioned previously. A four-camera system was also implemented for this experimental setup. Velocity field information

![Figure 3.2. Typical instantaneous tomo-PIV velocity magnitude field](image2.jpg)
was measured in a volume around the recompression/reattachment zone of the base flow [Kirchner et al., 2017]. Recently, volumetric measurements in the shear layer have also been made of this flowfield [Kirchner et al., 2018]. The resulting velocity information contains all three velocity components, but now throughout a volume instead of on a plane. A typical volumetric velocity magnitude field of the shear layer region can be seen in Figure 3.2.

**FABEEMD Results of S-PIV Data**

First, FABEEMD is performed on the three-component planar S-PIV results. However, the analysis is not performed on the entire flowfield at once. This is due to the nature of FABEEMD. The length scales of the resulting IMFs from FABEEMD are determined by the window size. So, performing the analysis across the entire image would assume one length scale across the entire image. Therefore, the flowfield is divided into four different sub-regions, as shown in Figure 3.3. Region I contains the recirculation zone directly behind the base. Region II contains a portion of the shear layer downstream of the base. Region III contains the recompression/reattachment zone around the mean stagnation point. Region IV contains the trailing wake just beyond Region III. These are the same regions used by Koll et al. [2017], when performing a MEEMD analysis of this flow. For completeness, the ensemble size and noise coefficient are 100 and 0.2, respectively, for this analysis.

![Reconstructed stereo-PIV instantaneous velocity magnitude scalar field with Regions I-IV marked in boxes [Koll et al., 2017].](image)

**Figure 3.3.** Reconstructed stereo-PIV instantaneous velocity magnitude scalar field with Regions I-IV marked in boxes [Koll et al., 2017].

For an instantaneous analysis, FABEEMD is used to process three different S-PIV instantaneous velocity fields as examples: Field 291, Field 1422, and Field 2273. Figure 3.4
displays the results of the FABEEMD analysis performed on velocity magnitude for Field 291 in Region II, the upper shear layer region. Black contour lines of the 0.1 and 0.9 $|u|/U_\infty$ magnitudes are added as a reference to approximately locate the freestream and recirculation regions that

\[
\begin{array}{cccc}
\text{a} & \text{BIMF 1} & \text{b} & \text{BIMF 2} \\
\text{c} & \text{BIMF 3} & \text{d} & \text{BIMF 4} \\
\text{e} & \text{Residual} \\
\end{array}
\]

**Figure 3.4.** First four velocity magnitude BIMFs, a-d, and residual, e, of Image 291, in Region II.

bound the shear layer in the images.

The first BIMF displays small-scale structure and noise throughout the entire field. However, due to the addition of noise in the analysis, the small-scale structure and noise cannot be distinguished from one another. The second BIMF displays structure of a larger-scale. These structures are split from being streamwise-oriented and slightly angled with respect to the local flow direction. The streamwise-oriented structures could indicate the presence of quasi-streamwise vortices. These structures are similar to the longitudinal structures observed by
Sandberg and Fasel [2006a] in DNS at lower Reynolds number, while the strongly angled, ~45° structures could indicate the presence of conventional-hairpin and/or counter-hairpin vortices [Kirchner et al., 2018]. These hairpin vortices have been observed using linear stochastic estimation of the same flowfield by Kirchner et al. [2018] and numerically with DNS by Sandberg and Fasel [2006a, 2006b], albeit at lower Reynolds number. Similar structures also appear in the third BIMF, but on a larger scale. By the fourth BIMF, most instantaneous structure appears to have been extracted. Instead, the main features captured are mean velocity variations across the shear layer, until approximately 2.3 radii downstream of the base. After this point, a large angled structure is present as the flowfield transitions into the recompression zone. Finally, the residual resembles the mean flowfield at this location and offers no more insight into the instantaneous structure. Overall, the second and third BIMFs appear to hold the most physical significance, with the potential for the fourth BIMF as the length scales are potentially larger in the recirculation region, recompression zone, and trailing wake.

Figure 3.5. The second, third, and fourth BIMFs of velocity magnitude associated with Region I for the three instantaneous velocity fields.
Figure 3.5 displays the BIMFs of interest for Region I of all three velocity fields. Once again, the trend of increasing length scale is seen with increasing BIMF number in all three instantaneous fields, as expected, demonstrating the scale separation property of EMD. The scales between the images for a given BIMF number also appear to be relatively similar in size. BIMF 2 contains the smallest scales, and the structures in these BIMFs appear to be rounder and randomly oriented in the upstream portion. The structures in the downstream portion of the images in the second BIMFs are more elongated and appear to be more oriented with the shear layer. The third BIMFs display similar trends to the second BIMFs; however, more elongated structures occur in the upstream portion. This could indicate the presence of longitudinal vortices that occur in the region due to the instabilities around the base [Sandberg and Fasel, 2006a]. The fourth BIMFs are similar to the third BIMFs, but there are some extremely large fluctuations occurring that could be more indicative of the mean trends that would occur in this region, as seen in Field 1422. Overall,

**Figure 3.6.** The second and third BIMFs of velocity magnitude associated with Region II for the three instantaneous velocity fields.
the fluctuations are larger in amplitude in the downstream portion than upstream portion in this region.

Figure 3.6 displays the BIMFs of interest associated with Region II for the three instantaneous velocity fields. In the second BIMFs, there is a large variety of structures present and these structures can be broken, nominally, into two separate sub-regions: the upstream portion, from $x/R_0 = 1.4$ to 2, and downstream portion, $x/R_0 = 2$ to 2.8. Starting with the upstream portion, the structures appear to be more elongated and oriented with the local flow direction. There are some slightly angle structures in this region, and these structures can be angle either upwards, towards the freestream, or downwards, towards the recirculation region. These angled structures potentially indicate the presence of hairpin and counter-hairpin vortices in the shear layer, as the ‘legs’ of these hairpins would be elongated and angled [Sandberg and Fasel, 2006a; Kirchner et al., 2018]. However, as S-PIV is limited to a plane, a full three-dimensional spatial analysis would be required to fully resolve these potential hairpins. Downstream, the structures appear to be more broken apart and rounded. There are a few elongated structures seen in Field 2273 that are oriented with the local flow angle. These small structures correspond to the adverse pressure gradient of the recompression zone breaking apart the larger structures from the upstream portion and the formation of new structures after the adverse pressure gradient. These observations are based on an analysis of only three instantaneous velocity fields, but a more meaningful statistical analysis is required to support these observations.

Figure 3.7 shows the results from the recompression zone, Region III. A large portion of the structures shown are elongated, streamwise-oriented structures. Several of the structures appear to have originated from the recirculation zone and become stretched outside of the zone. This phenomenon is due to the larger streamwise velocity component outside of the recirculation zone and was observed in DNS studies by Sandberg and Fasel [2006a] at lower Reynolds number. Upstream of the stagnation point, the structures appear to be more elongated and oriented at relatively steep angles with the local flow directions. Once again, these structures could indicate the presence of hairpin vortices that require a three-dimensional analysis to fully resolve. Downstream of the stagnation point, the structures become more oriented in the streamwise direction. There is a mix of small, rounded and elongated structures in the downstream portion. The elongated structures that are angled steeply with the streamwise direction could indicate the formation of new hairpin vortices that are introduced from the instabilities of the adverse pressure
gradient. The amplitude of the structures are evenly split between the upstream and downstream portions of the flowfield.

Figure 3.7. The second and third BIMFs of velocity magnitude associated with Region III for the three instantaneous velocity fields.
Figure 3.8 shows the results for the trailing wake, Region IV. The second BIMF, for all fields, shows that a majority of this smaller-scale structure is at an angle with the local flow direction while a few are elongated and streamwise-oriented. The third BIMF shows a mix between rounder structures and larger-scale streamwise-oriented structures. The fourth BIMF results in structures similar to the third BIMF, but are simply larger in scale with some global fluctuations from the freestream interaction. In these three fields, the angled structures appear near the edges of the wake, while the more elongated streamwise-oriented structures occur closer to the centerline. This supports the idea of hairpin-like structures with ‘legs’ extending from the centerline region to ‘heads’ near the edge of the wake. From Region III, the elongated structures...
appear more downstream, further supporting the idea of the legs extending to the recirculation zone [Sandberg and Fasel, 2006a and 2006b]. However, these are only observations from three different instantaneous velocity fields, and no definite conclusions can be drawn without using a statistical analysis, such as linear stochastic estimation.

For completeness of this analysis, the index of orthogonality, $IO$, and reconstruction error, $RE$, are calculated when all the IMFs are extracted. The $IO$ for all regions and all velocity fields is on the order of $10^{-2}$, which is in good agreement with the literature for EEMD-type analysis [Wu et al., 2009 and Bhuiyan et al., 2008] The $RE$ is on the order of $10^{-4}$ percent. The low $IO$ indicates that the IMF sets extracted form a nearly orthogonal basis. The low $RE$ indicates that the signals were not altered significantly by the addition of noise for the ensemble technique, thus forming a complete basis. In general, an alternating pattern of strong maxima followed by similar minima is observed in all BIMFs. This trend indicates a similar number of extrema and zero crossings and that the mean envelope of these BIMFs is close to zero. With all of this in mind, the criteria that define IMFs, described above, appear to be met.

The next quantity to be discussed is the out-of-plane vorticity, $\zeta_z$. For this analysis, the $u$- and $v$-components of velocity are processed with FABEEMD, resulting in two sets of BIMFs. The BIMFs from the velocity components are used to calculate the out-of-plane vorticity BIMFs. For example, the first BIMF from the $u$-component and the first BIMF from the $v$-component are used in conjunction with one another to calculate the first BIMF of vorticity. To avoid potential problems of mode mixing, the same window sizes are enforced when processing both the $u$- and $v$-components of velocity with FABEEMD. The window sizes must be known a priori in order to enforce them, so that the window sizes found during the velocity magnitude analysis are used. This provides a fair representation, so that the window sizes are not biased towards one component over another. For conciseness, only one of the three instantaneous fields, Field 1422, and the BIMFs of interest are shown during this analysis, Figure 3.9.

There are many similar features present when comparing out-of-plane vorticity BIMFs, Figure 3.9, to the corresponding velocity magnitude BIMFs, Figures 3.4-3.8. However, in Region I, the $\zeta_z$ BIMFs contain more transverse-aligned structures, and a few of the structures appear to be more rounded than observed in the velocity magnitude case. In Region II, the $\zeta_z$ BIMFs are less elongated in the streamwise direction. In Region III, there is once again a split between elongated streamwise-oriented structures, rounded, and angled structures. The same trend can be observed,
Figure 3.9. The second and third BIMFs of $\zeta_z$, for field 1422 and for all regions.
as before; the angled structures often appear more upstream while the elongated structures appear downstream towards the trailing wake. Region IV, unlike the other regions, contains the least number of similarities between the ζ and velocity magnitude BIMFs. The ζ BIMFs are much more rounded and angled in the trailing wake. There are almost no streamwise-oriented structures in these BIMFs. Furthermore, there is a strong pairing effect in the third BIMF approximately 4.6 radii downstream of the base, indicating the presence of a vortex.

**Autocorrelation Analysis**

An autocorrelation analysis was used to determine the mean size of the structures in each sub-region. The equation for the autocorrelation analysis with one reference point is listed below. However, this autocorrelation analysis was extended to include multiple reference points. The analysis also requires an ensemble of images to converge; therefore, FABEMED was performed on 300 instantaneous velocity fields for the four different regions for the autocorrelation analysis.

\[
R(\Delta x, \Delta y) = \frac{\frac{1}{n} \sum_{k=1}^{n} I_k'(x_{ref}, y_{ref}) I_k(x_{ref}+\Delta x, y_{ref}+\Delta y)}{I_{rms}(x_{ref}, y_{ref}) I_{rms}(x_{ref}+\Delta x, y_{ref}+\Delta y)}
\]

The reference points for the analysis were chosen by finding the two-dimensional extrema and using the highest three to five maxima and the lowest three to five minima in each component. Figure 3.10 demonstrates how the reference points were chosen on one of the images of BIMF3.

**Figure 3.10.** Example of reference points for the autocorrelation of Region II, BIMF3.
The extrema coincide with several of the main structures observed in the flowfield. If six extrema did not exist in an image, such as for BIMF5, then the maximum number of extrema available were chosen for the analysis.

The results of the autocorrelation analysis for Region II BIMF3 can be seen in Figure 3.11. A color scale for the autocorrelation is presented, although black contour lines separated by 0.1, starting at 0.9 in each case, are added for clarity. Using the autocorrelation analysis, a typical structure size within each component can be extracted. The structure size definition used here is based on the major axis of the 0.5 correlation contour, which is also the definition used by Smith et al. [1999].

![Figure 3.11. Example autocorrelation result of Region II, BIMF3. The red ellipse represents the ellipse fit around the 0.5 contour line to determine the shape properties. The black dots indicate the points the ellipse was fit upon.](image-url)

The 0.5 contour line was fitted with an ellipse for each correlation image to find the structure size. Once the ellipse was fit, the size, orientation and shape of the structures could be characterized. An example of this fitting can also be seen in Figure 3.11. This ellipse fitting has very good agreement with the 0.5 contour line, indicating that the mean structure in this component is indeed elliptical in shape and that the scale size is being correctly captured. However, the ellipse requires five points to be defined completely. Therefore, the first component was ignored in the analysis because it could not be fitted with five points around the 0.5 contour line. The grid
resolution of the second component was also increased by a factor of two to accommodate the required number of points. This increase in grid resolution did not change how well the ellipse could fit the 0.5 contour line of the original data.

A clear separation of scales can be seen from the autocorrelation results of velocity magnitude in Figure 3.12. Looking at major axis length, BIMF 2 contains the smallest scale structure in each of the regions analyzed. Then, as observed from the bar chart, as BIMF number increases the length of the major axis also increases. Region II consistently contains the largest structures between the BIMF numbers and agrees with the instantaneous analysis, as the structures in this region are fairly elongated and streamwise oriented. The eccentricity in this region is also relatively high, suggesting that the structures in the shear layer are more elliptical in shape. The angle of the structures in Region II is also fairly constant, approximately 6-8 degrees relative to the streamwise direction. The structures in Region III appear to be slightly smaller than what is observed in Region IV. This trend corresponds to what has been observed in numerical analyses of this flowfield [Sandberg and Fasel, 2006a, 2006b]. The elongated structures in the shear layer are broken apart by the adverse pressure gradient in the recompression zone; hence the reason Region II has the larger structures compared to Region III. This adverse pressure gradient breaks
the structures apart from the shear layer, but it may also cause new structures to form. The formation of these new structures is a potential cause of the increase in scale size in Region IV when compared to Region III. The eccentricity in Regions III and IV share the same trend, increasing eccentricity with BIMF number. The structures in Region IV appear to be slightly more elliptical than what is observed in Region III except in BIMF 5.

Figure 3.13 displays the results of an autocorrelation performed on an ensemble of out-of-plane vorticity BIMFs for each region. Overall, the trends are the same as observed above in the velocity magnitude results. The sizes of structures observed in \( \zeta_z \) are similar to that observed in velocity magnitude, albeit slightly smaller for the vorticity structures. The structure orientation reveals that the \( \zeta_z \) structures are at a larger angle relative to the streamwise direction. In Region II, the angles are the largest compared to the other Regions. In Region II, the mean shear layer angle is 14.7°, so any angle greater than the mean shear layer angle is being directed towards the recirculation region and any structure angle less than the mean shear layer angle is being directed towards the freestream.

![Graphs showing results of out-of-plane vorticity autocorrelation](image)

**Figure 3.13.** Results of the out-of-plane vorticity autocorrelation results.
**FABEEMD Computational Time**

The last topic of discussion for this two-dimensional study is computational time. Koll et al. [2017] performed a similar analysis using MEEMD, but MEEMD is much more computationally intensive. For comparison, the velocity magnitude IMFs were computed using FABEEMD and MEEMD for each region in Field 291. The ensemble size and the noise amplitude were set to 100 and 0.2, respectively, for each decomposition. The results are shown in Table 1. The same processing computer and coding program, MATLAB, were used to measure these times. The codes were optimized for each technique and parallelized for similar core efficiency. As seen below, there is consistently over a 100 times speedup with FABEEMD compared to MEEMD when processing two-dimensional data.

<table>
<thead>
<tr>
<th>Region</th>
<th>FABEEMD</th>
<th>MEEMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>2.9 s</td>
<td>325 s</td>
</tr>
<tr>
<td>II</td>
<td>2.5 s</td>
<td>284 s</td>
</tr>
<tr>
<td>III</td>
<td>3.5 s</td>
<td>515 s</td>
</tr>
<tr>
<td>IV</td>
<td>2.6 s</td>
<td>500 s</td>
</tr>
</tbody>
</table>

**Table 3.1.** Computational times for Field 291.

**FATEEMD Results of Tomo-PIV Data**

After the successful implementation of FABEEMD on the three-component planar S-PIV data, a three-dimensional code was developed and implemented for three-component volumetric tomo-PIV data. For the upcoming analysis, the following independent variables were processed using FATEEMD separately: \( V_r \), \( V_a \), and \( V_\theta \). This results in a set of TIMFs for each variable. As before in the S-PIV data, the dependent variables are calculated from the corresponding TIMFs of each velocity component. The same window sizes were imposed during the FATEEMD process for each velocity component to avoid any issues with mode mixing when computing the dependent variables. The windows sizes were once again chosen by implementing the FATEEMD analysis on the velocity magnitude to avoid biasing the scales for one component. For this analysis, an ensemble size of 20 and noise amplitude of 0.2 are used to extract the TIMFs. The dependent variables of interest are Q-criterion, \( Q_{cr} \), swirling strength, \( \lambda_{ci} \), and the vorticity magnitude, \( \zeta \).
Tomographic PIV data obtained by Kirchner et al. [2017, 2018] for two different regions of the flowfield are being studied during this analysis. The two regions are similar to Regions II and IV for the S-PIV data. For conciseness, the first TIMF will not be shown for any of the results, since no structural information can be discerned from the measurement noise and synthetic noise, added during the ensemble process, as observed in the S-PIV results. The first region to be studied is the shear layer. The data are obtained such that the volumetric field follows the shear layer angle, so as to minimize the amount of freestream in the velocity field.

Figure 3.14 displays the second BIMFs for each of the dependent variables mentioned above. For each of the dependent variables, a number of streamwise-oriented structures are present approximately 2 radii downstream of the base. However, in general, a majority of the present structures either irregularly shaped or rounded with a few exceptions of hairpin-shaped structures.

**Figure 3.14.** The second TIMFs of the dependent variables for the tomo-PIV shear layer data, a), $\lambda_{ci} = 2$, b), $Q_{cr} = 4.5$, and c), $\zeta = 5$. 

42
While some of these structures are interesting, especially the hairpin-shaped structures, they are of relatively small scale and not a focus of this analysis.

However, these structures increase in size in the third TIMFs, Figure 3.15. Streamwise-oriented structures dominate the results for all of the dependent variables. A few of these structures are angled at relatively steep angles with the local flow direction, ~45°, by observation. These angles agree with the angles of the different hairpin-like structures that Kirchner et al. [2018] found during LSE analysis of the data. A couple hairpin-like structures can be located in the $Q_{cr}$ and $\lambda_{ci}$ TIMFs; however, these structures are relatively small. The presence of numerous longitudinal structures could be a result of the global modes associated with azimuthal modulation near the base of the cylinder as speculated by Sandberg and Fasel [2006a] for lower Reynolds number cases.

**Figure 3.15.** The third TIMFs of the dependent variables for the tomo-PIV shear layer data, a, $\lambda_{ci} = 2$, b, $Q_{cr} = 4$, and c, $\zeta = 5$. 

43
The fourth TIMFs are displayed in Figure 3.16; however, only $\lambda_{ci}$ and $\zeta$ are shown due to the close similarities between $\lambda_{ci}$ and $Q_{cr}$ for the higher-order TIMFs. Once again, the main structures are dominated by the streamwise-oriented features in the swirling strength case. However, there are only a few streamwise-oriented structures in the $\zeta$ TIMF. The reduction of vorticity structures for higher modes is also observed above for the S-PIV analysis. The third BIMF of $\zeta_z$ contains more information about the mean trend and only a few discernible larger-scale structures.

![Figure 3.16](image)

**Figure 3.16.** The fourth TIMFs of the dependent variables for the tomo-PIV shear layer data, a, $\lambda_{ci} = 1$ and b, $\zeta = 3$.

The next data set analyzed with FATEEMD is three-component volumetric velocity field information for the trailing wake obtained with tomo-PIV by Kirchner et al. [2017]. The resulting second, third, and fourth TIMFs for the swirling strength and vorticity magnitude are displayed in Figure 3.17. In this region, the resulting TIMFs of these two dependent variables contain similar length scales and features. As always, the length scales increase with growing TIMF number. However, the streamwise-oriented structures that dominate the TIMFs in the shear layer, are not as prevalent in the trailing wake, except for the fourth TIMF, and one large streamwise feature approximately 3.75 radii downstream of the base in the second and third TIMFs. Instead, the structures are smaller, more rounded, and angled compared to the resulting shear layer TIMFs. A similar trend was observed above in the out-of-plane vorticity results for the S-PIV measurements. The BIMFs associated with Region IV resulted in minimal streamwise-oriented structures with a vast majority of the structures being circular or perpendicularly aligned with the local flow direction. However, these observations are made from only a few instantaneous images, so it would be more conclusive to do a statistical analysis, such as autocorrelation or linear stochastic
Figure 3.17. The resulting second, third, and fourth TIMFs of the dependents variables, $\lambda_{ci}$ (left column) and $\zeta$ (right column), for an instantaneous velocity field in the region of the trailing wake. The values of the dependent variables are listed above the
estimation. However, these kinds of statistical analyses would not have been possible before the implementation of the FATEEMD technique. It takes 20 s to process one component of one volumetric flowfield with FATEEMD and 26,820 s to process the same flowfield with MEEMD, each with an ensemble size of 20. FATEEMD opens up the possibility for these statistical analyses to be performed within a reasonable time frame.

**Linear Stochastic Estimation of TIMFs**

Linear stochastic estimation (LSE) was implemented to statically evaluate the coherence of the turbulent structures. LSE was employed as a way to approximate conditional averages, avoiding the need to conditionally sort the data [Adrian et al., 1989]. The conditional average is approximated in LSE by expanding the conditional average in a power series about an event signature, \( E = 0 \). Only the linear terms from this power series are considered, and the minimization of the mean-squared error between the estimate and actual conditional average is solved for. For example, if the radial velocity component is given by \( V_r \) and the turbulent event signature is given by \( E \), then the linear estimate of the conditional average of \( V_r \), denoted \( \hat{V}_r \), at location \((x,y,z)\) given some \( E \) at the reference location \((x_{ref},y_{ref},z_{ref})\) is:

\[
\hat{V}_r = \langle V_r(x,y,z) | E(x_{ref},y_{ref},z_{ref}) \rangle \approx \frac{\langle V_r(x,y,z) \cdot E(x_{ref},y_{ref},z_{ref}) \rangle}{\langle E(x_{ref},y_{ref},z_{ref})^2 \rangle} E(x_{ref},y_{ref},z_{ref})
\] (3.2)

Hence, the conditional average approximation of \( V_r \), given \( E \), is a function of their unconditional two-point correlation data [Christensen and Adrian, 2001]. Since LSE is a function of \( E \), and \( E \) is simply a scalar value, it is sufficient to specify \( E \) as a non-trivial event, e.g. \( E < 0 \) or \( E > 0 \). If this lies true, then the relative motion of the conditionally averaged velocity will remain the same for any non-zero event and will only differ by a constant scaling factor [Kirchner et al., 2018]. Therefore, it is sufficient to specify \( E = 1 \) or \( E = -1 \) as the event for this LSE analysis. More details of this LSE analysis can be found in Kirchner et al. [2018] as this exact analysis was performed on the velocity fluctuations of the volumetric shear layer data. This LSE analysis will be implemented on the TIMFs resulting from FATEEMD of 916 instantaneous velocity field measurements. Furthermore, only the third and fourth TIMFs will be analyzed using LSE, as they are the most promising TIMFs for locating large-scale, coherent turbulent structures. The event
that the LSE was conditioned on was the signed, two-dimensional, tangentially-oriented swirling strength, $\lambda_{ci,\theta}$, considering only negative events, the same as in Kirchner et al. [2018]. The TIMFs are used here in place of the velocity fluctuations in Kirchner’s LSE analysis, and the resulting conditionally averaged velocity fields are used to compute the swirling strength, $\lambda_{ci}$, for structure identification.

Figure 3.18 demonstrates the results of this LSE procedure for the third TIMFs of the velocity components. The location of this stochastic estimation corresponds to a mean velocity of $|V|/V_{\infty} = 0.8$ (left) and $|V|/V_{\infty} = 0.15$ (right), both upstream of recompression. The stochastic estimation of the point around the high-speed velocity contour reveals a near circular ring of vorticity. On the other hand, the estimation around the low-speed velocity contour reveals the existence of counter-hairpin. The counter-hairpin corresponds to the stochastically estimated structures observed by Kirchner et al. [2018] when the same analysis was performed in this region. The counter-hairpin displayed in Figure 3.18 is relatively unsmooth in geometry. This can be explained by the relatively small number of instantaneous velocity fields used in this analysis, 916, while an ensemble of 2100 instantaneous velocity fields was used in Kirchner et al. [2018].
Figure 3.19 displays the LSE results for the same location as above, when performed with the velocity fluctuations obtained from the fourth TIMF, instead of the third TIMF. The resulting structure along the high-speed \( |V|/V_\infty = 0.8 \) mean contour (left) strongly resembles a conventional hairpin, as observed by Kircher et al. [2018] in this location. The resulting structure along the low-speed \( |V|/V_\infty = 0.15 \) mean contour is rather unusual in shape, and but it somewhat resembles a counter-hairpin as was observed in the results from the third TIMF.

The LSE results from the third and fourth TIMFs suggest that the conventional hairpins found along the high-speed velocity contours are statistically larger in size than the counter-hairpins found along the low-speed velocity contours. With this knowledge, the LSE analysis was performed upon more points along the low- and high-speed velocity contours, but only on their corresponding TIMFs, TIMF 3 and TIMF 4, respectively. These results are displayed in Figure 3.20. When analyzing the results, the furthest two points upstream of recompression on the high-speed contour line result in conventional hairpins. However, upon encountering the adverse pressure gradient, these structures break apart, as observed by the two more downstream points. In fact, the third structure more resembles a cane structure than a full hairpin. This trend was also observed by Kirchner et al. [2018] during a similar analysis. Analyzing the points along the low-speed contour, the overall counter-hairpin shape is maintained even further downstream.
Figure 3.20. Conventional and counter-hairpin structures (depicted by $\lambda_{ci}$) resulting from the LSE analysis along the low- and high-speed contours. The dark blue iso-contours represent the structures found by TIMF 4 and the light blue from TIMF 3.
Chapter 4: Conclusions

Overall, several different variants of empirical mode decomposition were studied in great detail. The variations explored include ensemble EMD and fast and adaptive EMD, as well as its multidimensional extensions. Upon examining the pros and cons, it was decided to combine FAEMD with EEMD in an attempt to reduce the mode mixing that usually occurs in EMD while also trying to drastically reduce the overall computational cost of this technique. As a result, an in-house fast and adaptive bi-dimensional/tri-dimensional ensemble empirical mode decomposition code was created.

The new FABEEMD code was verified by decomposing well-known images of Lena and a Brodatz texture. The resulting modes from the decomposition not only agreed with decompositions of other FABEMD results, but the modes also agreed with BEMD results and MEEMD results of the same images. This demonstrates that the code can avoid common mode mixing problems, while also having a small total computational time when compared to BEMD and especially MEEMD.

After verification, a parameter study was done to explore how the different parameters in the FABEEMD code affect the results. This was done primarily by looking at the index of orthogonality, reconstruction error, and computational time for different combinations of parameters. The parameter study revealed that the amplitude of the noise added during the ensemble process had a strong effect on the index of orthogonality and reconstruction error, but not computational time. Only the ensemble size used during the analysis significantly increased the computational time. The sifting convergence criterion was also examined to determine the effects of over- and under-sifting. The amount of sifting had no apparent effect on the reconstruction error, but there was a significant change in the index of orthogonality; more sifting iterations led to a decrease in the index of orthogonality. However, upon visually examining the results from the different sifting tolerances, the effect of over-sifting could be observed by the strong increase of small fluctuations around the borders of the image features. This suggests that under-sifting is preferable to over-sifting.

With this study in mind, the FABEEMD code was used to analyze the S-PIV velocity field measurements of a supersonic base flow. The flowfield was broken into four different regions for analysis, so that length scales from different regions of the flowfield would not affect one another. The FAEMD analysis showed that elongated, either streamwise-oriented or angled structures are
more prevalent in the recirculation zone, shear layer, and recompression zone, and less prevalent in the trailing wake, especially when analyzing vorticity. These conclusions are supported by the work of Sandberg and Fasel [2006a, 2006b] who found that the instabilities associated with hairpin vortices in the mixing layer lead to the generation of small-scale structures further downstream in the trailing wake. An autocorrelation of these instantaneous results revealed the different length scales associated with the different regions of the flowfield. It also revealed that the mean structures are typically oriented with the streamwise direction, except in the shear layer. In the shear layer, the structures were oriented either upwards, towards the freestream, or downwards, towards the recirculation region, at relatively small angles.

Finally, a three-dimensional, FATEEMD, code was developed and implemented to study the decomposition of volumetric tomo-PIV measurements. The decomposition in the shear layer exposed many longitudinal vortical structures with some smaller-scale, hairpin-like structures present. The decomposition in the trailing wake revealed a significant decrease in the number of longitudinal structures and the presences of smaller-scale structures. A linear stochastic estimation was applied to conditionally average the different structures within the shear layer. The LSE was applied to the third and fourth TIMFs found during the volumetric shear layer analysis. The LSE revealed the presence of conventional hairpin vortices along the high-speed velocity contours and counter-hairpin vortices along the low-speed velocity contours. However, the conventional hairpins were only observed in the larger-scale TIMFs and the counter-hairpins were only observed in the smaller-scale TIMFs, suggesting that conventional hairpins are typically larger in size than the counter-hairpins.
References


Appendix A: Multi-dimensional Ensemble Empirical Mode Decomposition

The original EEMD is limited to only one-dimensional data analysis, and MEEMD was developed to help expand the use of EEMD to multiple spatial dimensions. The method is relatively simple conceptually. First, EEMD is performed on the original data in one dimension. For example, first all the columns of the scalar field would have EEMD performed on them individually. If three IMFs are being extracted, this process would result in four new images: three IMFs and a residual. Then EEMD would be performed on the four new images in the orthogonal direction, i.e., the rows of the four images. This would result in a total of 16 scalar fields. These 16 IMFs are then combined using a combination strategy such that the resulting components have information of similar scale from IMFs in all directions. Figures A1-4 provide a visual demonstration of the image breakdown and combination technique. In Figure A3, the colored lines show how the IMFs are added to obtain the resulting components.

![Figure A1](image1.png)

**Figure A1.** Original two-dimensional scalar field (velocity magnitude) that MEEMD is performed on.

![Figure A2](image2.png)

**Figure A2.** The results of EEMD being performed on all the columns of the scalar field in Figure. A1: three IMFS (a-c) and one residual (d).
When performing the MEEMD analysis, two parameters drive the quality of the resulting components. The first is the amplitude of the white noise added to the signal. This magnitude is determined by multiplying the standard deviation of the signal by a constant. The constant is typically chosen to be less than 0.2 as stated by Wu et al. [2008]. The second parameter is the ensemble size. This parameter is how many times random white noise is added to the signal, the corresponding IMFs are determined, and the ensemble of IMFs are averaged to find the final IMFs. Wu et al. [2008] recommend an ensemble size of 100. A small parametric study was performed.

Figure A3. The resulting 16 IMFs after EEMD has been performed on all the rows of the four IMFs in Figure 3, where the colored lines show the combination strategy for MEEMD.

Figure A4. The result of using the combination strategy to obtain the final images: three components (a-c) and the final residual (d).
herein to determine the effects that these parameters have on the resulting final IMFs. This was done with the objective of minimizing the computational time required for the three-dimensional volumetric tomo-PIV data, which was expected to be significant. Since the S-PIV velocity fields were not computationally demanding, the recommend parameters were used, 0.2 noise coefficient and an ensemble size of 100 [Wu et al., 2008]. When processing the tomo-PIV data, it was decided that keeping the noise coefficient the same, 0.2, while reducing the ensemble size, to 30, would minimize the computational time while retaining the same features. The final results for the example decomposition example can be found in Figure A5.

![Figure A5](image)

**Figure A5.** The first five components of velocity magnitude in Region II (a-e), along with the residual (f), for a typical instantaneous image.
Appendix B: FABEEMD Code (MATLAB)

function [Results] = FABEEMD(Signal,param)
% param.Nesb: Number of noise iterations being used in the ensemble
% param.Nmode: Number of IMFs to be extracted
% param.Namp: Amplitude of the noise being added to the signal
% param.Type: Window Size type, Standard types 1-6, 7 to specify windows
% param.window: If type 7, enter the specified windows as a vector
% param.tol: Sifting tolerance

tic
[Nr,Nc] = size(Signal);
Signal_rms = rms(rms(Signal));
BIMF_esb = zeros(Nr,Nc,param.Nmode,param.Nesb);
Res_esb = zeros(Nr,Nc,param.Nesb);
sift_count = zeros(param.Nmode,param.Nesb);
Windows_mat = zeros(6,param.Nmode,param.Nesb);
parfor esb = 1:param.Nesb
    noise = randn(Nr,Nc);
    NoiseySignal = Signal + Signal_rms*param.Namp*noise;
    BIMF = zeros(Nr,Nc,param.Nmode);
    temp_win = zeros(6,param.Nmode);
    sifts = zeros(param.Nmode,1);
    for mode = 1:param.Nmode
        A = NoiseySignal;
        if param.Type == 7
            w_size = param.Window(mode);
        else
            Windows = sizing(A);
            w_size = Windows(param.Type);
        end
        Flag = 0;
        sift = 0;
        while Flag == 0
            A_old = A;
            Env = orderstatfilt(A,w_size);
            Env = padding(Env,w_size);
            Env = smoothing(Env,w_size);
            Env.mean = (Env.smooth.max + Env.smooth.min)/2;
            A = A - Env.mean;

            if sift == 0
                std_check = immse(A,A_old);
                if std_check < param.tol
                    Flag = 1;
                end
            end
            sift = sift + 1;
        end
        if param.Type ~=7
            temp_win(:,mode) = Windows;
        end
        sifts(mode) = sift-1;
        BIMF(:,mode) = A;
        NoiseySignal = NoiseySignal - A;
    end
    if param.Type ~=7

59
Windows_mat(:,:,esb) = temp_win;
end
sift_count(:,esb) = sifts;
BIMF_esb(:,:,esb) = BIMF;
Res_esb(:,:,esb) = NoiseySignal;
end
Results.BIMF = mean(BIMF_esb,4);
Results.RESD = mean(Res_esb,3);
Results.Time = toc;
[Results.IO, Results.Error] = IOandError(Results.BIMF,Results.RESD,Signal);
if param.Type ~= 7
    Results.Windows = Windows_mat;
end
Results.sifts = sift_count;
end

function Windows = sizing(A)
    [Nx,Ny] = size(A);
    [LMMAX, LMMIN] = extrema2d(A);
    I_max = isfinite(LMMAX);
    I_min = isfinite(LMMIN);
    I_max = find(I_max);
    I_min = find(I_min);
    n_max = length(I_max);
    n_min = length(I_min);
    [x_max,y_max] = ind2sub([Nx,Ny],I_max);
    tri_max = delaunay(x_max,y_max);
    [x_min,y_min] = ind2sub([Nx,Ny],I_min);
    tri_min = delaunay(x_min,y_min);
    min_neigh = zeros(length(I_min),1);
    max_neigh = zeros(length(I_max),1);
    for i = 1:length(tri_max(:,1))
        for j = 1:2
            for k = j+1:3
                max_distance(j,k-1) = sqrt((x_max(tri_max(i,k)) - x_max(tri_max(i,j)))^2 + (y_max(tri_max(i,k)) - y_max(tri_max(i,j)))^2);
                max_distance(k,j) = max_distance(j,k-1);
            end
        end
    end
    min_vec = min(max_distance,[],2);
    for j = 1:3
        if max_neigh(tri_max(i,j)) > min_vec(j) || max_neigh(tri_max(i,j)) == 0
            max_neigh(tri_max(i,j)) = min_vec(j);
        end
    end
end
for i = 1:length(tri_min(:,1))
    for j = 1:2
        for k = j+1:3
            max_distance(j,k-1) = sqrt((x_min(tri_min(i,k)) - x_min(tri_min(i,j)))^2 + (y_min(tri_min(i,k)) - y_min(tri_min(i,j)))^2);
            max_distance(k,j) = max_distance(j,k-1);
        end
    end
    min_vec = min(max_distance,[],2);
    for j = 1:3
        if max_neigh(tri_min(i,j)) > min_vec(j) || max_neigh(tri_min(i,j)) == 0
            max_neigh(tri_min(i,j)) = min_vec(j);
        end
    end
end
for i = 1:length(tri_min(:,1))
    for j = 1:2
for k = j+1:3
    min_distance(j,k-1) = sqrt...
    (x_min(tri_min(i,k)) - x_min(tri_min(i,j)))^2 ...
    + (y_min(tri_min(i,k)) - y_min(tri_min(i,j)))^2);
    min_distance(k,j) = min_distance(j,k-1);
end
end
min_vec = min(min_distance,[],2);
for j = 1:3
    if min_neigh(tri_min(i,j)) > min_vec(j) || min_neigh(tri_min(i,j)) == 0
        min_neigh(tri_min(i,j)) = min_vec(j);
    end
end
Type_1 = min([min(max_neigh),min(min_neigh)]);
Type_2 = max([min(max_neigh),min(min_neigh)]);
Type_3 = min([max(max_neigh),max(min_neigh)]);
Type_4 = max([max(max_neigh),max(min_neigh)]);
Type_5 = mean([Type_1, Type_2, Type_3, Type_4]);
Type_6 = sqrt(Nx*Ny/(n_max + n_min));
Type_matrix = [Type_1; Type_2; Type_3; Type_4; Type_5; Type_6];
end
function [max,min] = extrema2d(A)
[Nx,Ny] = size(A);
max = NaN(Nx,Ny);
min = NaN(Nx,Ny);
for i = 1:Nx
    for j = 1:Ny
        if i == 1
            if j == 1
                if A(i,j) > A(i+1,j) && A(i,j) > A(i,j+1) && A(i,j) > A(i+1,j+1)
                    max(i,j) = A(i,j);
                end
                if A(i,j) < A(i+1,j) && A(i,j) < A(i,j+1) && A(i,j) < A(i+1,j+1)
                    min(i,j) = A(i,j);
                end
            end
            if j == Ny
                if A(i,j) > A(i+1,j) && A(i,j) > A(i,j-1) && A(i,j) > A(i+1,j-1)
                    max(i,j) = A(i,j);
                end
                if A(i,j) < A(i+1,j) && A(i,j) < A(i,j-1) && A(i,j) < A(i+1,j-1)
                    min(i,j) = A(i,j);
                end
            end
        end
        if j > 1 && j < Ny
            if A(i,j) > A(i,j+1) && A(i,j) > A(i+1,j+1) && A(i,j) > A(i+1,j)...
                && A(i,j) > A(i,j-1) && A(i,j) > A(i+1,j-1)
                max(i,j) = A(i,j);
            end
        end
    end
end
if A(i,j) < A(i,j+1) && A(i,j) < A(i+1,j+1) && A(i,j) < A(i+1,j)
    min(i,j) = A(i,j);
end
end
end
if j == 1
    if i > 1 && i < Nx
        if A(i,j) > A(i-1,j+1) && A(i,j) > A(i,j+1) && A(i,j) > A(i+1,j+1)
            max(i,j) = A(i,j);
        end
        if A(i,j) < A(i-1,j+1) && A(i,j) < A(i,j+1) && A(i,j) < A(i+1,j+1)
            min(i,j) = A(i,j);
        end
    end
end
end
if i > 1 && i < Nx && j > 1 && j < Ny
    if A(i,j) > A(i-1,j+1) && A(i,j) > A(i,j+1) && A(i,j) > A(i+1,j+1)
        max(i,j) = A(i,j);
    end
    if A(i,j) < A(i-1,j+1) && A(i,j) < A(i,j+1) && A(i,j) < A(i+1,j+1)
        min(i,j) = A(i,j);
    end
end
end
if i == Nx
    if j == 1
        if A(i,j) > A(i-1,j) && A(i,j) > A(i,j+1) && A(i,j) > A(i-1,j+1)
            max(i,j) = A(i,j);
        end
        if A(i,j) < A(i-1,j) && A(i,j) < A(i,j+1) && A(i,j) < A(i-1,j+1)
            min(i,j) = A(i,j);
        end
    end
end
end
if j == Ny
    if A(i,j) > A(i-1,j) && A(i,j) > A(i,j-1) && A(i,j) > A(i-1,j-1)
        max(i,j) = A(i,j);
    end
    if A(i,j) < A(i-1,j) && A(i,j) < A(i,j-1) && A(i,j) < A(i-1,j-1)
        min(i,j) = A(i,j);
    end
end
end
if j > 1 && j < Ny
    if A(i,j) > A(i,j+1) && A(i,j) > A(i-1,j+1) && A(i,j) > A(i-1,j)
        max(i,j) = A(i,j);
    end
    if A(i,j) < A(i,j+1) && A(i,j) < A(i-1,j+1) && A(i,j) < A(i-1,j)
        min(i,j) = A(i,j);
end
end
end
end
if j == Ny
  if i > 1 && i < Nx
    if A(i,j) > A(i-1,j-1) && A(i,j) > A(i-1,j) && A(i,j) > A(i-1,j)
                  && A(i,j) < A(i-1,j-1) && A(i,j) < A(i-1,j) && A(i,j) < A(i+1,j)
      max(i,j) = A(i,j);
    end
    if A(i,j) < A(i-1,j-1) && A(i,j) < A(i-1,j) && A(i,j) < A(i+1,j)
                  && A(i,j) < A(i-1,j) && A(i,j) < A(i+1,j)
      min(i,j) = A(i,j);
    end
  end
end
end
end
end
end
function Env = orderstatfilt(A,w_size)
  Env.filt.max = ordfilt2(A,w_size^2,true(w_size),'symmetric');
  Env.filt.min = ordfilt2(A,1,true(w_size),'symmetric');
end
function Env = padding(Env,w_size)
  h = floor(w_size/2);
  Env.padded.max = padarray(Env.filt.max,[h h],'replicate');
  Env.padded.min = padarray(Env.filt.min,[h h],'replicate');
end
function Env = smoothing(Env,w_size)
  h = floor(w_size/2);
  l = ceil(w_size/2);
  temp = movmean(Env.padded.max,w_size,2,'endpoints','discard');
  Env.smooth.max = movmean(temp,w_size,1,'endpoints','discard');
  temp = movmean(Env.padded.min,w_size,2,'endpoints','discard');
  Env.smooth.min = movmean(temp,w_size,1,'endpoints','discard');
end
function [IO, Error] = IOandError(BIMFS,Residual,Signal)
  I = sum(BIMFS,3) + Residual;
  Error.map = (Signal-I)./Signal;
  Error.global = immse(I,Signal);
  % Error.freestream = mean(mean(Error.map(1:5,95:100)))*100;
  [Nr,Nc,Nmode] = size(BIMFS);
  temp = zeros(Nr,Nc);
  for i = 1:Nmode-1
    for j = i:Nmode
      temp = temp + (BIMFS(:,:,i).*BIMFS(:,:,j))/sum(sum((I.^2)));
    end
  end
end
IO.map = temp;
IO.global = sum(sum(temp));
end
Appendix C: FATEEMD Code (MATLAB)

```matlab
function [Results] = FATEEMD(Signal,param)
    % param.Nesb: Number of noise ensembles being used
    % param.Nmode: Number of IMFs to be extracted
    % param.Namp: Amplitude of the noise being added to the signal
    % param.Type: Window Size type, Standard types 1-6, 7 to specify windows
    % param.Window: If type 7, enter the specified windows as a vector
    Namp = param.Namp;
    Nmode = param.Nmode;
    Nesb = param.Nesb;
    Type = param.Type;

    [Nx,Ny,Nz] = size(Signal);
    Signal_rms = rms(rms(rms(Signal)));
    Sift_m = zeros(Nmode,Nesb);
    IMF = zeros(Nx,Ny,Nz,Nmode,Nesb);

    parfor esb = 1:param.Nesb
        noise = Signal_rms*randn(Nx,Ny,Nz)*Namp;
        Signal_n = Signal + noise;
        for mode = 1:Nmode
            A = Signal_n;
            Flag = 0;
            sift = 0;
            if Type ~= 7
                Windows = sizing(A,Type);
                w_size = Windows(Type);
                disp(w_size)
            else
                w_size = param.Window(mode);
            end
            temp = 0;
            while Flag == 0
                A_old = A;
                [Env] = filt(A,w_size);
                [Env] = padding(Env,w_size);
                [Env] = smoothing(Env,w_size);
                LMMEAN = (Env.LMMAX.Smooth + Env.LMMIN.Smooth)./2;
                A = A - LMMEAN;
                if sift ~= 0
                    Check = immse(A,A_old);
                    if Check < param.tol
                        Flag = 1;
                        temp = sift;
                    end
                end
                sift = sift + 1;
            end
            Sift_m(mode,esb) = temp;
            IMF(:,:,,:,mode,esb) = A;
            Signal_n = Signal_n - IMF(:,:,,:,mode,esb);
        end
    end
    resd(:,:,,:,esb) = Signal_n;
end
```
True_IMFS = mean(IMF,5);
Mean_resd = mean(resd,4);
TIMFS = True_IMFS;
TIMFS(:,:,param.Nmode+1) = Mean_resd;
Results.TIMFS = TIMFS;
Results.TIMFS(:,:,Nmode+1) = Mean_resd;

Results.Sift = Sift_m;
end

function [Windows] = sizing(A,Type)
    [~,I_max,~,I_min] = MinimaMaxima3D(A,1,1);
    n = length(I_max(:,1)) + length(I_min(:,1));
    [Nx,Ny,Nz] = size(A);

    if Type ~= 6
        x_max = I_max(:,1);
        y_max = I_max(:,2);
        z_max = I_max(:,3);
        % [x_max,y_max,z_max] = ind2sub([Nx,Ny,Nz],I_max);
        tri_max = delaunay(x_max,y_max,z_max);
        % [x_min,y_min,z_min] = ind2sub([Nx,Ny,Nz],I_min);
        x_min = I_min(:,1);
        y_min = I_min(:,2);
        z_min = I_min(:,3);
        tri_min = delaunay(x_min,y_min,z_min);

        min_neigh = zeros(length(I_min(:,1)),1);
        max_neigh = zeros(length(I_max(:,1)),1);
        for i = 1:length(tri_max(:,1))
            for j = 1:3
                for k = j+1:4
                    max_distance(j,k-1) = sqrt((x_max(tri_max(i,k)) - x_max(tri_max(i,j)))^2 ...
                                                + (y_max(tri_max(i,k)) - y_max(tri_max(i,j)))^2 ...
                                                + (z_max(tri_max(i,k)) - z_max(tri_max(i,j)))^2);
                    max_distance(k,j) = max_distance(j,k-1);
                end
            end
        end
        min_vec = min(max_distance,[],2);
        for j = 1:4
            if max_neigh(tri_max(i,j)) > min_vec(j) || max_neigh(tri_max(i,j)) == 0
                max_neigh(tri_max(i,j)) = min_vec(j);
            end
        end
    end

    for i = 1:length(tri_min(:,1))
        for j = 1:3
            for k = j+1:4
                min_distance(j,k-1) = sqrt((x_min(tri_min(i,k)) - x_min(tri_min(i,j)))^2 ...
                                               + (y_min(tri_min(i,k)) - y_min(tri_min(i,j)))^2 ...
                                               + (z_min(tri_min(i,k)) - z_min(tri_min(i,j)))^2);
            end
        end
    end
end

\[ \text{min}\_\text{distance}(k,j) = \text{min}\_\text{distance}(j,k-1); \]
\[
\text{end}
\]
\[
\text{end}
\]
\[
\text{min}\_\text{vec} = \text{min}(\text{min}\_\text{distance},[],2);
\]
\[
\text{for } j = 1:4 \\
\quad \text{if } \text{min}\_\text{neigh}(\text{tri}\_\text{min}(i,j)) > \text{min}\_\text{vec}(j) || \text{min}\_\text{neigh}(\text{tri}\_\text{min}(i,j)) == 0 \\
\quad \text{min}\_\text{neigh}(\text{tri}\_\text{min}(i,j)) = \text{min}\_\text{vec}(j);
\quad \text{end}
\quad \text{end}
\]
\[
\text{Type1} = \text{min}([\text{min}\_\text{neigh}, \text{min}\_\text{max}\_\text{neigh}]);
\]
\[
\text{Type2} = \text{max}([\text{min}\_\text{neigh}, \text{min}\_\text{max}\_\text{neigh}]);
\]
\[
\text{Type3} = \text{min}([\text{max}\_\text{neigh}, \text{max}\_\text{max}\_\text{neigh}]);
\]
\[
\text{Type4} = \text{max}([\text{max}\_\text{neigh}, \text{max}\_\text{max}\_\text{neigh}]);
\]
\[
\text{Type5} = \text{mean}([\text{Type1} \text{Type2} \text{Type3} \text{Type4}])); \quad \% \text{mean}
\]
\[
\text{Type6} = (\text{mean(Nx*Ny*Nz)/n})^{1/3};
\]
\[
\text{Windows} = [\text{Type1} \text{Type2} \text{Type3} \text{Type4} \text{Type5} \text{Type6}];
\]
\[
\text{Windows} = \text{round}((\text{Windows} + 1)/2).*2 - 1;
\]
\[
\text{else}
\quad \text{Type6} = (\text{mean(Nx*Ny*Nz)/n})^{1/3};
\quad \text{Windows} = [0 0 0 0 0 \text{Type6}];
\quad \text{Windows} = \text{round}((\text{Windows} + 1)/2).*2 - 1;
\quad \text{end}
\]
\[
\text{function}
\[
[\text{Maxima,MaxPos,Minima,MinPos}]=\text{MinimaMaxima3D(Input,Robust,LookInBoundaries,numbermax,numbermin)}
\]
\[
\% \text{V 1.0 Dec 13, 07}
\%
\% Author Sam Pichardo.
\% This function finds the local minima and maxima in a 3D Cartesian data.
\% It's assumed that the data is uniformly distributed.
\% The minima and maxima are calculated using a multi-directional derivation.
\%
\% Use:
\%
\% [Maxima,MaxPos,Minima,MinPos]=MinimaMaxima3D(Input,[Robust],[LookInBoundaries],[numbermax],[numbermin])
\%
\% where Input is the 3D data and Robust (optional and with a default value
\% of 1) indicates if the multi-directional derivation should include the
\% diagonal derivations.
\%
\% Input has to have a size larger or equal than [3 x 3 x 3]
\%
\% If Robust=1, the total number of derivations taken into account are 26: 6
\% for all surrounding elements colliding each of the faces of the unit cube;
\% 10 for all the surrounding elements in diagonal.
\%
\% If Robust =0, then only the 6 elements of the colliding faces are considered
\%
\% The function returns in Maxima and MaxPos, respectively,
\% the values (numbermax) and subindexes (numbermax x 3) of local maxima
\% and position in Input. Maxima (and the subindexes) are sorted in
\% descending order.
% Similar situation for Minima and MinimaPos with a numbermin elements but
% with the exception of being sorted in ascending order.
% IMPORTANT: if numbermin or numbermax are not specified, ALL the minima
% or maxima will be returned. This can be a useless for highly
% oscillating data
%
% LookInBoundaries (default value of 0) specifies if a search of the minima/maxima should be
% done in the boundaries of the matrix. This situation depends on the
% the desire application. When it is not activated, the algorithm WILL NOT
% FIND ANY MINIMA/MAXIMA on the 6 layers of the boundaries.
% When it is activated, the finding minima and maxima on the boundaries is done by
% replicating the extra layer as the layer 2 (or layer N-1, depending of the boundary)
% By example (and using a 2D matrix for simplicity reasons):
% For the matrix
% [ 4 1 3 7
%   5 7 8 8
%   9 9 9 9
%   5 6 7 9]
% the calculation of the partial derivate following the -x direction will be done by substrascting
% [ 5 7 8 8
%   4 1 3 7
%   5 7 8 8
%   9 9 9 9]
% to the input. And so on for the other dimensions.
% Like this, the value "1" at the coordinate (1,2) will be detected as a
% minima. Same situation for the value "5" at the coordinate (4,1)

if nargin <1
    test=load('temp.mat');
    pf=test.uresTot(test.EvalLims(2,1):test.EvalLims(2,2));
    pf=reshape(pf,length(test.EvalCoord{2}.Ry),length(test.EvalCoord{2}.Rx),length(test.EvalCoord{2}.Rz));
    Input = abs(pf)*1.5e6;
    clear test;
    clear pf;
    Robust =1;
end

Asize=size(Input);

if length(Asize)<3
    error('MinimaMaxima3D can only works with 3D matrices ');
end

if (Asize(1)<3 || Asize(2)<3 || Asize(3)<3)
    error('MinimaMaxima3D can only works with matrices with dimensions equal or larger to [3x3x3]');
end

if ~isreal(Input)
    warning('ATTENTION, complex values detected!!, using abs(Input)');
    Input=abs(Input);
end
if ~exist('Robust','var')
    Robust=1;
end

if ~exist('LookInBoundaries','var')
    LookInBoundaries=0;
end

if ~exist('numbermax','var')
    numbermax=0;
end

if ~exist('numbermin','var')
    numbermin=0;
end

[xx_base,yy_base,zz_base]=ndgrid(1:Asize(1),1:Asize(2),1:Asize(3));

IndBase=sub2ind(Asize,xx_base(:),yy_base(:),zz_base(:));

if Robust ~= 0
    Number_dd=26;
else
    Number_dd=6;
end

if LookInBoundaries==0
    lx=1:Asize(1);
    lx_p1=[2:Asize(1),Asize(1)];
    lx_m1=[1,1:Asize(1)-1];
    ly=1:Asize(2);
    ly_p1=[2:Asize(2),Asize(2)];
    ly_m1=[1,1:Asize(2)-1];
    lz=1:Asize(3);
    lz_p1=[2:Asize(3),Asize(3)];
    lz_m1=[1,1:Asize(3)-1];
else
    lx=1:Asize(1);
    lx_p1=[2:Asize(1),Asize(1)-1]; %We replicate the layer N-1 as the layer N+1
    lx_m1=[2,1:Asize(1)-1]; %We replicate the layer 2 as the layer -1
    ly=1:Asize(2);
    ly_p1=[2:Asize(2),Asize(2)-1]; %We replicate the layer N-1 as the layer N+1
    ly_m1=[2,1:Asize(2)-1]; %We replicate the layer 2 as the layer -1
    lz=1:Asize(3);
    lz_p1=[2:Asize(3),Asize(3)-1]; %We replicate the layer N-1 as the layer N+1
    lz_m1=[2,1:Asize(3)-1]; %We replicate the layer 2 as the layer -1
end

for n_dd=1:Number_dd
    switch n_dd
    case 1
        This index is used to calculate elem(x)-elem(x+1)
        [xx,yy,zz]=ndgrid(lx_p1,ly,lz);
end
case 2
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(x-1)
[xx,yy,zz]=ndgrid(lx_m1,ly,lz);

case 3
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(y)-elem(y+1)
[xx,yy,zz]=ndgrid(lx,ly_p1,lz);

case 4
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(y)-elem(y-1)
[xx,yy,zz]=ndgrid(lx,ly_m1,lz);

case 5
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(z)-elem(z+1)
[xx,yy,zz]=ndgrid(lx,ly_lz_p1);

case 6
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(z)-elem(z-1)
[xx,yy,zz]=ndgrid(lx,ly_lz_m1);

case 7
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(x+1,y+1)
[xx,yy,zz]=ndgrid(lx_p1,ly_p1,lz);

case 8
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(x+1,y-1)
[xx,yy,zz]=ndgrid(lx_p1,ly_m1,lz);

case 9
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(x-1,y-1)
[xx,yy,zz]=ndgrid(lx_m1,ly_m1,lz);

case 10
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(x-1,y+1)
[xx,yy,zz]=ndgrid(lx_m1,ly_p1,lz);

case 11
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(x+1,z+1)
[xx,yy,zz]=ndgrid(lx_p1,ly,lz_p1);

case 12
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(x+1,z-1)
[xx,yy,zz]=ndgrid(lx_p1,ly,lz_m1);

case 13
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(x-1,z-1)
[xx,yy,zz]=ndgrid(lx_m1,ly,lz_m1);

case 14
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(x-1,z+1)
[xx,yy,zz]=ndgrid(lx_m1,ly,lz_p1);

case 15
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(y+1,z+1)
[xx,yy,zz]=ndgrid(lx,ly_p1,lz_p1);

case 16
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(y+1,z-1)
[xx,yy,zz]=ndgrid(lx,ly_p1,lz_m1);

case 17
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(y-1,z-1)
[xx,yy,zz]=ndgrid(lx,ly_m1,lz_m1);

case 18
%%%%%%%%%%%%%%%%%% %% This index is used to calculated elem(x)-elem(y-1,z+1)
\[
[xx,yy,zz]=\text{ndgrid}(lx,ly_m1,lz_p1);
\]

\textbf{case 19}

\begin{verbatim}
\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\% This index is used to calculated elem(x)-elem(x+1,y+1,z+1) 
[xx,yy,zz]=\text{ndgrid}(lx_p1,ly_p1,lz_p1);
\end{verbatim}

\textbf{case 20}

\begin{verbatim}
\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\% This index is used to calculated elem(x)-elem(x+1,y+1,z-1) 
[xx,yy,zz]=\text{ndgrid}(lx_p1,ly_p1,lz_m1);
\end{verbatim}

\textbf{case 21}

\begin{verbatim}
\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\% This index is used to calculated elem(x)-elem(x+1,y+1,z+1) 
[xx,yy,zz]=\text{ndgrid}(lx_p1,ly_m1,lz_p1);
\end{verbatim}

\textbf{case 22}

\begin{verbatim}
\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\% This index is used to calculated elem(x)-elem(x+1,y-1,z-1) 
[xx,yy,zz]=\text{ndgrid}(lx_p1,ly_m1,lz_m1);
\end{verbatim}

\textbf{case 23}

\begin{verbatim}
\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\% This index is used to calculated elem(x)-elem(x-1,y+1,z+1) 
[xx,yy,zz]=\text{ndgrid}(lx_m1,ly_p1,lz_p1);
\end{verbatim}

\textbf{case 24}

\begin{verbatim}
\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\% This index is used to calculated elem(x)-elem(x-1,y+1,z-1) 
[xx,yy,zz]=\text{ndgrid}(lx_m1,ly_p1,lz_m1);
\end{verbatim}

\textbf{case 25}

\begin{verbatim}
\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\% This index is used to calculated elem(x)-elem(x-1,y-1,z+1) 
[xx,yy,zz]=\text{ndgrid}(lx_m1,ly_m1,lz_p1);
\end{verbatim}

\textbf{case 26}

\begin{verbatim}
\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\% This index is used to calculated elem(x)-elem(x-1,y-1,z-1) 
[xx,yy,zz]=\text{ndgrid}(lx_m1,ly_m1,lz_m1);
\end{verbatim}

end

\text{Ind_dd}=\text{sub2ind}(Asize,xx(:,),yy(:,),zz(,));

partderiv = \text{Input(IndBase)}-\text{Input(Ind_dd)};

if n_dd >1
  \text{MatMinMax}= (\text{sign_Prev_deriv==sign(partderiv))}\times\text{MatMinMax};
else
  \text{MatMinMax}=\text{sign(part_deriv)};
end

\text{sign_Prev_deriv}=\text{sign(part_deriv)};
end

\%Well, now the easy part, all values MatMinMax ==1 are local maximum and 
\%the values MatMinMax ==-1 are minimum

\text{AllMaxima}=\text{find(MatMinMax==1)};
\text{AllMinima}=\text{find(MatMinMax==1)};

\text{if numbermax ==0}
  \text{nmax=\text{length(AllMaxima)}};
\text{else}
  \text{nmax=numbermax};
\text{end}
\text{nmax=min([nmax,\text{length(AllMaxima)}])};
\text{smax=1:nmax};
if numbermin == 0
    nmin=length(AllMinima);
else
    nmin=numbermin;
end

nmin=min([nmin,length(AllMinima)]);

smin=1:nmin;

[Maxima,IndMax]=sort(Input(AllMaxima), 'descend');
Maxima=Maxima(smax);
IndMax=AllMaxima(IndMax(smax));

MaxPos=zeros(nmax,3);
[MaxPos(:,1),MaxPos(:,2),MaxPos(:,3)]=ind2sub(Asize,IndMax);

[Minima,IndMin]=sort(Input(AllMinima));
Minima=Minima(smin);
IndMin=AllMinima(IndMin(smin));

MinPos=zeros(nmin,3);
[MinPos(:,1),MinPos(:,2),MinPos(:,3)]=ind2sub(Asize,IndMin);
end

function [Env] = filt(A,w_size)
    [Nr,Nc,Nz] = size(A);
    for k = 1:Nz
        LMMINr(:,:,k) = ordfilt2(A(:,:,k),1,ones(w_size,1), 'symmetric');
        LMMINc(:,:,k) = ordfilt2(LMMINr(:,:,k),1,ones(1,w_size), 'symmetric');
        LMMAXr(:,:,k) = ordfilt2(A(:,:,k),w_size,ones(w_size,1), 'symmetric');
        LMMAXc(:,:,k) = ordfilt2(LMMAXr(:,:,k),w_size,ones(1,w_size), 'symmetric');
    end
    for i = 1:Nr
        temp(1:Nc,1:Nz) = LMMINc(i,:);
        LMMINz(i,1:Nc,1:Nz) = ordfilt2(temp,1,ones(1,w_size), 'symmetric');
        temp(1:Nc,1:Nz) = LMMAXc(i,:);
        LMMAXz(i,1:Nc,1:Nz) = ordfilt2(temp,w_size,ones(1,w_size), 'symmetric');
    end
    Env.LMMAX.Filt = LMMAXz;
    Env.LMMIN.Filt = LMMINz;
end

function Env = padding(Env,w_size)
    b = floor(w_size/2);
    Env.LMMAX.Pad = padarray(Env.LMMAX.Filt,[b b b], 'replicate');
    Env.LMMIN.Pad = padarray(Env.LMMIN.Filt,[b b b], 'replicate');
end

function Env = smoothing(Env,w_size)
    h = ceil(w_size/2);

b = floor(w_size/2);

temp = movmean(Env.LMMAX.Pad,[b b],1,'omitnan','Endpoints','discard');
temp = movmean(temp,[b b],2,'omitnan','Endpoints','discard');
Env.LMMAX.Smooth = movmean(temp,[b b],3,'omitnan','Endpoints','discard');

% Env.LMMAX.Smooth = temp(h:end-b,h:end-b,h:end-b);

temp = movmean(Env.LMMIN.Pad,[b b],1,'omitnan','Endpoints','discard');
temp = movmean(temp,[b b],2,'omitnan','Endpoints','discard');
Env.LMMIN.Smooth = movmean(temp,[b b],3,'omitnan','Endpoints','discard');

% Env.LMMAX.Smooth = temp(h:end-b,h:end-b,h:end-b);
end