

5-1-2020

## Discrete Vortex Modeling of Aerodynamic Flutter of a Flat Plate with Damped Oscillations

Emma Chao

Follow this and additional works at: <https://digitalscholarship.unlv.edu/thesesdissertations>



Part of the [Aerospace Engineering Commons](#), and the [Mechanical Engineering Commons](#)

---

### Repository Citation

Chao, Emma, "Discrete Vortex Modeling of Aerodynamic Flutter of a Flat Plate with Damped Oscillations" (2020). *UNLV Theses, Dissertations, Professional Papers, and Capstones*. 3874.  
<http://dx.doi.org/10.34917/19412036>

This Thesis is protected by copyright and/or related rights. It has been brought to you by Digital Scholarship@UNLV with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis has been accepted for inclusion in UNLV Theses, Dissertations, Professional Papers, and Capstones by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact [digitalscholarship@unlv.edu](mailto:digitalscholarship@unlv.edu).

DISCRETE VORTEX MODELING OF AERODYNAMIC FLUTTER OF A FLAT PLATE WITH  
DAMPED OSCILLATIONS

by

Emma Chao

Bachelor of Science in Engineering – Mechanical Engineering  
University of Nevada, Las Vegas  
2019

A thesis submitted in partial fulfillment  
of the requirements for the

Master of Science in Engineering – Mechanical Engineering

Department of Mechanical Engineering  
Howard R. Hughes College of Engineering  
The Graduate College

University of Nevada, Las Vegas  
May 2020



## Thesis Approval

The Graduate College  
The University of Nevada, Las Vegas

February 24, 2020

This thesis prepared by

Emma Chao

entitled

Discrete Vortex Modeling of Aerodynamic Flutter of a Flat Plate with Damped Oscillations

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Engineering – Mechanical Engineering  
Department of Mechanical Engineering

William Culbreth, Ph.D.  
*Examination Committee Chair*

Kathryn Hausbeck Korgan, Ph.D.  
*Graduate College Dean*

Woosoon Yim, Ph.D.  
*Examination Committee Member*

Mohamed Trabia, Ph.D.  
*Examination Committee Member*

Evangelos Yfantis, Ph.D.  
*Graduate College Faculty Representative*

## ABSTRACT

Aerodynamic flutter is the unstable oscillation of a body caused by the interaction of aerodynamic forces, structural elasticity, and inertial effects induced by vortex shedding. Current models of this phenomenon require finite element analysis and extensive computational power and processing time. The purpose of this study was to develop and validate a program that is faster and more efficient than existing approaches by using the discrete vortex method (DVM). By reducing the complexities of flutter to the shedding of vortices in an inviscid model of a two-dimensional flat plate with a torsional spring constant at its center, this phenomenon can be modeled for a demonstration. A discrete vortex model of inviscid flow past a cylinder is transformed through conformal mappings to model the behavior of a flat plate in an impulsively started streamline flow. Discrete vortices are shed at the tips of the plate and the moment induced by the vortices on the flat plate result in its angular displacement varying over time to simulate flutter. The results of this study provide a rudimentary example of the potential benefit of implementing DVM in the field of structural dynamics.

A FreeBASIC program was developed for this thesis and utilized in several parametric studies. The program includes an iterative time loop in which discrete vortices are added to the flow field and the angle of the flat plate is updated in every time step. The program is fast enough that it runs in almost real time though it slows down as more vortices are added into the field. The resulting flow field displays like a video which allows the viewer to visually observe how a plate of user-defined properties will behave in a uniform flow. Furthermore, the user can observe the force and pressure distributions on the plate in the same window to visualize the aerodynamic forces acting on the plate due to the wake. Finally, the deflection angle, drag force, and pressure at a specified probe location in the wake are calculated in each time step and recorded in a text file; for data analysis, these values can be extracted and plotted with respect to time.

Throughout development, the program was validated for realistic results by plotting inviscid streamlines, observing the wake generated by discrete vortices, and analyzing the flutter simulation results. Once the program development was complete, the parameters varied in this study included plate width, uniform velocity, mass moment of inertia, damping coefficient, and torsional stiffness. By performing Fast

Fourier Transforms on the deflection angle and pressure data, the damped frequency and the Strouhal number of the wake, respectively, were approximated for each case simulated in the program. The correspondence between the predicted and actual values for damped frequencies was extremely accurate. However, the Strouhal numbers were less conclusive as they were difficult to extract from autospectral plots; this can be attributed to various minor issues within the program due to singularities at the tips of the plate and at the center of each vortex. Additionally, the times at which specimens failed were recorded for each parametric study. It was expected that the failure time would increase for stiffer specimens with more damping; the results generated by the model supported this prediction by demonstrating positive correlations between these parameters and failure time yet no clear correlation between mass moment of inertia and failure time. The correlation between predictions and simulated results provides support that the program used is viable. With future adjustments and developments, the discrete vortex model has the potential to revolutionize the industry of flutter simulation, providing a faster and more efficient analysis technique that can be implemented early on in a structure's design process.

## ACKNOWLEDGMENTS

The author would like to thank Dr. William Culbreth for presenting a thesis on this topic and for his perpetual eagerness to teach, support, and guide the author through the research and development of this thesis work. The author would like to recognize the distinguished members of the graduate advisory committee, Dr. Woosoon Yim and Dr. Mohamed Trabia, and the graduate faculty representative, Dr. Evangelos Yfantis, for their interest and assistance with this project. Finally, the author would like to thank her friends and family for their continued support.

TABLE OF CONTENTS

ABSTRACT.....iii

ACKNOWLEDGMENTS ..... v

LIST OF TABLES .....viii

LIST OF FIGURES ..... ix

LIST OF SYMBOLS ..... xi

INTRODUCTION ..... 1

INVISCID FLOW THEORY..... 6

    2.1 Potential Flow ..... 6

    2.2 Complex Potential..... 8

    2.2 Conformal Mapping..... 10

DISCRETE VORTEX METHOD ..... 16

    3.1 Potential Flow Equation..... 16

    3.2 Introduction of Nascent Vortices ..... 18

    3.3 Transportation of Discrete Vortices..... 19

DYNAMICS OF TORSIONAL OSCILLATION ..... 23

    4.1 Pressure..... 23

    4.2 Force ..... 24

    4.3 Moment..... 25

    4.4 Dynamic Motion Equation..... 26

MODELING AND ANALYSIS OF FLUTTER ..... 28

    5.1 Calculations Performed..... 28

    5.2 Description of the Program..... 33

RESULTS ..... 38

    6.1 Inviscid Flow Results..... 38

    6.2 Discrete Vortex Results ..... 43

6.3 Flutter Results .....	57
CONCLUSION.....	72
FREEBASIC PROGRAM .....	76
A.1 Main Program .....	76
A.2 Complex Functions .....	104
SAMPLE INPUT FILE.....	112
SAMPLE OUTPUT FILE.....	114
SAMPLE BATCH FILE.....	121
FFT AUTOSPECTRA .....	122
REFERENCES .....	132
CURRICULUM VITAE.....	134



LIST OF TABLES

**Table 1 Elementary Aerodynamic Flows**..... 7

**Table 2 Conformal Mapping of the Flow Field**..... 13

**Table 3 Position and Velocity Transformation Equations** ..... 17

**Table 4 Program Interactive Key Functions** ..... 34

**Table 5 Inviscid Flow Pressure Distribution Plots**..... 41

**Table 6 Time Step Comparison** ..... 46

**Table 7 Nascent Vortex Offset Distance Comparison** ..... 52

**Table 8 Number of Time Steps to Trigger Asymmetry Comparison** ..... 53

**Table 9 Reynolds and Strouhal Number Comparison**..... 54

**Table 10 Summary of Results** ..... 62

**Table 11 Summary of Results (continued)**..... 63

## LIST OF FIGURES

<b>Figure 1</b> Flow over a cylinder .....	8
<b>Figure 2</b> Argand diagram of a complex number .....	9
<b>Figure 3</b> Transformation from a cylinder to a flat plate .....	12
<b>Figure 4</b> Model of flow past a cylinder using DVM .....	20
<b>Figure 5</b> Relationship between Strouhal number and Reynolds number .....	21
<b>Figure 6</b> X and Y components of force vector .....	24
<b>Figure 7</b> Flow chart of program .....	31
<b>Figure 8</b> Flow chart of program (continued) .....	32
<b>Figure 9</b> Physical flow field (main screen) .....	36
<b>Figure 10</b> Pressure distribution plot .....	37
<b>Figure 11</b> Force distribution plot .....	37
<b>Figure 12</b> Locations of key points in each z-plane .....	39
<b>Figure 13</b> Progression of the wake using DVM .....	44
<b>Figure 14</b> Time step Strouhal frequency error .....	47
<b>Figure 15</b> Force distribution plot with DVM .....	48
<b>Figure 16</b> Pressure probe measurement for stationary plate .....	49
<b>Figure 17</b> Autospectrum for pressure probe behind stationary plate .....	50
<b>Figure 18</b> Drag force on stationary plate .....	51
<b>Figure 19</b> Strouhal numbers compared with pseudo Reynolds number .....	56
<b>Figure 20</b> Flutter simulation .....	58
<b>Figure 21</b> Pressure comparison for varying torsional stiffness .....	59
<b>Figure 22</b> Deflection angle comparison for varying torsional stiffness .....	60
<b>Figure 23</b> Autospectra of deflection angle for varying torsional stiffness .....	61
<b>Figure 24</b> Failure time for varying torsional stiffness .....	64
<b>Figure 25</b> Failure times for varying damping coefficient .....	65

<b>Figure 26 Failure times for varying mass moment of inertia.....</b>	<b>66</b>
<b>Figure 27 Strouhal numbers for varying torsional stiffness .....</b>	<b>67</b>
<b>Figure 28 Strouhal numbers for varying damping coefficient.....</b>	<b>68</b>
<b>Figure 29 Strouhal numbers for varying mass moment of inertia .....</b>	<b>68</b>
<b>Figure 30 Damped frequency for varying torsional stiffness.....</b>	<b>69</b>
<b>Figure 31 Damped frequency for varying damping coefficient .....</b>	<b>70</b>
<b>Figure 32 Damped frequency for varying mass moment of inertia.....</b>	<b>70</b>
<b>Figure 33 FFT: Time step comparison.....</b>	<b>122</b>
<b>Figure 34 FFT: Offset distance of nascent vortices comparison.....</b>	<b>123</b>
<b>Figure 35 FFT: Number of time steps to trigger offset comparison.....</b>	<b>124</b>
<b>Figure 36 FFT: Uniform velocity comparison.....</b>	<b>125</b>
<b>Figure 37 FFT: Plate width comparison .....</b>	<b>125</b>
<b>Figure 38 FFT: Strouhal numbers for varying torsional stiffness .....</b>	<b>126</b>
<b>Figure 39 FFT: Strouhal numbers for varying damping coefficient.....</b>	<b>127</b>
<b>Figure 40 FFT: Strouhal numbers for varying mass moment of inertia .....</b>	<b>128</b>
<b>Figure 41 FFT: Damped frequency for varying torsional stiffness.....</b>	<b>129</b>
<b>Figure 42 FFT: Damped frequency for varying damping coefficient .....</b>	<b>130</b>
<b>Figure 43 FFT: Damped frequency for varying mass moment of inertia.....</b>	<b>131</b>

## LIST OF SYMBOLS

$a$	Radius of circle
$b$	Damping coefficient
$C_p$	Coefficient of pressure
$F$	Force
$F_x$	$X$ -component of force
$F_y$	$Y$ -component of force
$f$	Frequency in Hertz
$G$	Shear modulus
$I_{zz}$	Mass moment of inertia about the $z$ -axis
$j, k$	Indices
$J$	Polar moment of inertia
$K$	Doublet strength
$L$	Characteristic length
$M_z$	Moment about the $z$ -axis
$P$	Pressure
$r$	Radial distance from center of flat plate
$Re$	Reynolds number
$St$	Strouhal number
$s$	Position along profile
$t, dt, \Delta t$	Time, time step
$U$	Freestream velocity
$U_s$	Magnitude of velocity at separation point
$u$	Velocity in the $x$ -direction
$V$	Complex velocity
$v$	Velocity in the $y$ -direction
$w$	Velocity potential function, $w$ -plane
$z$	Complex number, $z$ -plane
$\alpha_i$	Initial angle of attack
$\Gamma$	Circulation (vortex) strength
$\theta$	Deflection angle
$\kappa$	Torsional spring constant
$\mu$	Dynamic viscosity
$\nu$	Kinematic viscosity
$\Lambda$	Source/sink strength
$\phi$	Velocity potential equation
$\psi$	Stream function
$\rho$	Density
$\omega$	Angular frequency in radians per second

## CHAPTER 1

### INTRODUCTION

Aeroelasticity is the interaction between aerodynamic forces and the elastic deformation and inertial forces of solid bodies. When disturbed by some aerodynamic force, an elastic body will deflect and return to its original shape. This movement may induce an oscillatory behavior as the object's stiffness and inertial forces alternate. A prominent and problematic case of aeroelasticity is the phenomenon called flutter, the unstable oscillation of a body such as an aircraft wing that can lead to sudden structural failure. There have been several cases of aircraft wings breaking off during flight due to flutter-induced motion. Flutter is not strictly limited to aircraft, but its prominence and potential consequences deem it a desirable topic of investigation for the aerodynamics community. This study seeks to develop a numerical simulation of aerodynamic flutter of a flat plate as a demonstration that discrete vortex method (DVM) can be utilized to generate results accurately while limiting the necessary computational power and time. Such a powerful tool can be used to quickly model test cases of flutter and provide useful data for aircraft design, investigations to better understand flutter, and other applications.

The phenomenon of flutter is present in many engineering disciplines including the design of aircraft wings and stabilizers, construction of suspension bridges or buildings, and commonplace objects such as powerlines and street signs. When fluid flows past a body and boundary layer separation occurs, vortices are shed behind the body, generating a region of high velocity and low pressure. The pressure distribution along the back surface of the structure induces aerodynamic forces on the body, altering its shape or position. The elastic forces in the body then restore it to its initial form. This pattern repeats itself resulting in oscillations of the position or shape of the body. When the oscillations approach the resonant frequency of the structure, they can become amplified and surpass the yield stress or fatigue limit of the body and result in failure.

Design elements have been developed to combat the effects of flutter. For example, fairings on aircraft wings and horizontal strakes on chimneys are designed to reduce the size of vortices shed and consequently reduce the aerodynamic forces on the body. While these methods are beneficial in remediating the effects of flutter, they do not entirely prevent the possibility of it occurring. Crucial to aircraft performance and safety, wings and stabilizers must be designed stiff enough to prevent flutter within the aircraft's operating envelope. Accounting for this phenomenon early in the design process mitigates the chance of flutter at its source. Unfortunately, this has not yet been considered practical due to a lack of understanding of the phenomenon and the time-consuming nature of computationally modeling flutter.

Since the 1930s, most flutter analysis relied on extensive calculations of a vehicle's mass and stiffness, so flutter checks were not performed until later in the design process. If the flutter check revealed any shortcomings of the structure, the aircraft was subject to an expensive overhaul of the design. Furthermore, subsequent changes resulted in heavier, less efficient aircraft. In the 1950s and 1960s, Finite Element Analysis (FEA) provided a new and improved way to gather flutter data. However, the expertise in this method was limited. The time to gather stiffness and mass properties and to mesh and model the aircraft resulted in FEA proving no more useful than traditional flutter analyses [1].

Today, aircraft flutter is a heavily researched topic. Flutter analyses include the attachment of flutter exciters on the wingtips, manipulation of extensive FEA matrices, and other expensive and time-consuming methods. Testing always occurs late in the design process and any findings tend to favor active control rather than redesign of the structure itself. Many studies and technologies have been developed to mature active flutter suppression and other corrective solutions. Damping out structural vibrations, maintaining low airspeeds to avoid critical flutter, and automatic control systems using the control surfaces to correct flutter are some of the existing solutions proposed to combat aeroelasticity. While these methods of limiting flutter work effectively, it would be more ideal to conduct flutter analyses in the early stages of design through numerical modelling in order to mitigate rather than remediate the issue.

Early analyses of flutter are beneficial because they call for simple and inexpensive changes to the aircraft design. Such an adjustment may be increasing the thickness and therefore stiffness of the structure

of the aircraft wing, for example, increasing its resistance to deflection caused by the airstream. If completed early on in the design process, flutter analyses can prevent flutter with the lowest economic impact on the design process. That is, however, dependent on the efficiency of the analysis. In 2001, Gonzales developed a Multidiscipline Design Optimization (MDO) to map together a structural Finite Element Model (FEM) and an aerodynamic Finite Volume Model [2]. By combining a matrix of aerodynamic coefficients and one of mass and stiffness properties, the system could be solved for the set of parameters that would induce flutter. This MDO capitalizes on recent technological advancements in computational power, rapid meshing and modeling techniques, numerical methods, and Computer Aided Design (CAD). This numerical method allows flutter analyses to be conducted earlier in the design cycle and at a reasonable pace.

While effective and continually advancing, the FEM method remains unnecessarily lengthy and costly. In order to design an aircraft structure with appropriate mass and stiffness properties, an approximation of the structure's aeroelasticity will suffice. In this study, a method much simpler than previous approaches to flutter analysis will be presented using a numerical method involving the classical aerodynamic equations for modeling vortices.

The discrete vortex method (DVM) provides insight into the effects of high Reynolds number flow on a rigid body through the positioning of vortices in a flow field to model a continuous vorticity starting at the separation point on a body. The technique was developed in the 1930s by Rosenhead [3] and used extensively in the 1970s as a computational approach to the modeling of wakes. Since then, DVM has not been used very often despite its efficiency and speed. With the rise of computational technology, DVM proved to be a very simple and efficient means of simulating a wake behind a rigid body [4]. This method was applied to a two-dimensional analysis of vortex shedding by Sarpkaya et al. off a flat plate in 1975 and off a circular cylinder in 1979 [5,6]. This research uses inviscid potential flow and boundary-layer interactions to project the propagation of shed vortices from a cylinder with time. A complex velocity-potential function is used to describe the flow field across a cylinder and Pohlhausen's steady-state approximation is used to locate the separation points of the vortices which constantly oscillate around the circumference of the profile with every time step. Finally, the vortices are discretized as their positions

advance with time. The numerical results of Sarpkaya's study resulted in reasonable values and visuals to explain vortex formulation and shedding from a two-dimensional circle. The graphical results obtained by plotting the locations of all the discrete vortices provide excellent visual aids of a computed wake behind the body. Sarpkaya's work will be closely followed in this study; the cylinder will serve as a starting point and subsequently be transformed across several planes using conformal mapping to mimic a flat plate at various angles in inviscid flow.

In 1997, Walther et al. applied two-dimensional DVM to bluff bodies including a cylinder and a flat plate [7]. Walther's work included an analysis of a flat plate subject to a harmonic pitching motion. The following is a survey of the work produced in the effort to study the airflow past a stop sign undergoing torsional oscillations due to aeroelastic deformation, employing a feedback loop rather than simply a harmonic input. These results can be compared with the numerical study of flow past an oscillating flat plate studied by Walther and the vortices shed off the cylinder studied by Sarpkaya [6]. Assuming inviscid flow and treating a flat plate as a bluff body, the results should be similar.

The objective of this study is to develop a simpler, faster numerical method to model and understand the consequences of flutter on an airstream. A simple case of flutter is the torsional oscillation of a traditional stop sign subject to wind. This is a simple geometry which can be modeled as a flat plate that is free to rotate about its center. Boundary layer separation would occur at the tips of the flat plate. The stop sign post has a designated torsional stiffness, damping, mass moment of inertia, and torsional stress limit. This investigation is a precursor to analysis of a Joukowski airfoil of which the locations of vortex shedding are less obvious than those on a stop sign or flat plate. A Joukowski airfoil can be obtained through a series of conformal mappings slightly modified from those used to obtain a flat plate; this profile has been modeled in DVM before but never as an oscillating body [8]. With Inviscid Flow Theory and DVM as a basis for mathematical formulation and the use of compiled FreeBASIC software to generate a computational algorithm, a two-dimensional flat plate is transformed through various complex planes demonstrating changing angles of attack determined by the torsional damping and inertia of the plate. The benefit of using FreeBASIC is the availability of an existing graphics feature and a compiler that creates an



application just as fast as one built in C programming language. The discrete vortices are shed from the edges of the plate and their distance from the edge determined by DVM calculations; the vortices can subsequently be observed and the resulting pressure distribution, moment, and torsional resistance on the stop sign can be calculated to model the sign's movement in the wind. When this algorithm is eventually translated to analyze a Joukowski airfoil, this quick approximation of the airfoil's aeroelasticity may provide a better understanding of a very pertinent flutter for the aerodynamics community.

## CHAPTER 2

### INVISCID FLOW THEORY

The flow model in this thesis will be approximated using inviscid flow theory. Neglecting friction, thermal conduction, and diffusion of the fluid allows for a very simple and highly accurate model of flow at high Reynolds numbers. While zero viscosity flow does not exist in nature, many aerodynamic flows can be modeled without the influence of transport phenomena. Equations used in this section are the culmination of past work on the topics of inviscid flow and conformal mapping.

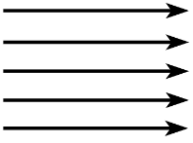
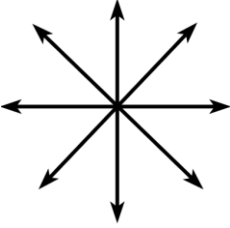
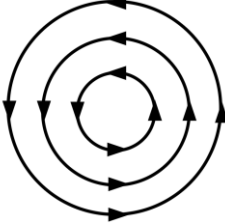
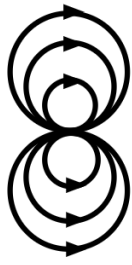
#### 2.1 Potential Flow

Inviscid flow theory is a branch of fluid dynamics that models the behavior of fluid while neglecting its viscosity. This assumption corresponds to highly turbulent flow. Note that flows with significant vorticity cannot be effectively predicted by this model. For inviscid flows, the no-slip condition does not apply. In other words, the transverse velocity gradient found in the boundary layer on a surface can be ignored. The flow beyond the boundary layer is considered *potential flow* because it obeys Laplace's equations and the same laws as electromagnetic fields. Without a boundary layer, the entire flow field can be modeled through the use of potential flow equations. Potential flow is defined as a vector field equal to the gradient of the velocity potential,  $\phi$ , a function of both space and time (Equation 1). The curl of the velocity field is always equal to zero indicating irrotationality except at singularity points and, in the case of incompressible flow, the velocity potential also satisfies Laplace's equation.

$$\vec{V} = \vec{\nabla}\phi, \quad \nabla^2\phi = 0 \quad (1)$$

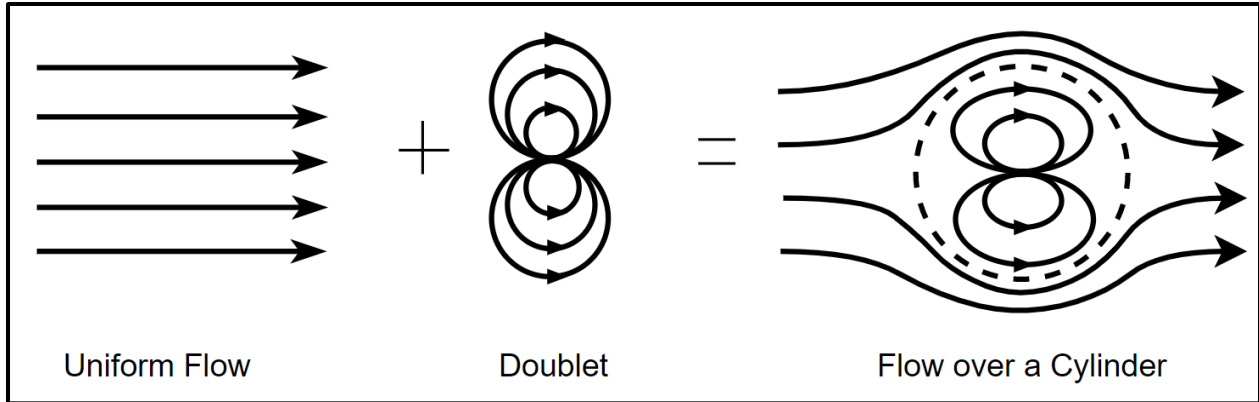
The advantage of using inviscid flow is that a potential flow field can be constructed simply by superimposing simple flow patterns in the same flow field. For this method, several elementary aerodynamic flows can be represented by their velocity potentials or stream functions in the following table [9]. These simple flow patterns create much more complex flow patterns via superposition.

**Table 1 Elementary Aerodynamic Flows**

Type of Flow	Velocity Potential	Stream Function
Uniform 	$\phi = Ur \cos \theta$	$\psi = Ur \sin \theta$
Source / Sink 	$\phi = \pm \frac{\Lambda}{2\pi} \ln r$	$\psi = \pm \frac{\Lambda}{2\pi} \theta$
Vortex 	$\phi = \frac{\Gamma}{2\pi} \theta$	$\psi = -\frac{\Gamma}{2\pi} \ln r$
Doublet 	$\phi = \frac{K \cos \theta}{2\pi r}$	$\psi = -\frac{K \sin \theta}{2\pi r}$

For more complex and practical flow patterns, these simple patterns can be summed to yield the desired result. A very common pattern is flow past a circular cylinder, demonstrated in Figure 1. This pattern is achieved through the addition of uniform flow and a doublet. The resulting equations for velocity potential and streamlines are

$$\phi = U \left( r + \frac{K}{2\pi U r} \right) \cos \theta, \quad \psi = U \left( r - \frac{K}{2\pi U r} \right) \sin \theta \quad (2)$$



**Figure 1 Flow over a cylinder**

Flow past a cylinder is symmetrical in inviscid flow but realistically would include a wake behind the cylinder inducing drag on it. However, in inviscid flow, there is no separation of the boundary layer. As a result, it is expected that the sum of the aerodynamic forces acting on a body in inviscid flow is always equal to zero and flutter cannot be induced.

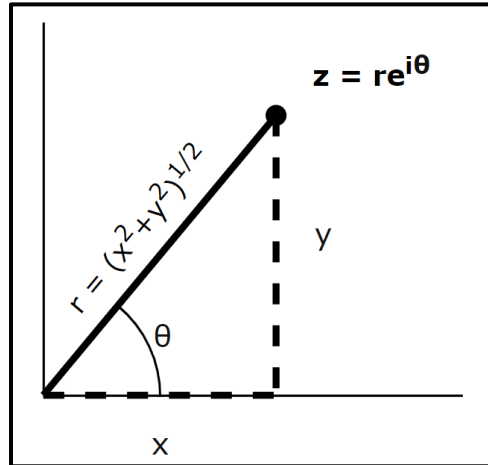
## 2.2 Complex Potential

At each node of the velocity field, the flow can be represented by a vector in the direction of the fluid motion at that position. Representation of a two-dimensional vector can be accomplished with a single complex number,  $z$ . For the purposes of this study, it is necessary to have a good understanding of complex numbers. Defined in the Argand-Gauss plane, or complex plane, a complex number,  $z$ , consists of a real part or abscissa ( $x$ ) and an imaginary part or ordinate ( $y$ ) and is plotted analogously to a coordinate on a Cartesian system. A complex number is mathematically represented by

$$z = x + iy = re^{i\theta} \quad (3)$$

where  $r$  represents the modulus of the complex number and can be calculated as the magnitude or absolute value of  $z$  while  $\theta$  is the argument of the number and is equal to the inverse tangent of the imaginary term divided by the real term. The Euler (polar) form of the complex number assigns both a magnitude and

direction to the specified point. All complex numbers abide by operations under the complex variable theory, the details of which will not be elaborated in this thesis.



**Figure 2 Argand diagram of a complex number**

With this formulation, a flow field may be described by the *complex potential* equation,  $w$ . At each point in the field, a discrete volume of flow has a position defined by a complex number,  $z$ , providing its  $x$  and  $y$  locations in a complex, two-dimensional space, thereby making  $w$  a function of  $z$ , the complex variable. This relationship allows for the mapping from the  $w$ -plane (uniform flow) to the  $z$ -plane (uniform flow around a circle, for example).

$$w = f(z) = \phi(x, y) + i \psi(x, y) \quad (4)$$

For every point  $(x, y)$  in the  $z$ -plane, there exists a corresponding point  $(\phi, \psi)$  in the  $w$  plane. In the above equation, the real components are combined into the term,  $\phi$  (phi). The velocity potential is given by the real part of the function  $w$  and, when set equal to a constant, yields the equipotential lines of the flow field. The imaginary components are combined into the term,  $\psi$  (psi). The stream function represents the imaginary part of  $w$  and the streamlines of the flow field. The streamlines of a flow field are an excellent tool for visualizing flow and will be utilized for this purpose. It can be demonstrated that both equations,  $\phi$  and  $\psi$ , satisfy the Laplace equation because they meet the conditions of continuity and irrotational flow,

thus restricting the function  $w$  to be considered analytic at all point except at singularities [10]. Singularities occur where the velocity of the flow is either zero or infinite. At these points, the conformal mapping process does not apply. For the purposes of a computational model, it is acceptable to neglect stagnation points, sharp corners, and the centers of vortices, doublets, etc.

Likewise, the velocity potential can be described by a function of complex numbers in a flow field. The *complex velocity*, of complex operator, is the complex conjugate of the derivative of the function  $w$  with respect to  $z$ . Each point in the complex plane can also be assigned a complex velocity which is the time derivative of  $z$ . Complex velocity also has both a real and imaginary component as it is represented by a vector at each point in the flow field with both a magnitude (speed) and direction. In the complex plane, the real part of complex velocity is denoted by  $u$  (velocity in the  $x$ -direction) and the imaginary part by  $v$  (velocity in the  $y$ -direction). It is equal to the complex conjugate of the derivative of the  $w$  function.

$$\frac{dw}{dz} = u - iv = \bar{V} \quad (5)$$

## 2.2 Conformal Mapping

The superposition of potential flow patterns can be executed mathematically with a conformal mapping technique. Conformal mapping is the process of using analytic functions to transform points from one complex plane to another through a consistent relation. Physically, the  $w$ -plane always displays uniform, horizontal flow and is transformed into the flow of interest in the  $z$ -plane. To obtain different physical flows, multiple relations such as  $z_2 = f(z_1)$  may be used to map the fluid flow across different planes to achieve the desired flow pattern. Common flow patterns such as a source, vortex, and doublet can be mapped from a  $w$ -plane of parallel flow using a simple transformation. For example, the transformation from a uniform flow pattern in the  $w$ -plane to a vortex of circulation strength  $\Gamma$  centered at  $z = a$  in the  $x$ -plane is given by

$$w(z) = -\frac{i\Gamma}{2\pi} \ln(z - a) \quad (6)$$

For every position in the  $w$ -plane, there is a corresponding position in the  $z$ -plane which represents the physical flow pattern of a vortex. These transformation equations resemble the velocity potential and streamline equations given in Table 1 due to the relationship between  $w$ ,  $\phi$ , and  $\psi$ . Another equation of interest is the transformation from uniform flow to flow past a cylinder given by the equation

$$w(z) = U \left( z + \frac{a^2}{z} \right) \quad (7)$$

where  $a$  is the radius of the cylinder and is also equivalent to  $\sqrt{\frac{K}{2\pi U}}$  where  $K$  is the strength of the doublet.

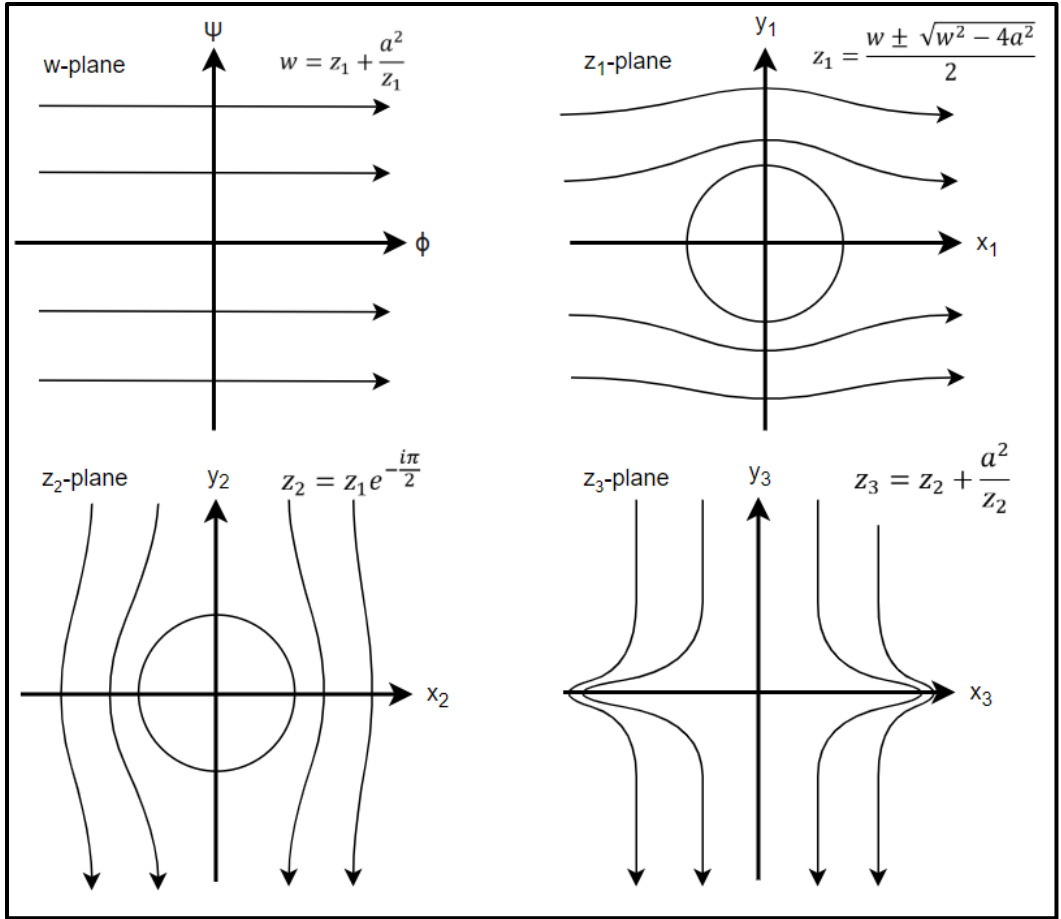
Again, this transformation equation can be derived from the summation of the equations for uniform flow and doublet flow given in Table 1. From this transformation, the complex velocity can also be derived as the derivative of the function.

$$\bar{v} = \frac{dw}{dz} = U \left( 1 - \frac{a^2}{z^2} \right) \quad (8)$$

Through a series of transformations, the circular streamline can be manipulated through several different  $z$ -planes to obtain a different profile. For example, a circular profile can be flattened to model flow past a flat plate. The flow field must first be rotated so that the free stream velocity travels from the top of the plane to the bottom. Rotation of the flow field through an angle  $\alpha$  can be achieved by multiplying the current  $z$  function by  $e^{i\alpha}$ . The circle with radius,  $a$ , can be flattened into a flat plate of width  $4a$  by adding the current  $z$  function and  $\frac{a^2}{z}$ . This series of transformations is illustrated in Figure 3.

The velocity at any point in the  $z_n$ -plane can be found by multiplying several derivatives. These derivatives are obtained by differentiating the equations shown in Figure 3 using the Chain Rule. When the derivative of the  $w$ -plane with respect to the  $z$ -plane of interest is found, the complex conjugate of this value gives the velocity vector at any position.

$$\frac{dw}{dz_n} = \frac{dw}{dz_1} * \frac{dz_1}{dz_2} * \dots * \frac{dz_{n-1}}{dz_n}, \quad V_n = \overline{\frac{dw}{dz_n}} \quad (9)$$


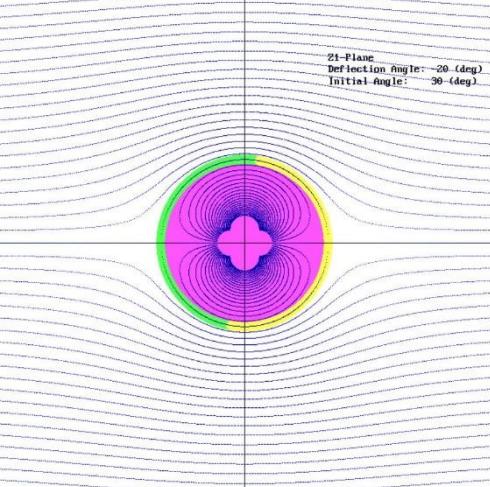
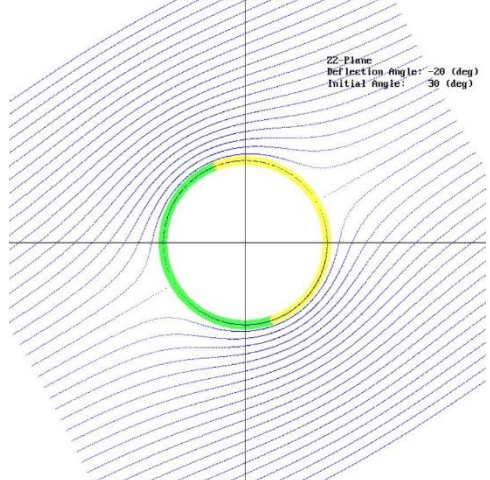


**Figure 3 Transformation from a cylinder to a flat plate**

To account for angle of attack and deflection angle of the plate, three more conformal mappings are added the series. The conformal mappings used in this study are provided in Table 2 with a description and corresponding equation for each plane. The  $z_6$ -plane is the physical plane and the plane of interest and will be observed for results in this study. In the study, the direction of uniform flow can be rotated by an angle of attack,  $\alpha$ , and the flat plate can elastically deflect through a deflection angle of  $\theta$  due to interaction with the flow. A conformal mapping of a flat plate at a deflection angle of -20 degrees and an angle of attack of 30 degrees is shown below. The bright green line represents the front surface of the flat plate and the yellow line the back; the junctions of the green and yellow lines are the location of the tips of the plate.



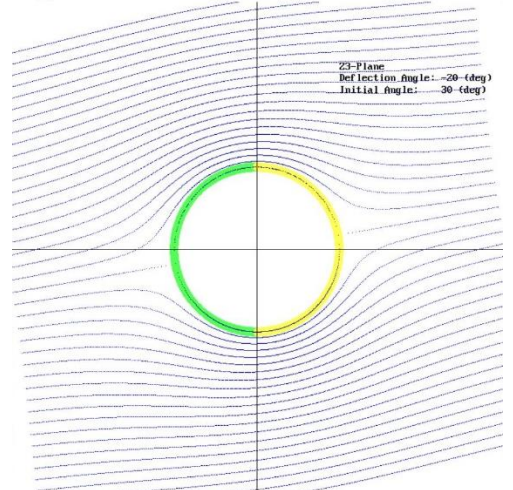
**Table 2 Conformal Mapping of the Flow Field**

Plane	Equation	Description	Example
W	$w = Uy$	Uniform flow	 <p>W-Plane Deflection Angle: -20 (deg) Initial Angle: 30 (deg)</p>
Z <sub>1</sub>	$z_1 = \frac{w \pm \sqrt{w^2 - 4a^2}}{2}$	Uniform flow around a circle	 <p>Z1-Plane Deflection Angle: -20 (deg) Initial Angle: 30 (deg)</p>
Z <sub>2</sub>	$z_2 = z_1 e^{i\alpha_i}$	Rotates the flow through initial angle of attach ( $\alpha_i$ )	 <p>Z2-Plane Deflection Angle: -20 (deg) Initial Angle: 30 (deg)</p>

$Z_3$

$$z_3 = z_2 e^{i\theta}$$

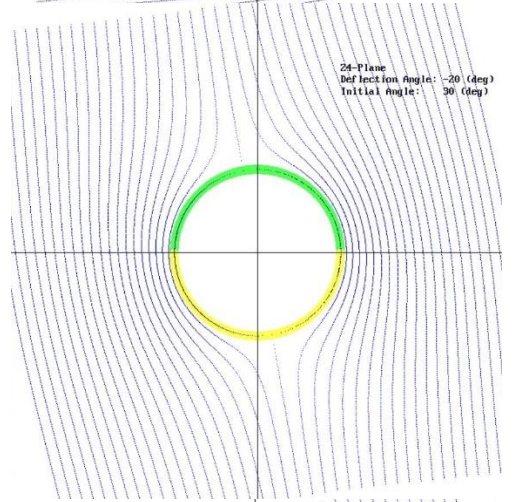
Rotates the flow through deflection angle ( $\theta$ )



$Z_4$

$$z_4 = -iz_3$$

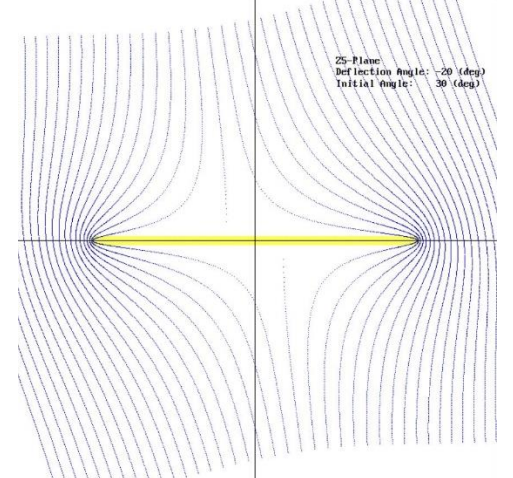
Rotates the circle 90 degrees; tips of flat plate fall on  $x$ -axis



$Z_5$

$$z_5 = z_4 + \frac{a^2}{z_4}$$

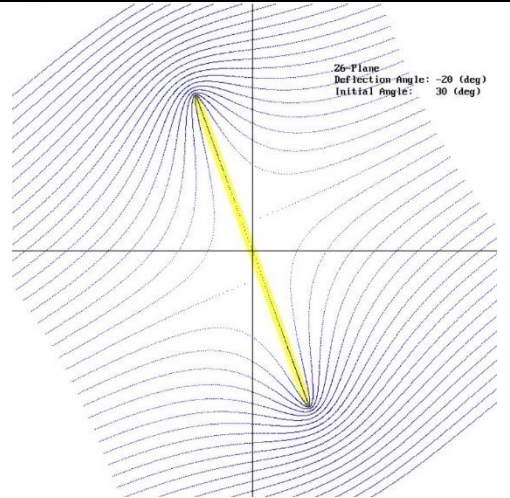
Flattens circular profile of radius  $a$  into flat plate of length  $4a$



$Z_6$

$$z_6 = iz_5 e^{-i\theta}$$

Rotates flat plate  
back to original angle  
of attack



## CHAPTER 3

### DISCRETE VORTEX METHOD

Flutter requires nonzero aerodynamic forces acting on a body to induce motion. These forces are a result of vortex shedding, a characteristic of viscous flow in which the no-slip condition applies and results in boundary layer separation. To incorporate rotational flow effects into an inviscid model, elementary vortex flow given by Equation 6 is introduced in the flow field which was previously an example of pure inviscid flow. The discrete vortex method (DVM) is a means of modeling a wake that is realistic of a viscous flow while still maintaining the simplistic calculations from inviscid flow theory.

#### 3.1 Potential Flow Equation

A discrete vortex model is used to model a continuous vorticity in a flow field as a series of line vortices. The flow field begins as an inviscid flow as demonstrated in the previous chapter and a number of individual vortex flows are superimposed onto the flow field. The vortices are introduced at the tips of the flat plate to account for the boundary layer separation that occurs and to meet the Kutta condition. The circulation of each discrete vortex introduced into the flow is computed to ensure that the Kutta condition is met at the tips of the plate which are separation points. The number of vortices included in the flow increases with time; with every time step, the complex potential function becomes more and more complicated. The complex potential equations of the inviscid flow field and each additional vortex and its image can all be superimposed since both equipotential and stream functions obey Laplace's equation. Sarpkaya used the following equation to model the wake behind a cylinder [6].

$$w(z_1) = U \left( z_1 + \frac{a^2}{z_1} \right) - \frac{i}{2\pi} \sum_{j=1}^m \Gamma_j \left[ \ln(z_1 - z_{1,j}) - \ln \left( z_1 - \frac{a^2}{\bar{z}_{1,j}} \right) \right] \quad (10)$$

The first term of the complex potential equation is the equation for inviscid flow past a cylinder, Equation 8. The second term is the addition of  $m$  vortices into the flow field. Each vortex has a strength  $\Gamma$

and position,  $z_1$ , in the  $z_1$ -plane. The first term in the summation comes from Equation 6 for vortex flow. The second term in the summation creates an image of the vortex inside the cylinder to maintain the shape of the profile streamline. The positions of the vortices, as well as any other points in the flow field, can be mapped to the physical plane by the equations summarized in Table 3.

To find the velocity of any point in the physical plane, the  $z_6$ -plane, complex potential function is differentiated and multiplied with the derivatives of the conformal mapping equations, as described in Equation 9. The derivatives used in this model are summarized in Table 3.

**Table 3 Position and Velocity Transformation Equations**

Planes	Position Transformation	Velocity Transformation
$w \rightarrow z_1$	$z_1 = \frac{w \pm \sqrt{w^2 - 4a^2}}{2}$	$\frac{dw}{dz_1} = 1 - \frac{a^2}{z_1^2}$
$z_1 \rightarrow z_2$	$z_2 = z_1 e^{i\alpha}$	$\frac{dz_1}{dz_2} = e^{-\alpha i}$
$z_2 \rightarrow z_3$	$z_3 = z_2 e^{i\theta}$	$\frac{dz_2}{dz_3} = e^{-\theta i}$
$z_3 \rightarrow z_4$	$z_4 = z_3 e^{-i\frac{\pi}{2}} = -iz_3$	$\frac{dz_3}{dz_4} = i$
$z_4 \rightarrow z_5$	$z_5 = z_4 + \frac{a^2}{z_4}$	$\frac{dz_4}{dz_5} = \frac{1}{1 - a^2/z_4^2}$
$z_5 \rightarrow z_6$	$z_6 = z_5 e^{i(\frac{\pi}{2} - \theta)}$	$\frac{dz_5}{dz_6} = -ie^{i\theta}$

The final function for velocity is given by the following equation. This allows the complex velocity to be found at any position in the  $z_6$ -plane.

$$\bar{V} = u - iv = \frac{dw}{dz_6} = \frac{e^{i(\alpha+2\theta)}}{e^{2i(\alpha+\theta)} + \frac{a^2}{z_1^2}} \left\{ U \left( 1 - \frac{a^2}{z_1^2} \right) - \frac{i}{2\pi} \sum_{j=1}^m \Gamma_j \left[ \frac{1}{z_1 - z_{1,j}} - \frac{1}{z_1 - \frac{a^2}{z_{1,j}}} \right] \right\} \quad (11)$$

### 3.2 Introduction of Nascent Vortices

For an impulsively started flow, pure inviscid flow past a body is used to begin the simulation. With every time step, a discrete vortex is introduced into the flow field at each of the two tips of the plate. The size of the time step is selected as 0.125 seconds in this study based on previous findings that this time step was adequate for a realistic model [6]. However, this time step can be changed. These vortices are called nascent vortices and two new nascent vortices are introduced in each time step. The location of the separation points can be found using Pohlhausen's approximation or by the assumption that separation occurs at the tips of a flat plate. The vortices are introduced at a certain distance away from the surface of the body and with a specific strength such that the velocity at the surface of the body is zero in order to maintain the no-slip condition and satisfy the Kutta condition.

The circulation strengths of these nascent vortices are given by Equation 12 and rely on a calculation of  $U_s$ , the velocity on the surface at the separation point. Velocity is calculated at the separation point using Equation 11. For inviscid flow past a flat plate, there is a singularity at the tips of the plate where the flow separates resulting in infinitely high velocities. As a result, the position used to solve Equation 11 is offset from the tip of the plate by a distance of  $0.25*a$ . Testing the program with varying offset distances reveals that this parameter has an insignificant effect on the wake behind the plate. Other studies have resolved this issue by a variety of means equally arbitrary. Free surface theory and an averaging technique [5] were considered for this study but were deemed significantly more laborious, yet no more defensible, than the offset of nascent vortices from the tips of the plate.

$$\Gamma_n = \frac{1}{2} U_s^2 * dt \quad (12)$$

The circulation strength of the vortex is largely constant and only varies with time due to the steady dissipation of all vortices or the amalgamation and elimination of individual vortices. The sign of the strength is found by taking the cross product of the complex position of the tip of the plate and the complex velocity at that point. A positive vortex strength indicates that it is spinning counterclockwise while a negative vortex strength indicates that the vortex is rotating clockwise. As vortices are continuously added

into the flow field, the total circulation in the field should remain zero due to the Kelvin Circulation Theorem which states that the total circulation in a system is conserved. While the circulation strengths of the nascent vortices formed at the tips of the plate are not expected to be exactly equal, the total circulation in the flow field should remain relatively close to zero in the implementation of DVM.

Once the circulation of the nascent vortex is found, the release position of the vortex is calculated using Equation 13. The angle of the release point aligns with the angle of the tips of the plate. The position is related to the strength of the vortex.

$$z_n = \left( \frac{1 + \frac{|\Gamma_n|}{2\pi U_s}}{1 - \frac{|\Gamma_n|}{2\pi U_s}} \right) e^{i(\pi-\theta)} \quad (13)$$

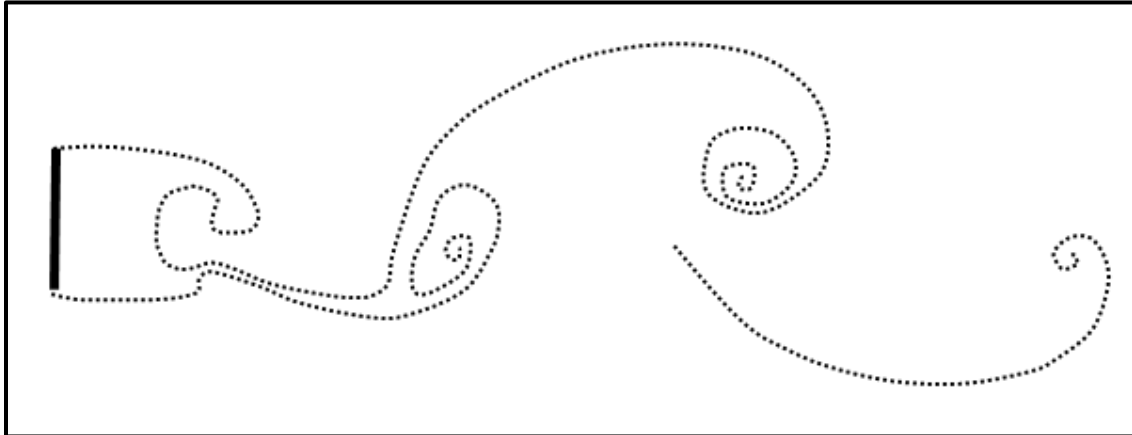
### 3.3 Transportation of Discrete Vortices

In each time step, all existing vortices in the flow are transported by calculating the velocity at their present locations through Equation 11 and moving them in the corresponding direction in the flow field using a first order Euler method equation, given in Equation 14. As a result, the complex potential function, Equation 10, becomes more complicated with time as the number of terms in the summation increases with each time step.

$$z_{i+1} = z_i + V * dt \quad (14)$$

where both  $z$  and  $V$  are two-dimensional vectors expressed as complex numbers. Each vortex's position,  $z$ , has an  $x$  and  $y$  component while its velocity,  $V$ , has a  $u$  (velocity in the  $x$ -direction) and  $v$  (velocity in the  $y$ -direction) component.

As the vortices move downstream behind the body, they interact and create a realistic wake in which groups of vortices move together in rotating pockets. As one pocket forms, it creates a low-pressure zone and pulls fluid from the other side of the body. As a result, these pockets of rotating vortices alternate and create a von Kármán vortex street [11]. An image of predicted results based on past work is shown below with the general trend of the vortices outlined to simplify the flow visualization.



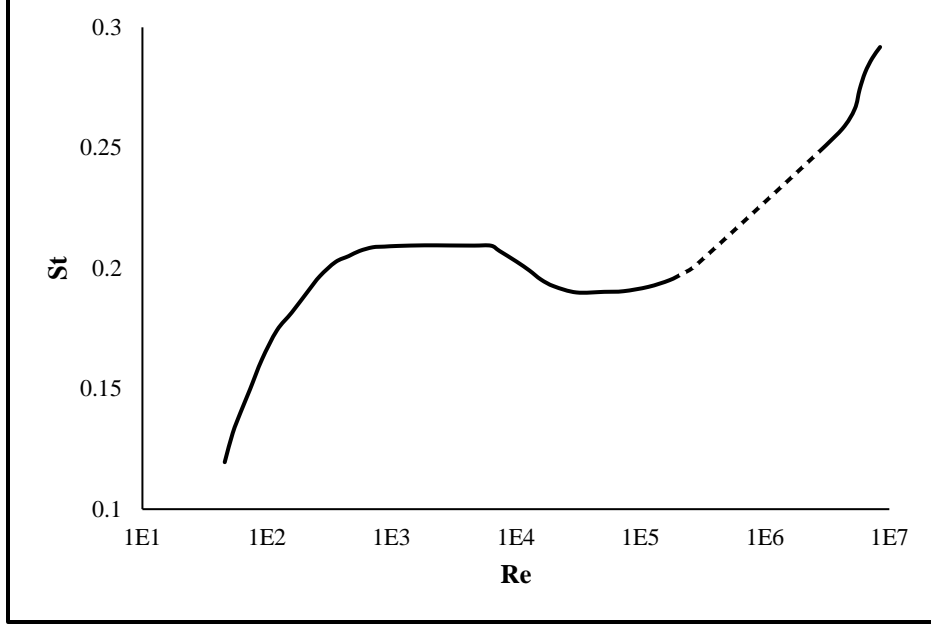
**Figure 4 Model of flow past a cylinder using DVM**

The alternating pattern of the wake can be described by Strouhal number, a dimensionless property given by Equation 15. This value is related to the frequency at which the rotating pockets of move past a given point and can be found by placing a pressure probe in the middle of the wake and measuring the velocity at that point in every time step. Strouhal number is related to Reynolds number by the following graph obtained from past work and many experimental tests [12]. One portion of the graph is dashed because there is insufficient data to characterize it. While some models interpolate the curve in this region, others hypothesize different types of behaviors. Reynolds number is calculated by Equation 16 and is also dimensionless. In both equations,  $L$  is the characteristic length of the body. For flow past a cylinder,  $L$  would be set equal to the diameter of the circle. For the case of a flat plate in this study,  $L$  is equivalent to  $4a$ . This correlation between  $St$  and  $Re$  can be used to validate results of the wake generated using DVM.

$$St = \frac{fL}{U} \quad (15)$$

$$Re = \frac{UL}{\nu} \quad (16)$$





**Figure 5 Relationship between Strouhal number and Reynolds number**

The frequency of the wake can be found by placing a probe in the wake to measure the pressure at each time step. A plot of the pressure at this location over time can be investigated with a Fast Fourier Transform (FFT). FFT is an extremely efficient algorithm designed to execute the transformation of a data set of  $N$  discrete values using a Discrete Fourier Transform (DFT). The Fourier transform of the data set is given by Equation 17 where  $f_n$  represents a discrete value of the input which is a function of time and  $F_k$  represents a discrete value of the Fourier Transform output,  $F(\omega)$ , which is a function of the angular frequency [13]. There exist several software packages that can compute the FFT of a data set which will be used to identify the Strouhal frequency of the wake.

$$F_k = \frac{1}{N} \sum_{n=0}^{N-1} f_n e^{-\frac{2\pi i k n}{N}} \quad (17)$$

Here,  $N$  represents the total number of values in the input data set,  $f(t)$ , which is measured at discrete time intervals,  $\Delta t$ . The complex Fourier function,  $F_k$ , is the discrete Fourier transform of  $f(t)$  at the frequency,  $\omega = 2\pi k/N$ . The input signal at time index,  $n$ , is  $f_n = f(n*\Delta t)$ .

The frequency interval of the autospectrum is given by the reciprocal of the total time elapsed during data collection. Only the first half of the autospectrum is plotted on a logarithmic scale and analyzed for peaks indicating dominating frequencies in the data. The autospectrum shows the correlation of the input signal with itself as a function of frequency. The autospectral or power spectral density,  $PSD(\omega)$ , can be found from the Fourier transform of the input signal,  $F(\omega)$ , as well as the phase log,  $\Phi(\omega)$ :

$$PSD(\omega) = F(\omega) * F(\omega) \quad (18)$$

$$\Phi(\omega) = \tan^{-1} \left( \frac{Im(F(\omega))}{Re(F(\omega))} \right) \quad (19)$$

Peaks in the power spectral density correspond to frequencies where the “energy” in the signal is concentrated and can indicate Strouhal frequencies or the natural resonant frequency of the flat plate.

## CHAPTER 4

### DYNAMICS OF TORSIONAL OSCILLATION

To model flutter of a body, the aerodynamic forces due to the wake behind the body must first be modeled with DVM. The forces can then be converted into moments which have a fundamental role in a feedback loop that drives the motion of the body. These calculations account for the aerodynamic, elastic, and inertial forces, providing a comprehensive model of flutter.

#### 4.1 Pressure

The pressures along the front and back surfaces of a body are not constant. When DVM is applied to the flow field, the distribution is not even symmetrical. As a result, it is necessary to divide the profile of the body up into several segments and find the point along each of them. The circular profile in the  $z_1$ -plane is divided into line segments. At the center of each line segment, the pressure of the fluid acting at the center of that line is found through Bernoulli's equation. Normalizing the equation allows the coefficient of pressure can be found easily.

$$C_p(r) = \frac{P - P_\infty}{\frac{1}{2}\rho U^2} = 1 - \frac{|V(r)|^2}{U^2} \quad (20)$$

where  $U$  is the streamline velocity,  $V$  is the velocity on the surface at a distance  $r$  from the center of the plate. The velocity is found from Equation 11. The coefficient of pressure can reach a maximum of one when the velocity is zero (stagnation conditions) and the static fluid is pushing on the plate. This means the maximum positive force exerted normal to the body is at the stagnation point. However, the coefficient of pressure can reach extremely low negative values when the velocity is high and the fluid is pulling on the plate. For this study, this is likely to occur on the back surface of the plate where high velocity activity occurs and particularly at locations close to the center of an individual vortex. Each discrete vortex has a

singularity of infinite velocity at its center, a deviation from realistic vortices which have viscous flow near their centers.

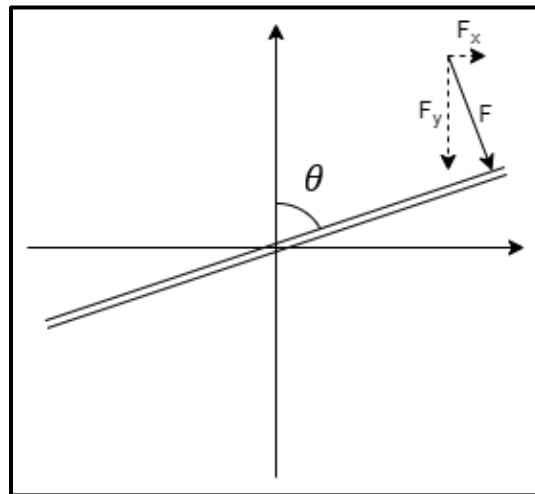
#### 4.2 Force

At the center of each two-dimensional line segment, the pressure can be equated to an equivalent force acting on the surface. Since pressure is the distribution of force over a certain area, the normalized force at a distance  $r$  from the center of the flat plate can be calculated by multiplying the pressure coefficient,  $C_p$ , by the length of the line segment,  $\Delta s$ .

$$F = P * A, \quad F(r) = C_p \Delta s \quad (21)$$

This force can be resolved into  $x$  and  $y$ -components using Equation 22. The purpose of finding the force in both dimensions is to simplify the calculations of the moments acting on the plate and also to be able to calculate the total drag force acting on the plate given by the sum of the  $x$ -components of force on each of the line segments. Figure 6 demonstrates the flat plate deflected at a positive angle  $\theta$ . The force acting on the front surface of the plate is perpendicular to it. Since the  $y$ -component is negative despite the positive angle, the equation for  $F_y$  must include a negative sign.

$$F_x(r) = F(r) \cos \theta, \quad F_y(r) = -F(r) \sin \theta, \quad (22)$$



**Figure 6 X and Y components of force vector**

The total drag force acting on the plate can be found by summing the  $x$ -components of force found on each line segment on each surface of the plate using Equation 23. By subtracting the total force on the back surface of the plate from the total force on the front surface of the plate, the net force at each point can be calculated. In inviscid flow when separation does not occur, the total force is expected to be zero. However, when DVM is applied to the flow field, most of the drag force will come from the pressure distribution on the back surface of the plate. Due to the rotating fluid behind the plate, there will be a region of high velocity and low pressure pulling the plate backward.

$$F_x = \int_{-2a}^{2a} F_x(r) dr, \quad F_y = \int_{-2a}^{2a} F_y(r) dr \quad (23)$$

Since the length of each line segment,  $\Delta s$ , was chosen based on an even distribution of line segments along the circumference of the cylinder in the  $z_1$ -plane, the length of these segments in the  $z_6$ -plane varies. The segments are longer near the center of the plate and shorter near the tips. This variation in the step size of  $r$  affects the force calculations and results in a slight difference between the shapes of the force and pressure distribution plots.

#### 4.3 Moment

The velocity, coefficient of pressure, and force are given as functions of the radius from the center of the plate. The reason for this notation is because the plate is assumed free to rotate about its center. The deflection angle of that rotation is dictated by the total moment about the plate's center. That moment can be calculated from the force moments previously obtained. Each  $y$ -component of force is multiplied with the  $x$ -component of the moment arm from the center of the plate which is simply the  $x$ -coordinate of the center point of the line segment. Additionally, each  $x$ -component of force is multiplied by the  $y$ -component of the moment arm from the center of the plate given by the  $y$ -coordinate of the point. These moments are summed up across the front and back surfaces of the plate to find the net moment acting on the plate about its center. Like with the forces, the net moment acting on the back of the plate is subtracted from the net

moment acting on the front of the plate. A positive moment is considered counterclockwise and a negative moment is considered clockwise.

$$M_z = (\vec{r} \times \vec{F}) \hat{k} = \int_{-2a}^{2a} (xF_y - yF_x) dr \hat{k} \quad (24)$$

#### 4.4 Dynamic Motion Equation

The net moment exerted by the aerodynamic forces on the plate cause it to rotate about its center. The angle by which it rotates is determined by a second order differential equation, Equation 25. The motion of the plate depends on the moment induced by the fluid and also by the plate's structural properties: mass moment of inertia,  $I_{zz}$ , damping coefficient,  $b$ , and torsional spring constant,  $\kappa$  [14]. All these values affect the rotation of the flat plate about the  $z$ -axis which is located at its center of the plate. A higher torsional stiffness would cause the plate to deflect less under the same aerodynamic loading and return to its original position faster. A higher mass moment of inertia would result in initial resistance to deflection from the starting position but also result in continued motion and failure to smoothly return to its initial position. A higher damping coefficient would cause the plate to oscillate back and forth less when restoring itself to its initial position.

$$I_{zz}\ddot{\theta} + b\dot{\theta} + \kappa\theta = M_z(t) \quad (25)$$

The first and second time derivatives can be approximated using finite differencing. For the second time derivative of the deflection angle, a second-order accurate central difference is utilized. For the first time derivative, a first-order accurate forward difference is used. A forward difference is chosen because it uses the most recently recorded values of theta.

$$\ddot{\theta} \cong \frac{\theta_{i+1} + \theta_{i-1} - 2\theta_i}{(\Delta t)^2}, \quad \dot{\theta} \cong \frac{\theta_{i+1} - \theta_i}{\Delta t} \quad (26)$$

By substituting these approximations into Equation 25 and rearranging the terms, the deflection angle in the current time step can be solved as a function of the net moment found in Equation 24, the plate properties assigned by the user, and the deflection angle recorded in the two previous time steps.

$$\theta_i = \frac{\theta_{i-1}(2I_{zz} - \kappa(\Delta t)^2 + b\Delta t) - I_{zz}\theta_{i-2} + M_z(\Delta t)^2}{I_{zz} + b\Delta t} \quad (27)$$

Because the motion of the plate can be described by a second order differential equation, it has both an undamped (natural) and damped resonant frequency given by the following equations. These frequencies are functions of the plate properties. The plate can be expected to fail when it begins to oscillate at its damped resonant frequency due to the aerodynamic flow. A stiffer plate with a higher rigidity,  $\kappa$ , would be expected to demonstrate a higher damped and natural frequency when oscillating.

$$\omega_n = \sqrt{\frac{\kappa}{I_{zz}}} \quad (28)$$

$$\omega_d = \sqrt{\frac{\kappa}{I_{zz}} - \left(\frac{b}{2I_{zz}}\right)^2} \quad (29)$$

From Equation 14, there exists a certain value of  $b$  that results in a damped frequency of zero. A system with this damping coefficient is considered critically damped. If  $b$  is below the critical value, the system is underdamped and if  $b$  is above the critical value, the system is considered overdamped. By recording the deflection angle of the plate and plotting the results with respect to time, a FFT assessment can be used again to find the damped frequency at which the plate tends to oscillate.

While failure can occur with enough load cycles in the elastic range, this project is simplified by assuming the failure of the plate at an arbitrary torsional stress limit. The torsional stress of the stop sign post, or flat plate's center, can be calculated in each time step with Equation 30 to then be compared to the stress limit. In this equation,  $c$  and  $H$  are the radius and height of the stop sign post, respectively, while  $G$  is its shear modulus.

$$\tau = \frac{Tc}{J} = \frac{\theta Gc}{H} \quad (30)$$

## CHAPTER 5

### MODELING AND ANALYSIS OF FLUTTER

The purpose of this study is to model a flat plate in flutter. The purpose for using a flat plate is to eliminate the step needed to find the separation points along the body. Assuming the plate to be fixed at its center, simulating a stop sign, simplifies the calculations for the motion of the plate with a second order differential equation. Because flutter is the motion of a body over a period of time, the results must include a measure of time to measure the time to failure, variations in drag force over time, the frequency of the wake, and, most importantly, the deflection angle of the plate with respect to time. By plotting the profile of the flat plate at each time step, the program helps the user to visualize the motion of the plate due to flutter. A copy of the program used for this study is provided in Appendix A.

#### 5.1 Calculations Performed

To develop an iterative simulation of flutter-induced motion, the program consists of a for loop in which all the necessary calculations are completed to update the deflection angle of the plate in each time step. First, the program reads the user's input file and extracts all the necessary information about the system. It calculates a Reynolds number using Equation 16 and sets up thousands of points in the parallel streamlines of the  $w$ -plane. Technically, the Reynolds number is infinite in inviscid flow; the value calculated by the program uses the viscosity of air (or another fluid) as a reference value to obtain merely a pseudo-Reynolds number. These points are later transformed to each of the  $z$ -planes to allow the viewer to visualize the inviscid streamlines around the flat plate in each of the conformal mappings.

In the for loop, time begins at zero seconds and increases by the time increment specified by the user with each iteration until the specified maximum time or until the specimen fails. The program constantly polls for whether a key has been struck to change the screen or display certain points. The deflection angle of the plate which initially begins as zero degrees is updated using Equation 27 and the



corresponding torsional stress in the stop sign post is calculated by Equation 30. The deflection angle is written to a data file for analysis later. Significant points in the flow field include the tips of the plate. They are first plotted in the  $z_5$ -plane where they lie along the  $x$ -axis and are then translated to the other  $z$ -planes using the equations given in Table 2.

Discrete vortices comprise a major portion of the program. The positions of all existing vortices are updated by Equation 14 where each vortex's current velocity is found from Equation 11. Their positions are transformed through all the planes using Table 2 so that the discrete vortices can be plotted in any  $z$ -plane. The updated positions of all the vortices are checked for proximity. Since each vortex has a singularity at its center, they can induce extremely high velocities on neighboring vortices. Realistically, a vortex has a viscous center and two interacting vortices approaching one another would eventually amalgamate into one large vortex. If any two vortices are close together within a  $1/80$  of the plate width, the two vortices are combined into one vortex with a strength equal to the sum of the two vortices' strengths. This value was determined by trial-and-error to eliminate the effect of infinite velocity at the center of each vortex. The strength of all remaining vortices is then decreased by a user-defined dissipation factor. Finally, vortices are checked for their distance from the plate. If they are so far that they have minimal effect on the motion of the plate, they are eliminated from the flow field and their strength becomes zero. To speed up the processing power of the program, garbage collection is performed on the arrays containing the position, velocity, and strength information of all existing vortices in the flow field. If the number of vortices with zero strength reaches 100, these vortices are removed from the arrays and the elements following them are moved up to shorten the arrays and speed up the calculations.

Nascent vortices are introduced into the flow field at each time step. The initial guesses for their positions are at the tips of the plate in the  $z_1$ -plane and offset from the tips' positions in the  $z_4$ -plane. The velocity is found at these points and used to solve for the circulation strengths and release points of each nascent vortex in the  $z_1$ -plane with Equations 12 and 13. The sign of the strength is determined by the sign of the cross product of the complex velocity and position at each tip of the plate. The positions of the nascent

vortices are then transformed through the other  $z$ -planes. The total circulation in the flow field can be calculated as the sum of the strengths of all the existing vortices.

Other significant points including the top and bottom of the cylinder, forward and rearward stagnation points, and the two recently calculated release points of the nascent vortices are defined in all six  $z$ -planes to be plotted for the user's interface experience. The stagnation points are known in the  $z_1$ -plane (at  $x$ -intercepts of the circular profile) and are plotted there first.

A certain number of points is defined by the input file; the points are identified along the front and back surfaces of the circular profile in the  $z_1$ -plane between the tips of the plate. All the points are subsequently transformed through the other  $z$ -planes. These sets of points are also available for viewing in the display window. The distance between the adjacent points in the  $z_6$ -plane is calculated. The velocity and coefficient of pressure at each point are found. Extremely large velocities are set equal to zero to minimize the effect of singularities. Using Equations 19 through 22, the net force in the  $x$  and  $y$  directions as well as the net moment of the fluid acting on the plate are calculated. These values are recorded in an output file.

A pressure probe is placed in the wake downstream of the plate and lies along the  $x$ -axis at a distance from the origin equal to three times the width of the plate. This distance can be varied according to the user's preference as long as it lies in the zone of established flow. The velocity at the probe location is calculated in every time step, converted to a coefficient of pressure, and written to another data file for analysis of the wake.

After plotting the display for that time step, the time is increased by the time increment selected by the user (0.125 seconds in this study) and the program returns to the beginning of the time loop. The calculations are repeated until the torsional stress calculated exceeds the torsional strength limit assigned by the user.

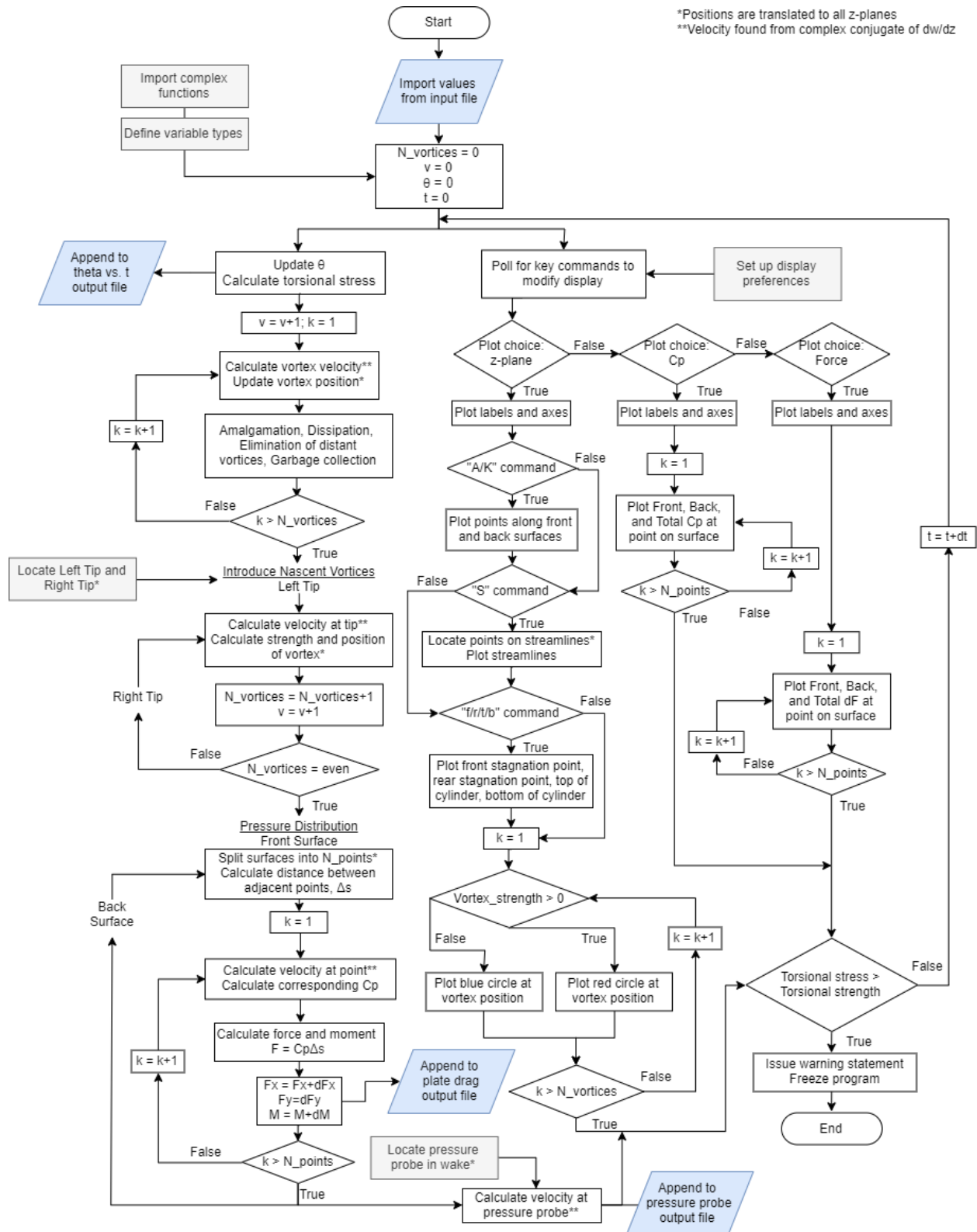


Figure 7 Flow chart of program

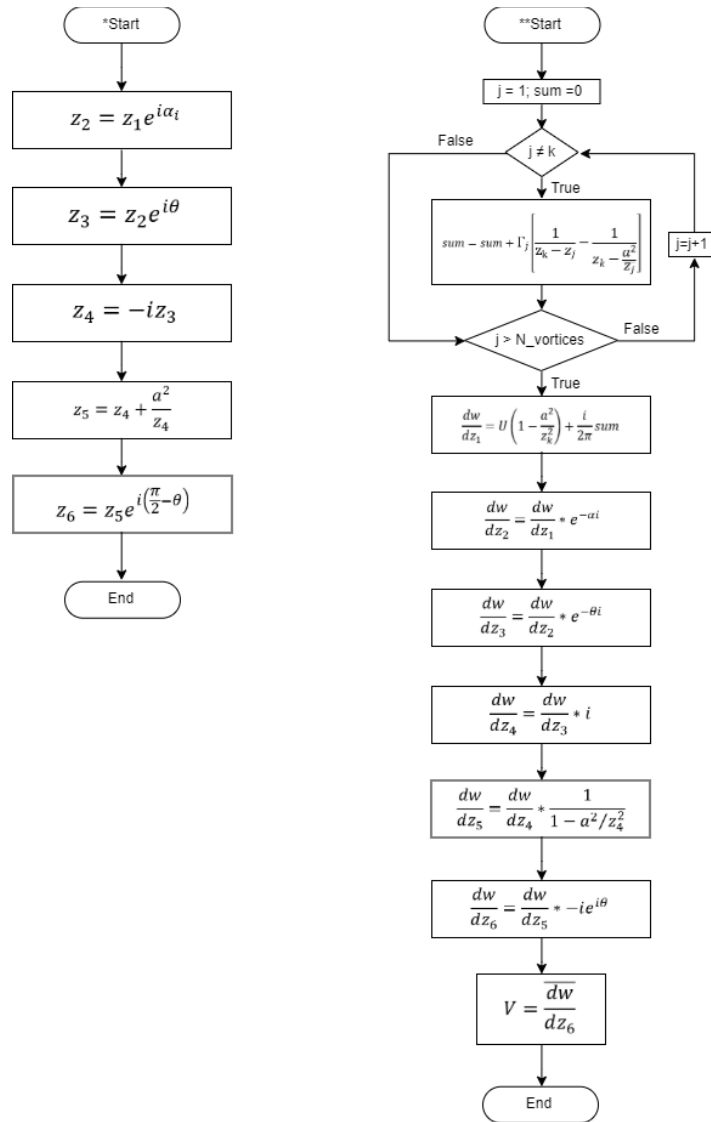


Figure 8 Flow chart of program (continued)

## 5.2 Description of the Program

The program written for this study was written and compiled in FreeBASIC [15]. When it runs, it first reads from an input text file (.txt) in which the user can define properties of the system to be simulated including freestream velocity, plate width, torsional stiffness, damping, mass moment of inertia, and more. An example of an input file is given in Appendix B. Once compiled, the program becomes an application which can be executed. As soon as it runs, the simulation time begins from zero and increases steadily as the program cycles through its “for” loop. A window opens and displays the physical  $z_0$  plane (as specified by the user in the input file) and has the capacity to plot the inviscid streamlines in the vicinity of the body as well as plot the locations of all the discrete vortices in the flow field. Each vortex is indicated by a circle of size proportional to its strength and color proportional to its direction. Not only does this window help the user visualize the physical motion of the plate but also of the air flow around and behind it. The user can change the  $z$  plane by striking the number key corresponding to the desired plane. The user can also locate significant points including stagnation points and nascent vortex release points by striking certain keys. The top left corner of the window displays the current plate properties,  $z$  plane, and time. A sample of this window is shown in Figure 9.

The program monitors interactive key strokes to change the information displayed on the screen. Table 4 contains a list of these interactive keys. If the user strikes the key, “C”, the window changes to display the pressure distribution plot. This plot has an  $x$ -axis corresponding to the radius,  $r$ , from the center of the plate and a  $y$ -axis corresponding to the coefficient of pressure,  $C_p$ , at each location. Three lines are plotted on this graph. The cyan line demonstrates the pressure distribution on the front surface. It is expected that the highest  $C_p$  correspond to the stagnation point on the front surface. The red line demonstrates the pressure distribution on the back surface. Positive values of  $C_p$  indicate the flow pushing the plate in the negative  $x$ -direction. Finally, a green line is the sum of the pressure distributions on the front and back surfaces to demonstrate the total load subjected to the plate. In the top, left corner of the window, the total force in the  $x$  and  $y$ -directions is given as well as the net moment. These can also be visualized by the shape of the green curve. If the user strikes the key, “F”, the window changes to display the force distribution

plot. This plot is largely similar to the pressure distribution plot except for some adjustments due to the discrepancies in the length of each line segment over which force is exerted. The forces on the surface are each acting on a short, linear segment which is part of the circular surface in the  $z_1$ -plane. These surface segments of length  $\Delta s$  are mapped into the  $z_6$ -plane onto the flat plate, but the conformal mapping is nonlinear. As a result, segments in the  $z_6$ -plane along the surface are not of uniform length. Due to the similarities between the force and pressure plots, they will be used interchangeably throughout Chapter 6 of this study. Other keystrokes allow the user to locate key points in the flow field, visualize inviscid streamlines, and observe the motion of the discrete vortices in all six of the  $z$ -planes.

**Table 4 Program Interactive Key Functions**

<b>Key</b>	<b>Name</b>	<b>Function</b>
1	$z_1$ -plane	Switch view to the $z_1$ -plane
2	$z_2$ -plane	Switch view to the $z_2$ -plane
3	$z_3$ -plane	Switch view to the $z_3$ -plane
4	$z_4$ -plane	Switch view to the $z_4$ -plane
5	$z_5$ -plane	Switch view to the $z_5$ -plane
6	$z_6$ -plane	Switch view to the $z_6$ -plane
C	$C_p$	Display pressure distribution on each side of the flat plate (and the total)
F	Force	Display the distribution of force on each side of the flat plate (and the total)
a	Axis	Turn the axes on/off the $z$ -plane
s	Sleep	Pause the program
f	Forward stagnation point	Turn on/off colored circle identifying forward stagnation point in $z$ -plane and $C_p$ and F plots
r	Rear stagnation point	Turn on/off colored circle identifying rear stagnation point in $z$ -plane and $C_p$ and F plots
t	Top	Turn on/off colored circle identifying top of circle in $z$ -plane
b	Bottom	Turn on/off colored circle identifying bottom of circle in $z$ -plane
L	Left tip	Turn on/off colored circle identifying left tip of flat plate in $z$ -plane
R	Right tip	Turn on/off colored circle identifying right tip of flat plate in $z$ -plane
P	Left release point	Turn on/off colored circle identifying vortex release point from the left tip of flat plate in $z$ -plane
Z	Right release point	Turn on/off colored circle identifying vortex release point from the right tip of flat plate in $z$ -plane
A	Front surface	Turn on/off colored line identifying front surface of flat plate in $z$ -plane
K	Back surface	Turn on/off colored line identifying back surface of flat plate in $z$ -plane
S	Streamlines	Turn on/off pure inviscid flow streamlines in $z$ -plane
V	Vortices	Switch style of vortex identifiers: colored circles or plus/minus signs
q	Quit	Exit the program

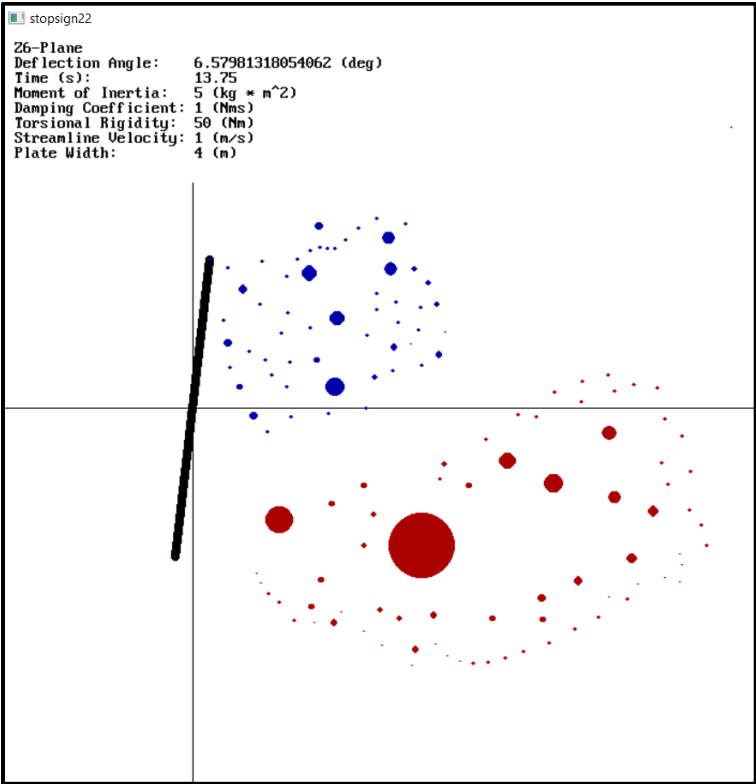
When the torsional stress in the stop sign post exceeds the torsional strength assigned to it in the input file, the program terminates and issues a warning statement to the user that the stop sign has failed. The user can continue to advance the program with the “s” key or exit the program with the “q” key.

With each iteration of the main program loop, the program appends data to several text files. Each of these files includes the time step and data of interest. The first file outputs the deflection angle against time. These data points can be graphed and investigated with a Fast Fourier Transform (FFT) package provided in Microsoft Excel or downloaded for LibreOffice Calc [16]. The Microsoft Excel package only accepts data sets with a lengths equal to powers of two and containing up to 4096 data points. While the LibreOffice Calc package accepts arrays of different lengths, it uses a Discrete Fourier Transform (DFT), an alternative algorithm that takes far longer to calculate the power spectral density of the set. An autospectrum produced by FFT can be used to find the resonant frequency of the structure just before it fails in addition to the Strouhal frequency of the wake. The second file outputs the total force in the x direction of the plate. This is also known as drag force which has been plotted with respect to time in several other papers. These results can also be observed to find the Strouhal number of the wake. The last data file includes the coefficient of pressure at the pressure probe’s location in each time step. This plot will provide a more accurate value for the Strouhal number of the wake.

The computation time to run the program is extremely fast and occurs in almost real time. Depending on the processing unit, the compiler takes a matter of seconds and when the window opens, the plate and vortex positions are updated several times every second. As more vortices are introduced into the flow field, the number of calculations that are performed increases and the program slows down. Amalgamation and elimination of vortices improves the speed of the program significantly.

To run multiple iterations of the program for results, a batch file (.bat) was created. A sample of the batch file is shown in Appendix D. For each test scenario, an input file was generated and put through the program. The program was modified to close instead of paused upon failure of the flat plate modeled so that each case could be executed subsequently in the absence of the user. During each time step, the deflection angle, forces in the x and y direction, net moment, and coefficient of pressure were recorded

together in the same output text file to simplify the data reduction process. From these data files, plots were created for deflection angle, drag force, and pressure over time. The failure times and peaks in the FFT plots for the deflection angle, drag force, and pressure coefficient were extracted for comparison and observations.



**Figure 9 Physical flow field (main screen)**



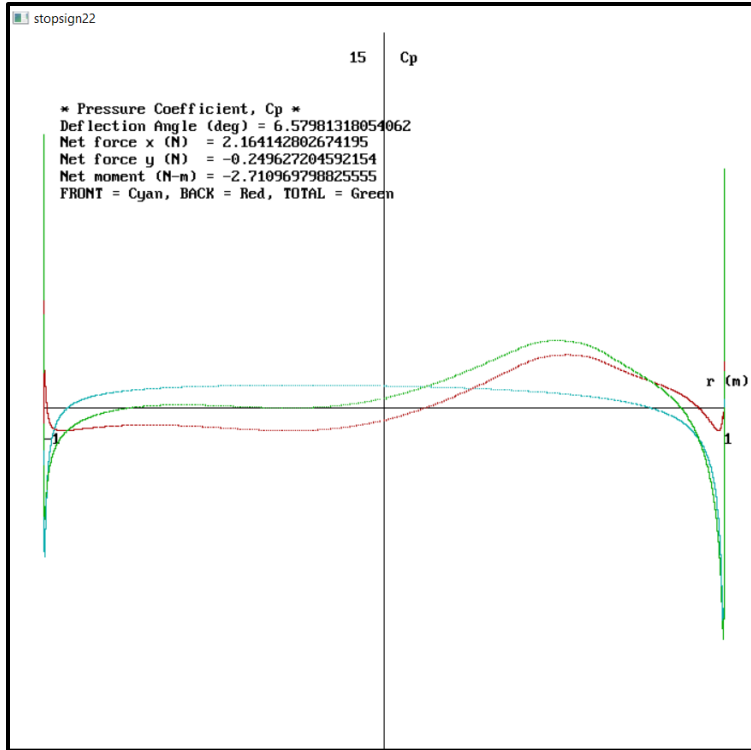


Figure 10 Pressure distribution plot

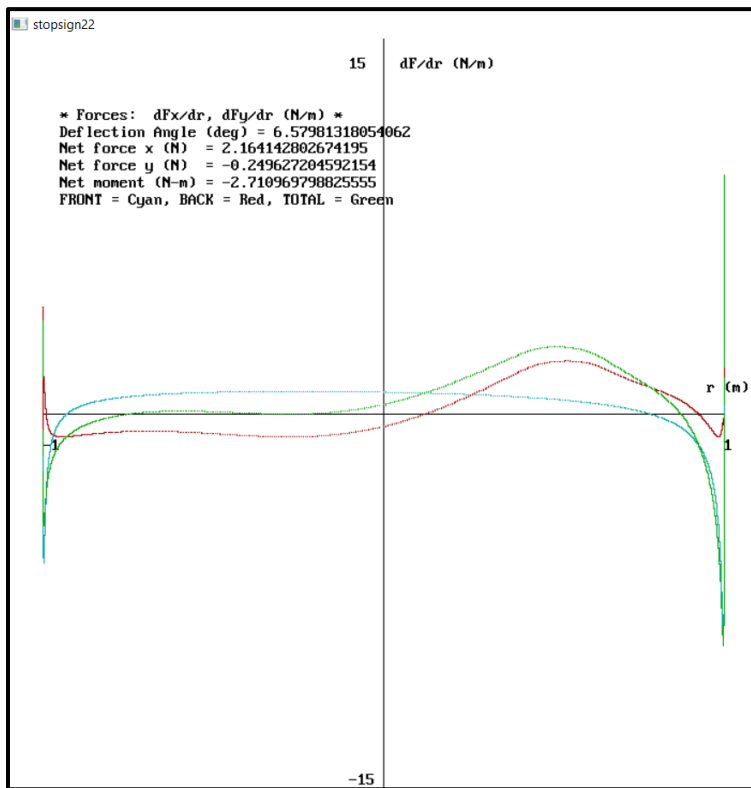


Figure 11 Force distribution plot

## CHAPTER 6

### RESULTS

Information was gathered from the program throughout its creation. By observing the output of the program at each step as it was being built, the creators were able to verify that the program was providing realistic and logical results. Following the theoretical approach to the problem, the results can also be broken into three sections: inviscid flow, DVM, and dynamic motion (or flutter). Results of this study comprise primarily of observations of the plots produced by the program including the flow field in each of the  $z$ -planes as well as the force and pressure distribution plots. Other data can be analyzed including the output files providing information on the wake generated behind the plate and motion of the plate due to flutter.

#### 6.1 Inviscid Flow Results

The first results were obtained before the addition of the discrete vortices into the flow field. By setting up the streamlines in the  $w$ -plane and transforming them to each of the  $z$ -planes along with significant points, inviscid flow results are obtained for analysis. The same conformal mapping of a flat plate at a deflection angle of -20 degrees and an angle of attack of 30 degrees from Table 2 is shown in Table 5 with all the relevant, key points. Here, the bright cyan line represents the front of the cylinder and the yellow line the back of the cylinder. The light and dark blue circles represent the top and bottom of the cylinder, respectively, and are clearly placed appropriately as observed in the  $z_1$ -plane. Meanwhile, the light and dark green circles represent the left and right tips of the plate, respectively, supported by their locations in the  $z_5$ -plane and the joints of the front and back surfaces of the plate. The red circle indicates the rearward stagnation point and the dark cyan circle represents the forward stagnation points. The locations of these points are logical because there are streamlines that terminate at these points when they meet the profile streamline. Thousands of small blue dots are plotted to represent the streamline and demonstrate the inviscid flow pattern of a fluid past a flat plate. The smooth, clean lines displayed in the  $z_6$ -plane are indicative of

inviscid flow and a lack of boundary-layer separation. The flow inside the circular cylinder is shown in Figure 12 in the  $z_1$ -plane, but since it is not used in the analysis, it is not displayed in the other five planes.

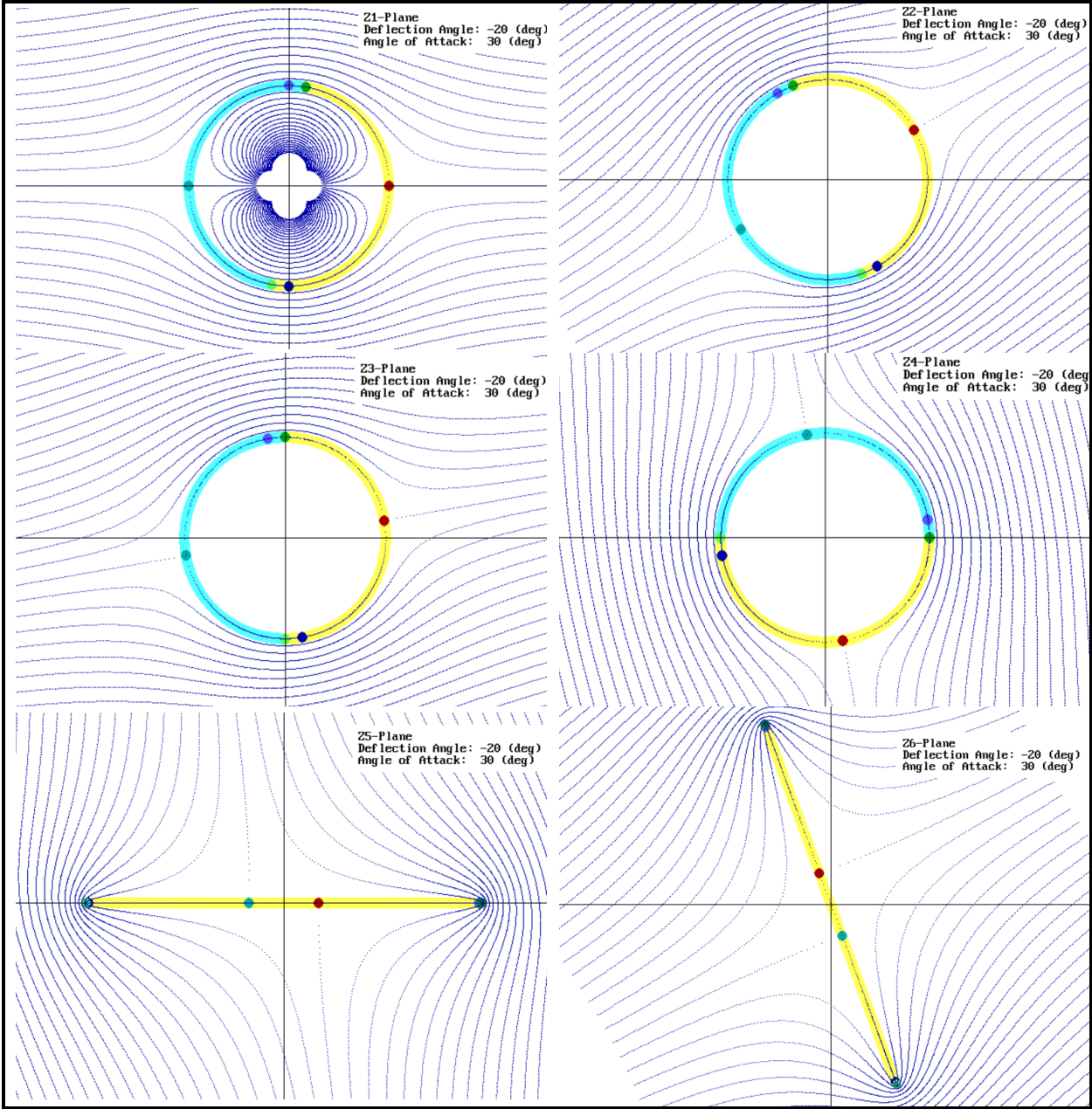


Figure 12 Locations of key points in each  $z$ -plane

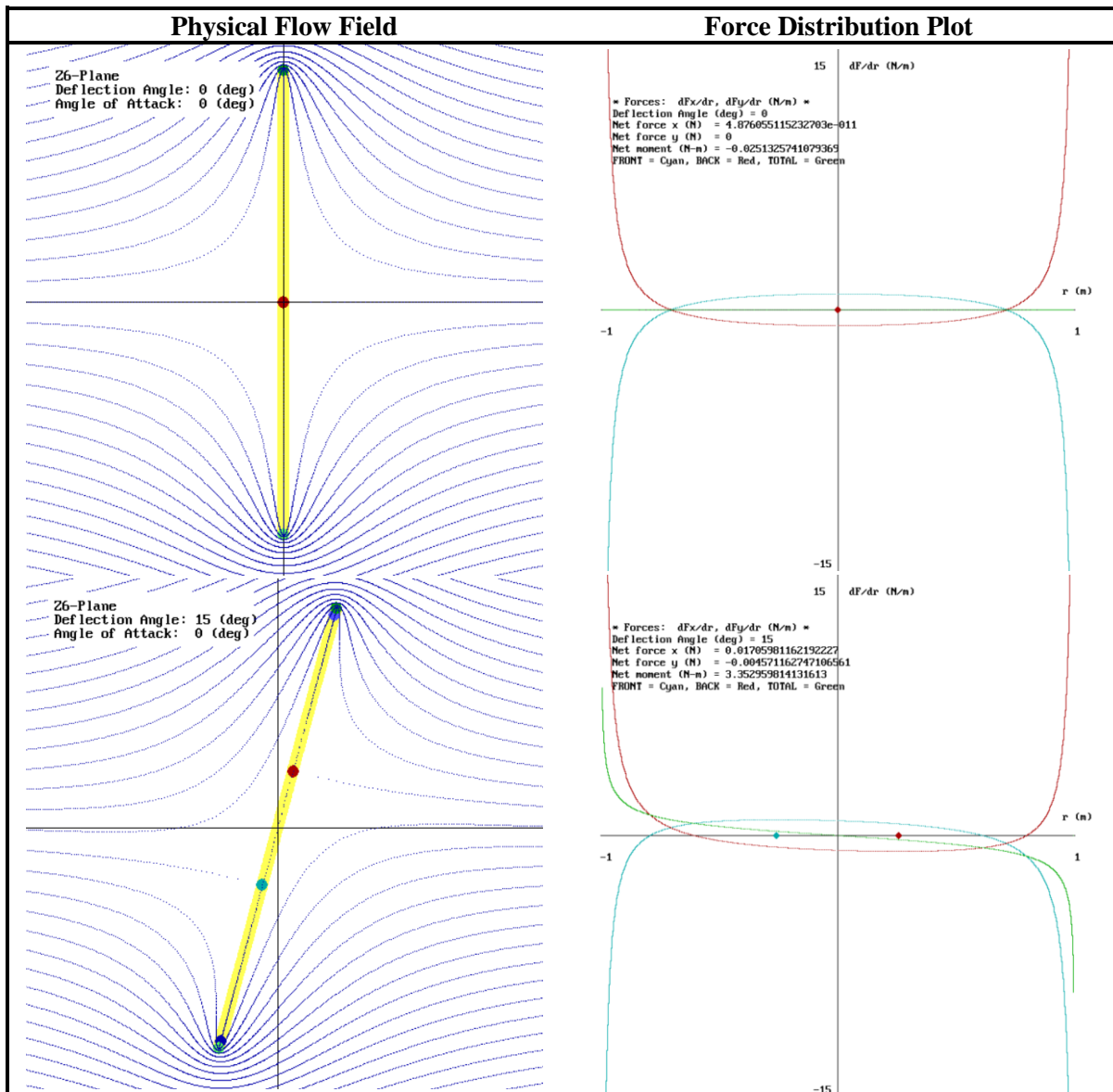
Validation of the proper execution of inviscid flow theory is also provided by the plots of the coefficient of pressure and the forces along the front and back of the flat plate with respect to location along the plate. The  $z_6$ -plane for flat plates at various deflection angles is shown alongside its associated force distribution plot. The stagnation points are shown in both diagrams. It is evident that the stagnation points on the front and back surfaces of the plate correspond with the maxima and minima of the force distributions on their corresponding surfaces. This is logical because stagnation points indicate a location where the velocity of the fluid is zero and the coefficient of pressure is subsequently one, the maximum value of pressure, and therefore, force. The forward stagnation point (cyan) corresponds to the maximum of the cyan pressure curve while the rearward stagnation point (red) corresponds to the minimum of the red pressure curve because the force on the back is plotted upside down as to help the viewer visualize the total force in the  $x$ -direction.

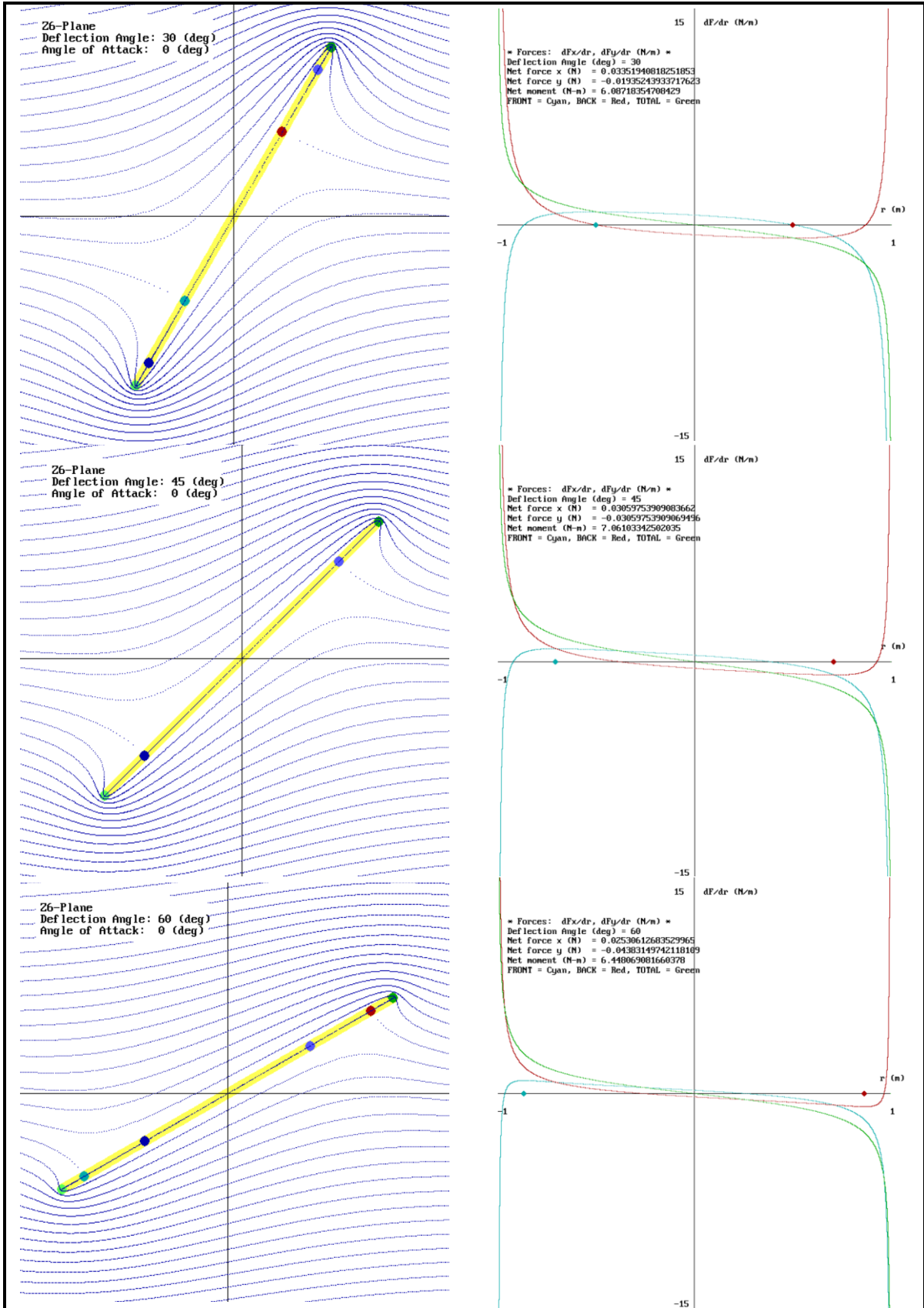
As the fluid goes around the tips of the plate, it speeds up resulting in very low, negative pressure and force values. This is demonstrated in the pressure distribution plots as the front surface and back surface curves approach negative and positive infinity, respectively, at the tips of the plate. In the  $z_5$  and  $z_6$ -planes, there are singularities at the tips of the plate. As a result of these infinite velocities, it is impossible to calculate the circulation strengths and positions of nascent discrete vortices using Equations 12 and 13. To work around this issue, the initial guesses for the nascent vortex locations are not chosen immediately at the plate tips, but at a distance just offset from them along the axis of the plate. Various offset distances were tested, and the results revealed that this value has little impact on the general behavior of the wake behind the plate when DVM was applied to the program. An offset distance of  $0.25a$  outward from the tips of the plate was selected to obtain the best results.

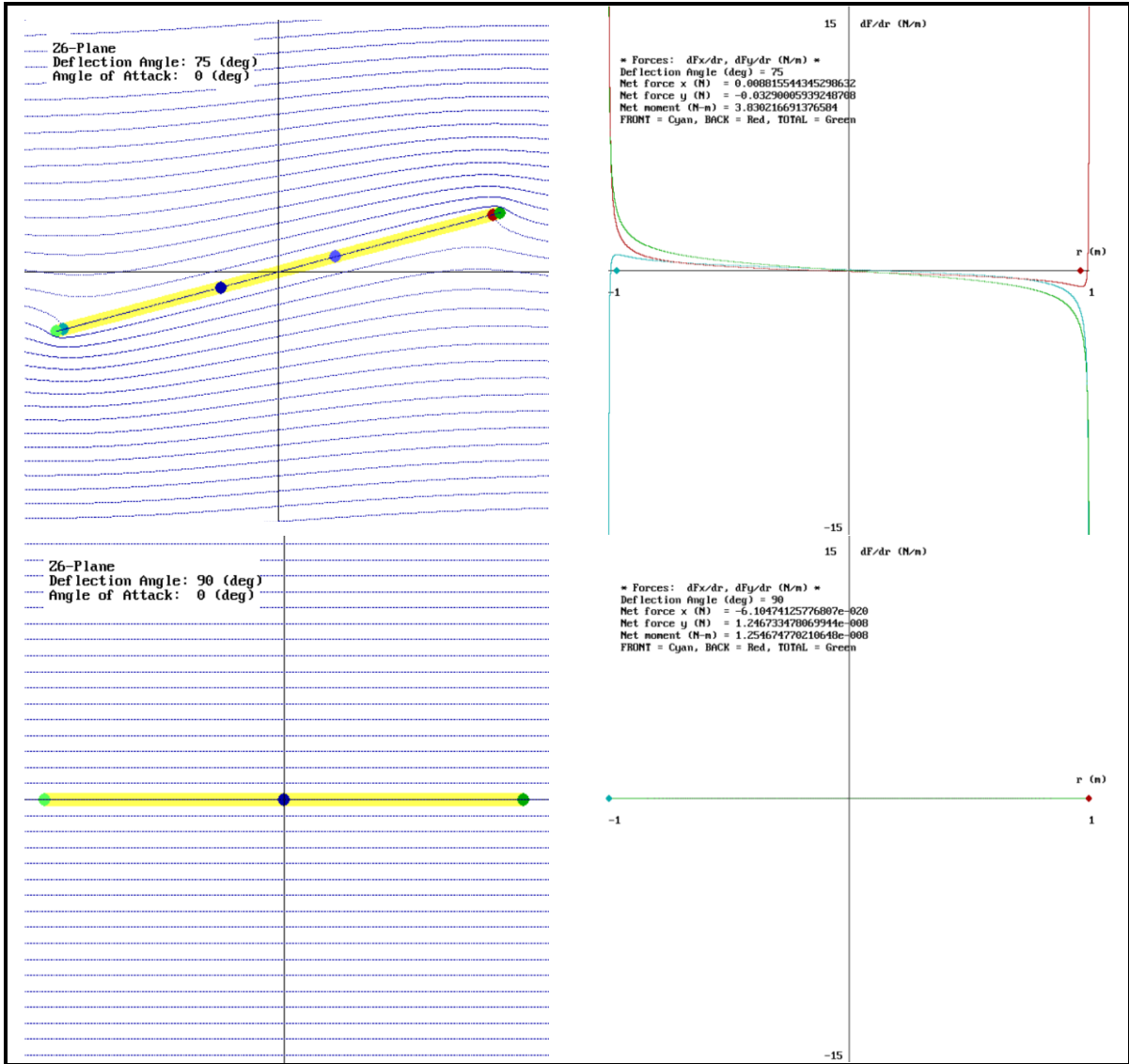
It is also interesting to note that the net forces calculated and displayed in the upper left corner of the force distribution plot always remains very close to zero in pure inviscid flow with no discrete vortices. Accordingly, the area under the green curve corresponds to the net force in the  $x$ -direction and is likely zero due to its symmetry about the origin. A zero net force is to be expected as the flow does not separate and generate a wake behind the plate creating drag, as it would in real life. On the other hand, the net moment

acting on the plate is not zero at all deflection angles, only at angles of zero and ninety degrees. This can be observed by the asymmetry of the green curve about the y-axis in the force distribution plots as well as the calculation of the net moment displayed in the upper left corners of the plots. All the aerodynamic forces and moments cancel out at these angles; without net forces and moments, a flat plate in inviscid flow would remain idle and not be subject to the effects of flutter. Inviscid flow without vortex shedding is an idealistic case and does not yield real-world results.

**Table 5 Inviscid Flow Pressure Distribution Plots**







## 6.2 Discrete Vortex Results

Once the inviscid flow and conformal techniques proved to be working properly, the next step was to include DVM into the program. For the DVM results, the display of streamlines were turned off because they were too complicated to calculate from the complex potential equation, Equation 10. Instead, the flow can be visualized through the positions of the vortices in each time step. Watching the program run allows the user to envision the movement of the air in almost real time. Results were examined for the case of a stationary plate perpendicular to the flow to simplify analysis.

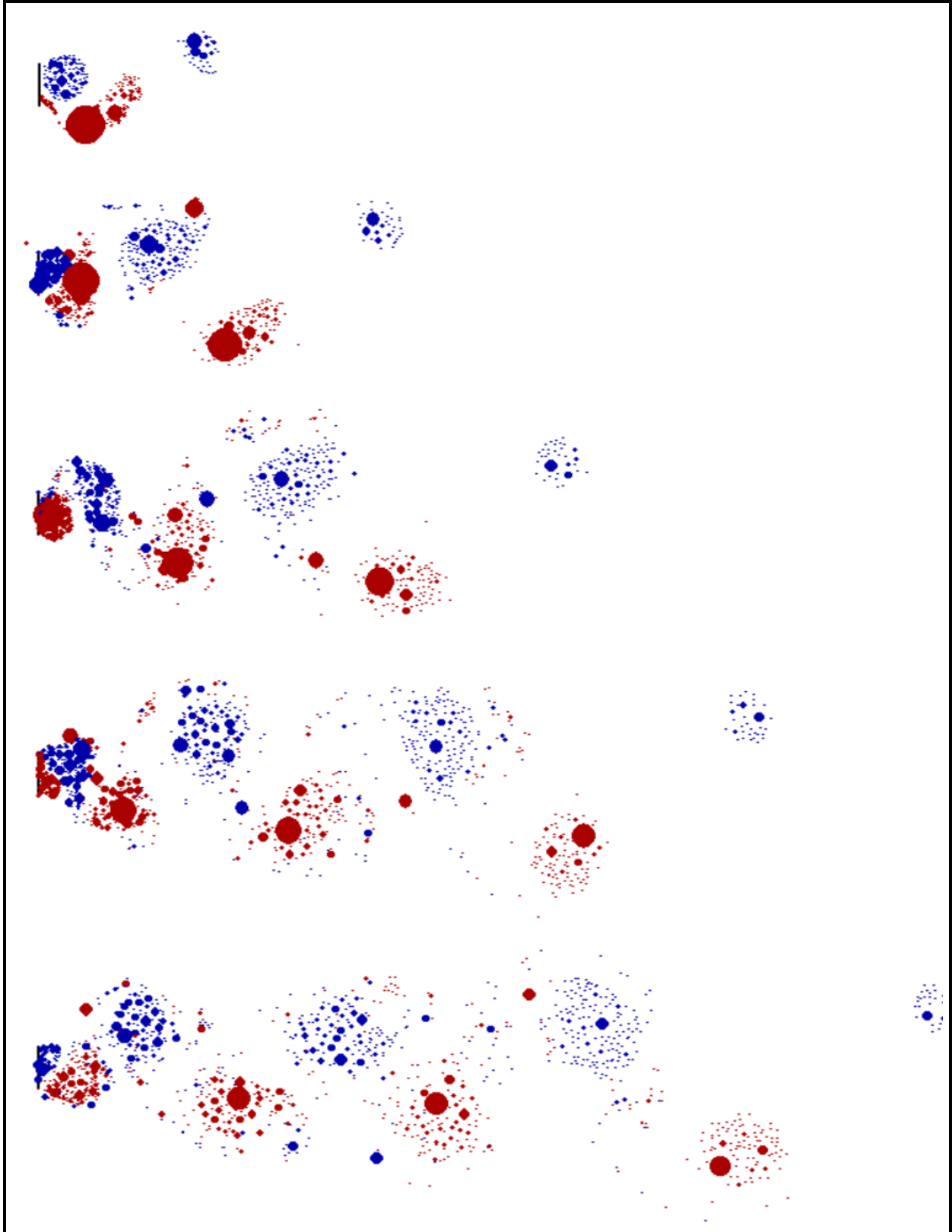


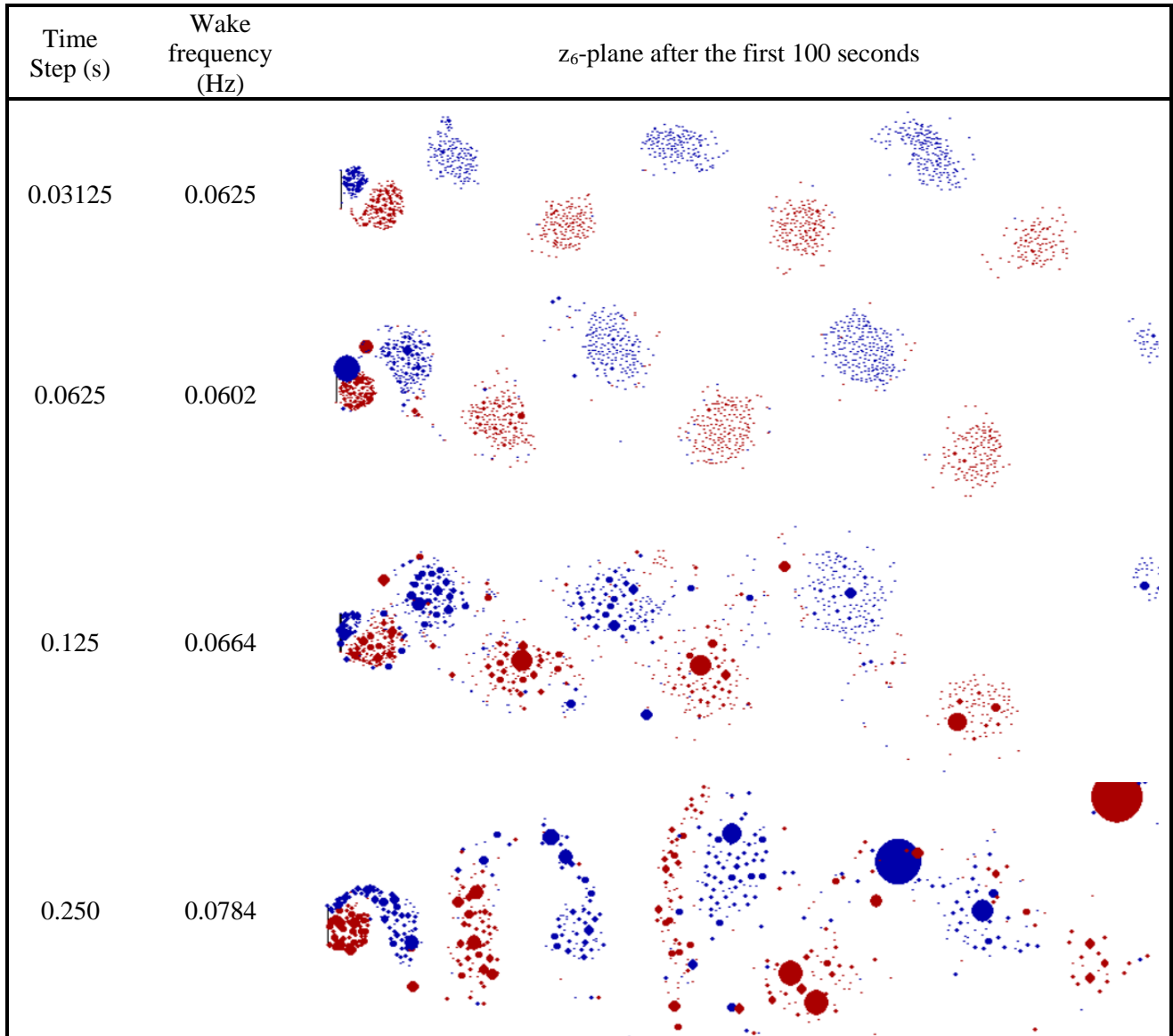
Figure 13 Progression of the wake using DVM



A progression of the flow field in the first 100 seconds of impulsively-started flow is shown at 20 second intervals Figure 13. The flat plate can be seen at the left of the flow field. Each vortex is represented by a filled circle. The blue circles represent vortices rotating clockwise and the red circles represent vortices spinning counterclockwise. The diameter of each circle is directly proportional to its corresponding vortex's circulation strength. The generation of the rotating pockets of vortices and the alternating pattern is very reminiscent of results obtained in previous studies as well as the natural phenomenon of von Kármán vortex streets. The patterns developed in this numerical simulation are very realistic and support the notion that the equations used to implement DVM on the system is a valid technique for modeling fluid flow. Another indicator of a functioning program is the calculation of the total circulation. The circulation of all the discrete vortices was summed up and displayed on the screen for observation and it remained very close to zero at all times, satisfying the Kelvin Circulation Theorem.

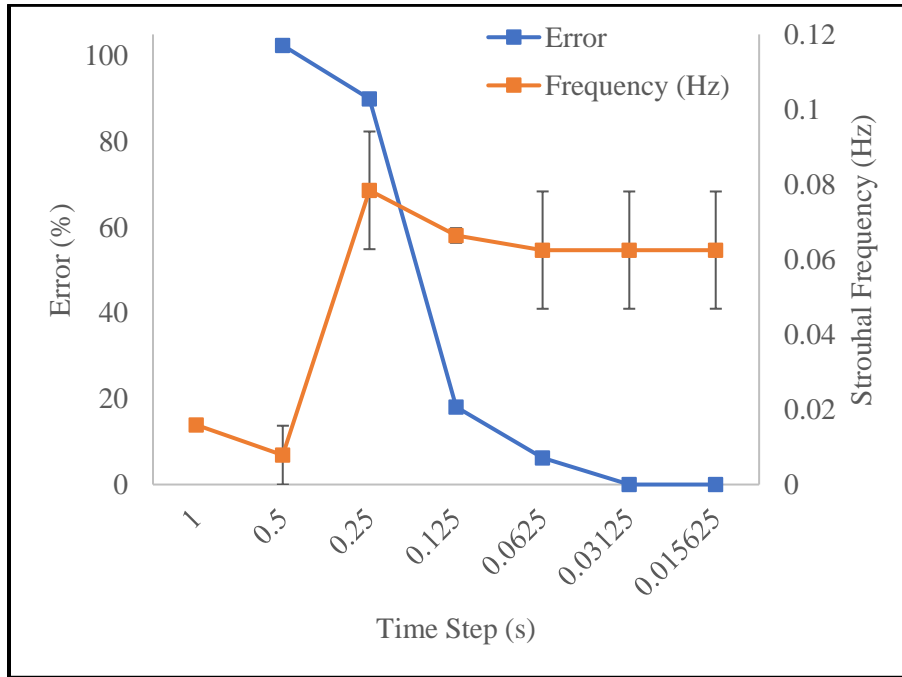
The behavior of the wake was observed for various time steps. Though 0.125 seconds was the time step selected by Sarpkaya [6] for accurate wake generation, verifying the time step was considered in this study as well. From Table 6, it is clear that a smaller time step results in more, smaller vortices while choice of a larger time step results in very large vortices that throw off the clarity of the flow field and hinders visual observation. Sarpkaya's recommended time step of 0.125 seconds provides a mixture of both large and small vortices and is the option that runs the fastest while still maintaining a clear picture and an accurate wake frequency. The time step 0.125 seconds will continue to be used as it is the point after which the precision error levels off at zero in Figure 14. Overall, the different time steps tested provided similar numerical results for wake frequency as well as similar  $z_6$ -planes.

**Table 6 Time Step Comparison**



The error between the different time steps is demonstrated in the Figure 14. The steady decline in the error indicates that once it levels off, it can be assumed that the time step results in an accurate simulation of the wake flow field. When run with time steps greater than 0.125 seconds, the Strouhal frequencies calculated were relatively far from one another. On the other hand, the same Strouhal frequency was consistently found in the wake when the program was run with time steps of 0.125 seconds or less. The repeatability demonstrated with these time steps shows that 0.125 seconds is the cutoff for obtaining

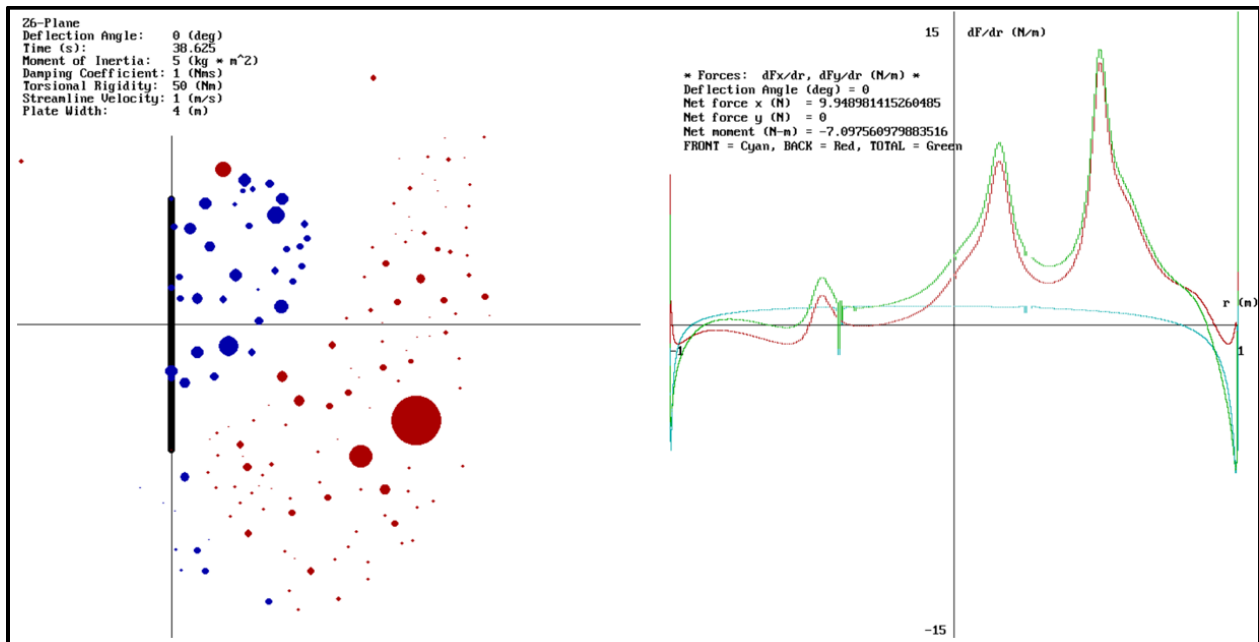
accurate results for Strouhal number. Any of the time steps tested below 0.125 seconds could be used for a valid model of the wake but 0.125 seconds is chosen because it generates results the fastest while maintaining accuracy.



**Figure 14 Time step Strouhal frequency error**

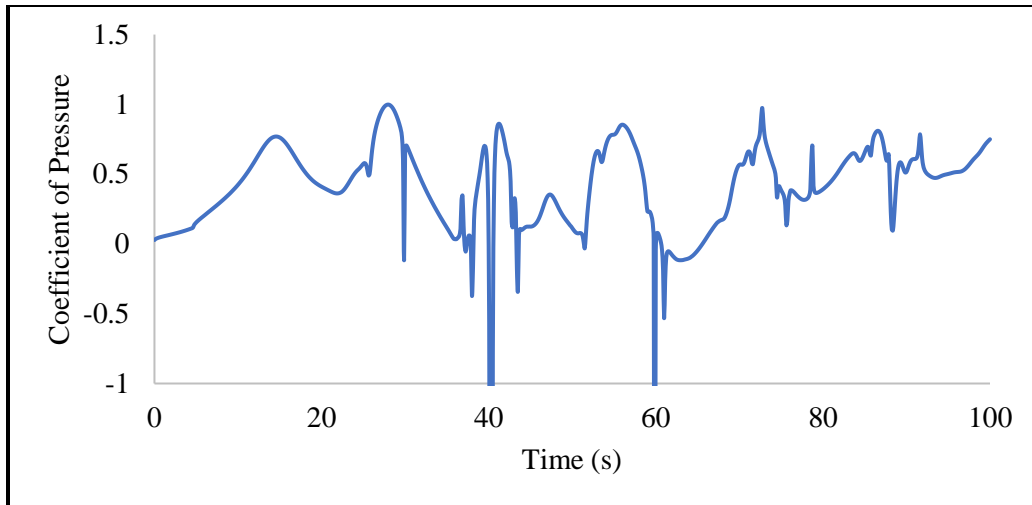
By displaying the force distribution in the window, it is easy to see the effect of the discrete vortices on the aerodynamic forces acting on the plate. In Figure 15, the top half of the plate in the  $z_6$ -plane corresponds to the right half of the  $x$ -axis in the force distribution plot. The abundance of large blue circles on the top of the plate is reflected in the peaks on the right side of the force plot. It is important to note that the cyan curve in the force plot is relatively unchanged from its shape before the addition of the discrete vortices. This makes sense because the vortices lie behind the plate and therefore do not impact the front surface. The peaks are positive on the curve, indicating that they are pulling the plate to the right of the screen which makes sense because separation of flow creates drag and would pull the plate backwards. The total force acting on the plate in the  $x$ -direction is equal to the area under the green curve in the force plot

and is also calculated and displayed at the top of the screen. The drag force is positive and the net force in the y-direction remains zero corresponding with the zero angle of attack and the direction of airflow past the plate. The asymmetry of the green curve is reflected in the net moment calculation which is nonzero. This calculation will become essential in finding the deflection angle of a moving plate in the third and final part of this study. Aberrations in the force curves may be attributed to discrete vortices very close to the surfaces of the plate, inducing infinite velocities and forces. The force at the tips of the plate is still infinite due to the singularities at these locations.



**Figure 15 Force distribution plot with DVM**

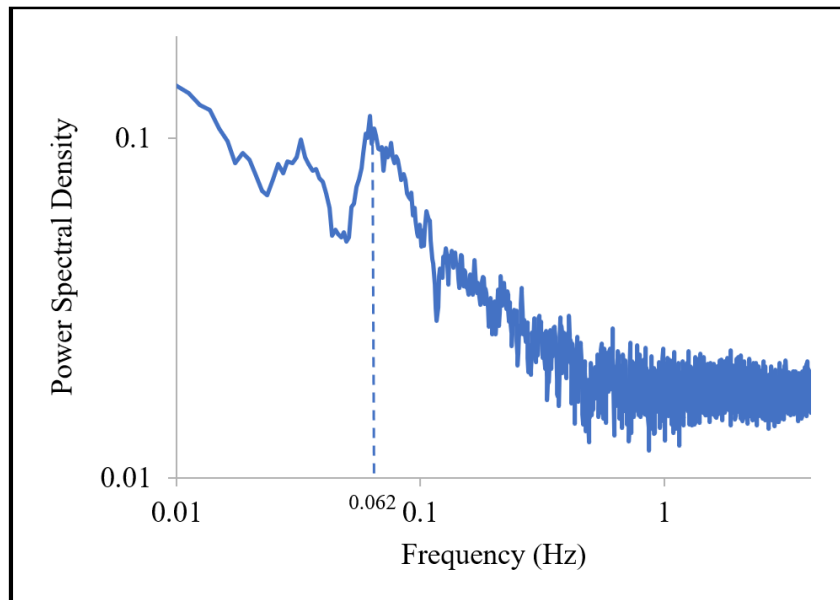
The data files from this iteration of the program were extracted and plotted in Microsoft Excel. The pressure probe placed out into the wake measured the pressure at that point during every time step; the following oscillatory pattern of pressure measurements is shown in Figure 16. The large negative values on the plot can be disregarded as they are caused by vortices and their singularity points moving directly over the probe providing unrealistic data at certain time steps.



**Figure 16 Pressure probe measurement for stationary plate**

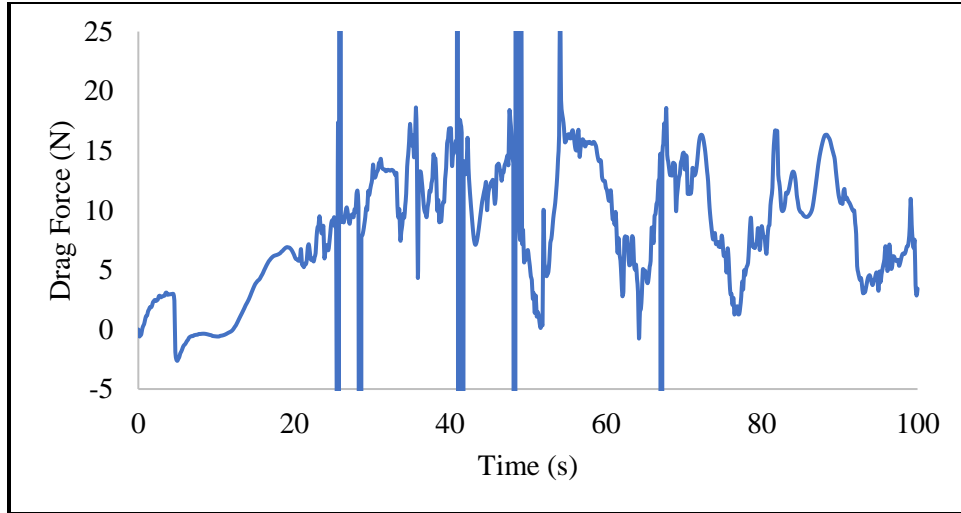
There is a clear characteristic frequency to this wake which can be used to describe the wake's behavior. Known as the Strouhal frequency, this value can be found by measuring the distance between the pockets of vortices or by simply analyzing the data in a Fast Fourier Transform (FFT) package and finding the peak in the autospectrum. For a cleaner autospectrum, the program is run for a duration of 512 seconds though only the first 100 are shown in the time plots. This provides the largest number of data points that the Excel FFT package can analyze. A higher number of iterations exaggerates the peaks in the autospectrum due to the multiples of peaks aligning at the Strouhal frequency. Furthermore, the curves can be smoothed by using a rolling average technique to clean up the graph and allow the user to identify the peak more easily. Sometimes, the data is the cleanest in the middle, beginning, or near the end of the data set. By visual observation of the  $C_p$  versus time plot, a smaller set of data can be selected and analyzed faster and with cleaner results for identifying the peak in the autospectrum. However, choosing a smaller data segment results in a greater frequency interval and therefore a coarser resolution of the autospectrum. For elegance, all the autospectra plotted to obtain Strouhal numbers and damped frequencies from the simulation results will not be shown in the main body of the thesis but are provided in Appendix E for support of the Strouhal frequencies and numbers discussed in this study. One example of an autospectrum is shown in Figure 17.

The peak on the power spectral density curve for the data set of the fixed plate indicates that the wake behind the flat plate has a characteristic frequency of 0.062 Hz (see the dashed line in Figure 17) which corresponds to a Strouhal number of 0.248 at a pseudo-Reynolds number of 265,604, based on the width of the flat plate and the uniform velocity of the fluid. These values match the plot of the St-Re relationship in Figure 5 and provide additional support for the accuracy of the wake produced using DVM.



**Figure 17 Autospectrum for pressure probe behind stationary plate**

The Strouhal frequency of the wake is also reflected in the plot of the drag force acting on the plate. The variable nature of this force supports proper execution of DVM and compares well with the results of previous works on this topic. The force oscillates due to the alternating of the rotating pockets of vortices. Unlike in the inviscid flow case where the drag force on the plate was zero, the drag force on a plate with a wake behind it is almost always positive, pulling the plate in the direction of the uniform velocity. Again, the extreme values on this plot can be overlooked and attributed to singularities at the center of each vortex becoming too close to the surfaces of the plate.

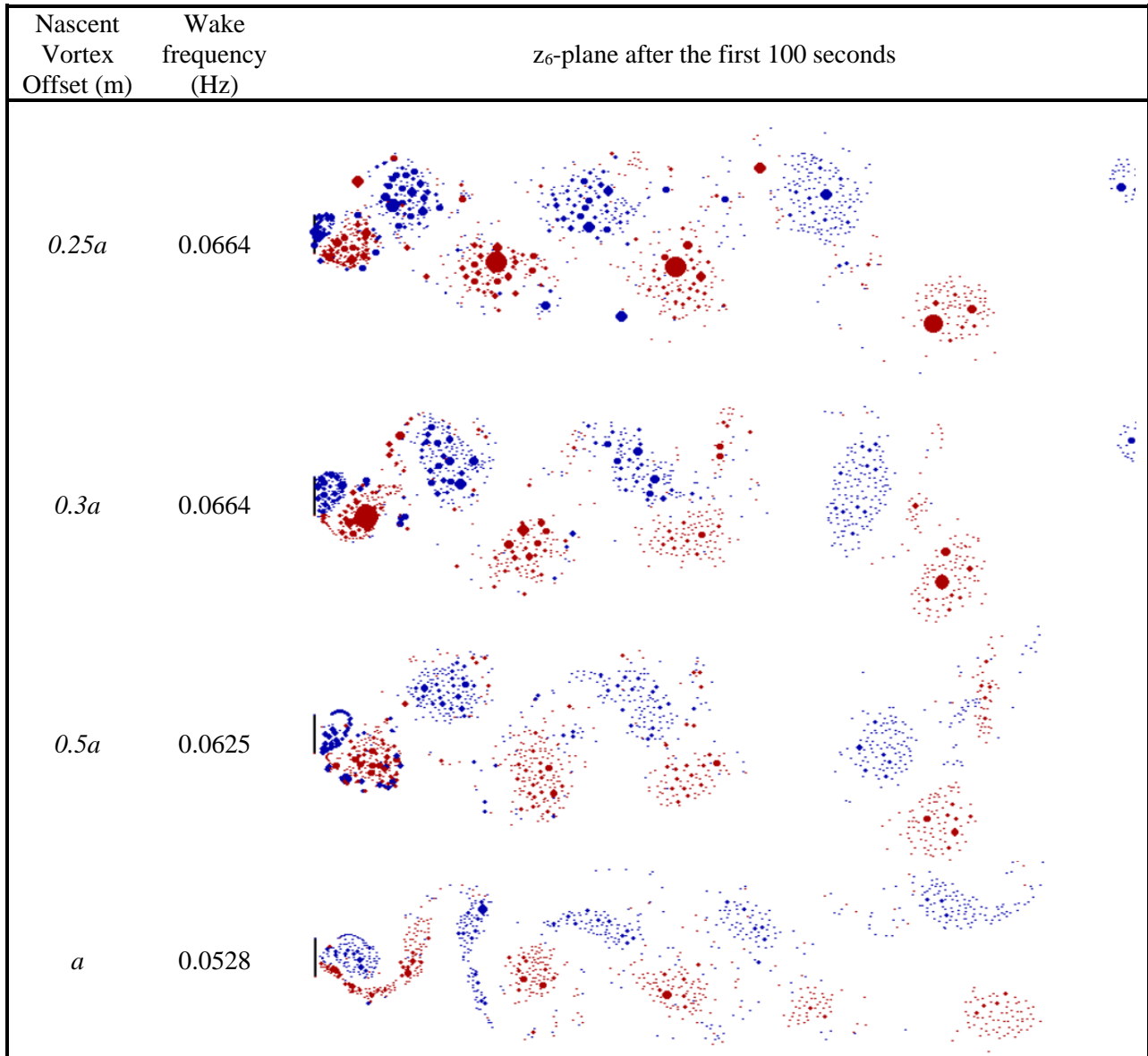


**Figure 18 Drag force on stationary plate**

As previously mentioned, the initial guess for the locations of the nascent vortices was offset from the tips of the plate by a factor of  $0.25a$ . The selection of this value can be supported by the results shown in Table 7. A comparison of the flow field and the Strouhal frequency of the wake shows that after enough time elapses, the value chosen to offset the nascent vortices does not affect the wake behavior. However, it is important to pick a distance far enough from the plate such that the circulation strengths of the nascent vortices are not unreasonably large due to high velocities near the tips of the plate. The pseudo-Reynolds number is calculated using Equation 16 with a kinematic viscosity of  $1.506E-5 \text{ m}^2/\text{s}$ , the viscosity of air at an assumed temperature of  $20^\circ\text{C}$  [17]. For a Reynolds number of 265,604,  $0.25a$  is the smallest nascent vortex offset distance that prevents this undesirable effect at the beginning of the time loop. It is also important to pick a distance close enough to the plate to not distort the flow field from its true form and result in skewed data used to characterize the wake frequency. The flow field for an offset distance of  $a$  shows the pockets of vortices closer together resulting in a slightly higher Strouhal frequency than the other test cases show. The discrepancies in the wake frequencies are relatively negligible as the frequency increment of the autospectra used to find them is  $0.001954 \text{ Hz}$ . If the values were able to be found to a higher precision, perhaps they would be closer together. Overall, the effect of the wake downstream of the

plate has little impact on the aerodynamic forces acting on the plate. This shows that the choice of this parameter is not vitally important to the results produced by the program.

**Table 7 Nascent Vortex Offset Distance Comparison**

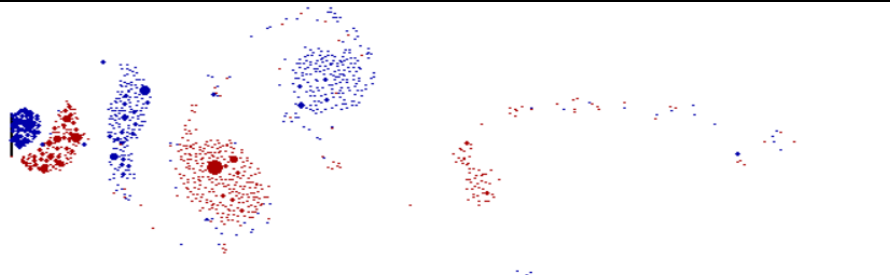
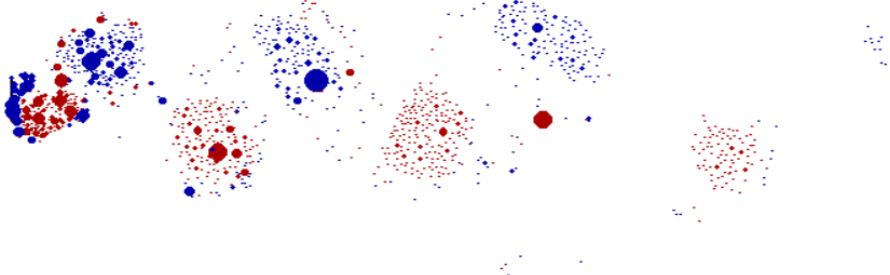
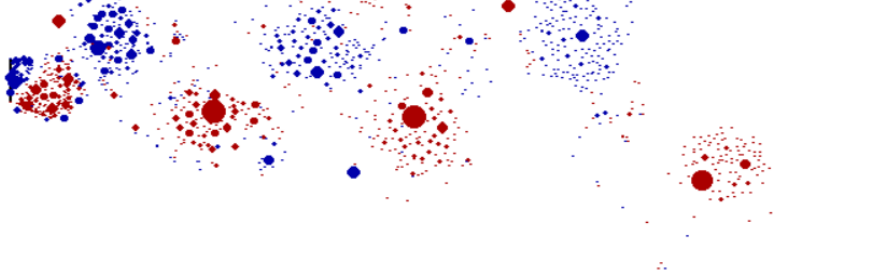
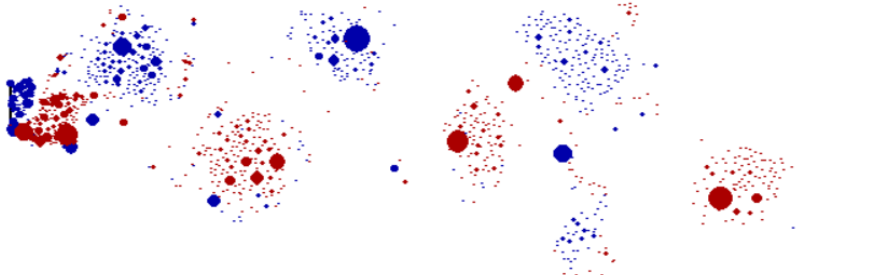


To quickly trigger asymmetry in the wake, vortices shed from the left tip were assigned zero strength for the first 75 time steps. Running the program for a pseudo-Reynolds number of 265,604 with



variations of this parameter showed that the results had little influence on the Strouhal frequency of the wake but choosing a low number resulted in the program taking a longer time to generate the desired wake and choosing a high of a number resulted in a large gap between the first pocket of vortices and its successor.

**Table 8 Number of Time Steps to Trigger Asymmetry Comparison**

Number of Time Steps	Wake frequency (Hz)	$z_6$ -plane after the first 100 seconds
25	0.0625	
50	0.0664	
75	0.0664	
100	0.0625	

Other issues that arose with the program included vortices passing across the plate due to too large of a time step. This was accounted for by simply eliminating vortices that passed through the plate from the flow field. Another issue was that unrealistically large nascent vortices would be introduced in the flow field due to high velocities near the tips. While the  $0.25a$  offset helped mitigate the problem at the beginning of the time loop, it still occurred at intermittent time steps in which the proximity of existing vortices to the tips of the plate resulted in large nascent vortices. To combat this issue, these vortices were checked and adjusted to match the strength of their precursor, assuming the previous vortex's strength was of reasonable size for that point in time.

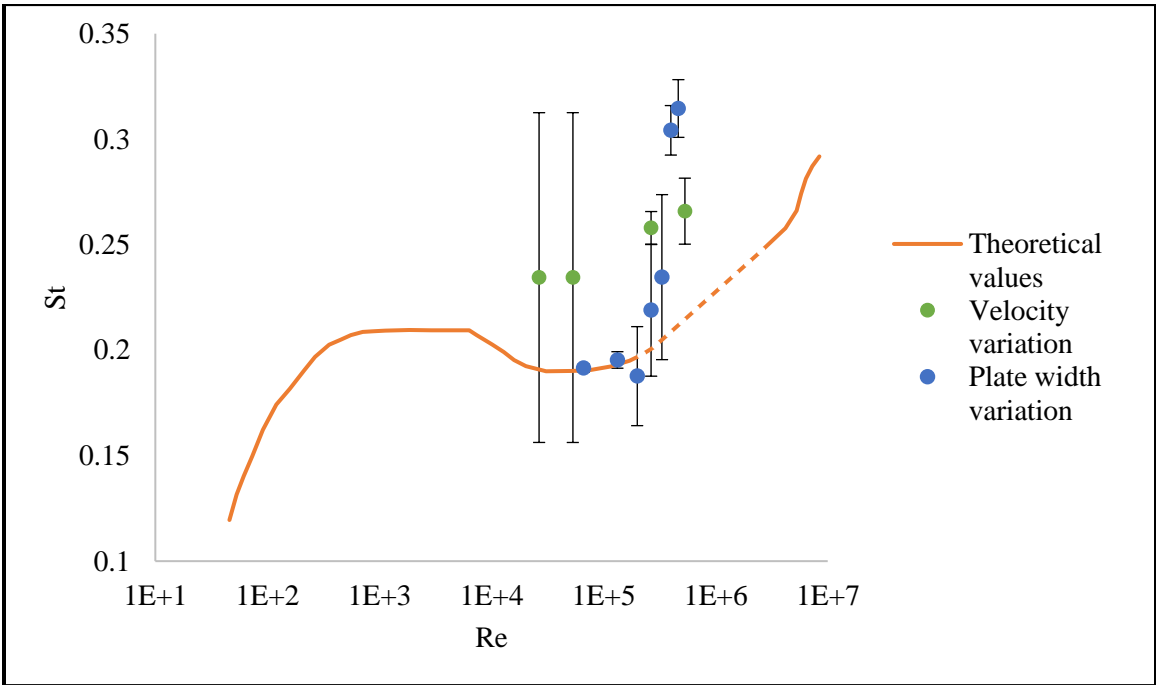
Different Reynolds numbers were tested by varying the plate width and uniform velocity of the fluid. For cases with lower Reynolds numbers, the number of time steps to trigger asymmetry was decreased and the threshold for eliminating large vortices was decreased. It was very difficult to obtain and analyze autospectra for very low and very high pseudo-Reynolds numbers so only a limited variety of cases were tested. By employing FFT for a clean segment of data to calculate the Strouhal number of each wake, the results can be compared to the theoretical Strouhal numbers interpolated from the  $Re$  vs.  $St$  curve developed from previous research.

**Table 9 Reynolds and Strouhal Number Comparison**

	Plate Width (m)	Uniform Velocity (m/s)	Pseudo-Reynolds Number	Strouhal Number	
				Theoretical	DVM Model
$U_{\infty}$ Variation	4	0.1	2.66E+04	0.191	0.2344
	4	0.2	5.31E+04	0.190	0.2344
	4	1	2.66E+05	0.200	0.2579
	4	2	5.31E+05	0.206	0.26588
$a$ Variation	1	1	6.64E+04	0.190	0.19145
	2	1	1.33E+05	0.193	0.19536
	3	1	1.99E+05	0.197	0.18768
	4	1	2.66E+05	0.200	0.21896
	5	1	3.32E+05	0.202	0.17582
	6	1	3.98E+05	0.204	0.3042
	7	1	4.65E+5	0.205	0.3145

A graphical representation of the results is shown below where the values obtained from autospectra are indicated by points and the theoretical values established by previous experimental studies are represented with the solid line. For some of the simulations, the nascent vortex offset distance was varied to improve the results and simplify the FFT analysis. Even then, it was very difficult to obtain an obvious peak from the autospectra generated by the program with only 512 seconds of run time. This parametric study would have benefitted from dedicating more time to running the program for more iterations and more time to running a FFT of the larger data set. Furthermore, it should be noted that most plots of  $St$  versus  $Re$  in most textbooks demonstrate uncertainty on the interval  $10^4 < Re < 10^6$  due to difficulty in obtaining repeatable Strouhal frequencies in wind tunnel experiments in this range. In this study, cases tested at Reynolds numbers below 10,000 resulted in autospectra that were extremely difficult to read and were not included in Figure 19. Subsequently, the cases that were successfully tested fell into the uncertain region of the plot. As the reference curve is derived from mere interpolation in this domain, the comparison between the computationally obtained results and theoretical results may be subject to error. Finally, the application of Inviscid Flow Theory mandates the assumption of zero viscosity. The following results were found by arbitrarily assigning the flow a viscosity that matched the properties of air at room temperature. Rather than maintaining a constant, arbitrarily chosen viscosity, perhaps varying the viscosity for each simulation may have yielded a better correlation between the results and the theoretical predictions.

Despite the large discrepancies with the velocity variation study and the higher Reynolds numbers in the plate width variation, the general trend of the data matches the theoretical curve. The simulation results demonstrate a correlation between the plate width, uniform velocity of the fluid, and the resulting frequency of the wake that simulates the correlation found in experimental testing carried out in prior studies on the subject. Since the rest of the results will be generated near the area of the curve in which the theoretical and computed values align fairly well, the results of the DVM portion of the study are acceptable and can be further developed to simulate flutter in the third and final portion of this thesis.



**Figure 19 Strouhal numbers compared with pseudo Reynolds number**

### 6.3 Flutter Results

After validating the functionality of the DVM program, the final stage of the project is to include the dynamic motion equation into the time loop so that the deflection angle updates in every time step according to the moment induced by the aerodynamic forces acting on the plate. The display window is also updated to display the plate's new position in the  $z_6$ -plane for the viewer to visualize the movement of the plate. A sample case is shown in the image sequence below.

This case will provide the baseline scenario against which all other test cases can be compared. In this simulation, air flows past a 4 m wide flat plate at 1 m/s. The torsional rigidity of the plate as it rotates about its center is 50 Nm, the torsional damping is 1 Nms, and the mass moment of inertia is 100 kgm<sup>2</sup>. The stop sign is set to fail at a torsional stress of 20 GPa. These properties are not based on any real materials and fall somewhere between plastic and wood; they were simply chosen to obtain results that clearly demonstrate the capacity of the program.

The case shown below oscillates back and forth until it reaches a certain angle such that the specimen fails due to a hypothetical torsional stress limit. This occurs at 29.625 seconds. Other cases can be run in the simulation by varying the different mechanical properties of the plate and their failure times can be compared to this baseline case. While the program is running, the drag force, wake pressure, and deflection angle of the plate are recorded in every time step. From the pressure data set, the Strouhal number of the wake can be found using FFT in the same way as it was found previously. In a similar manner, the deflection angle data array can be utilized to calculate the damped frequency of the stop sign. This value can then be compared with a theoretical value based on the stop sign's mechanical properties. This value can also be analyzed for each test case to provide support for the proper implementation of the dynamic motion equation of the plate.

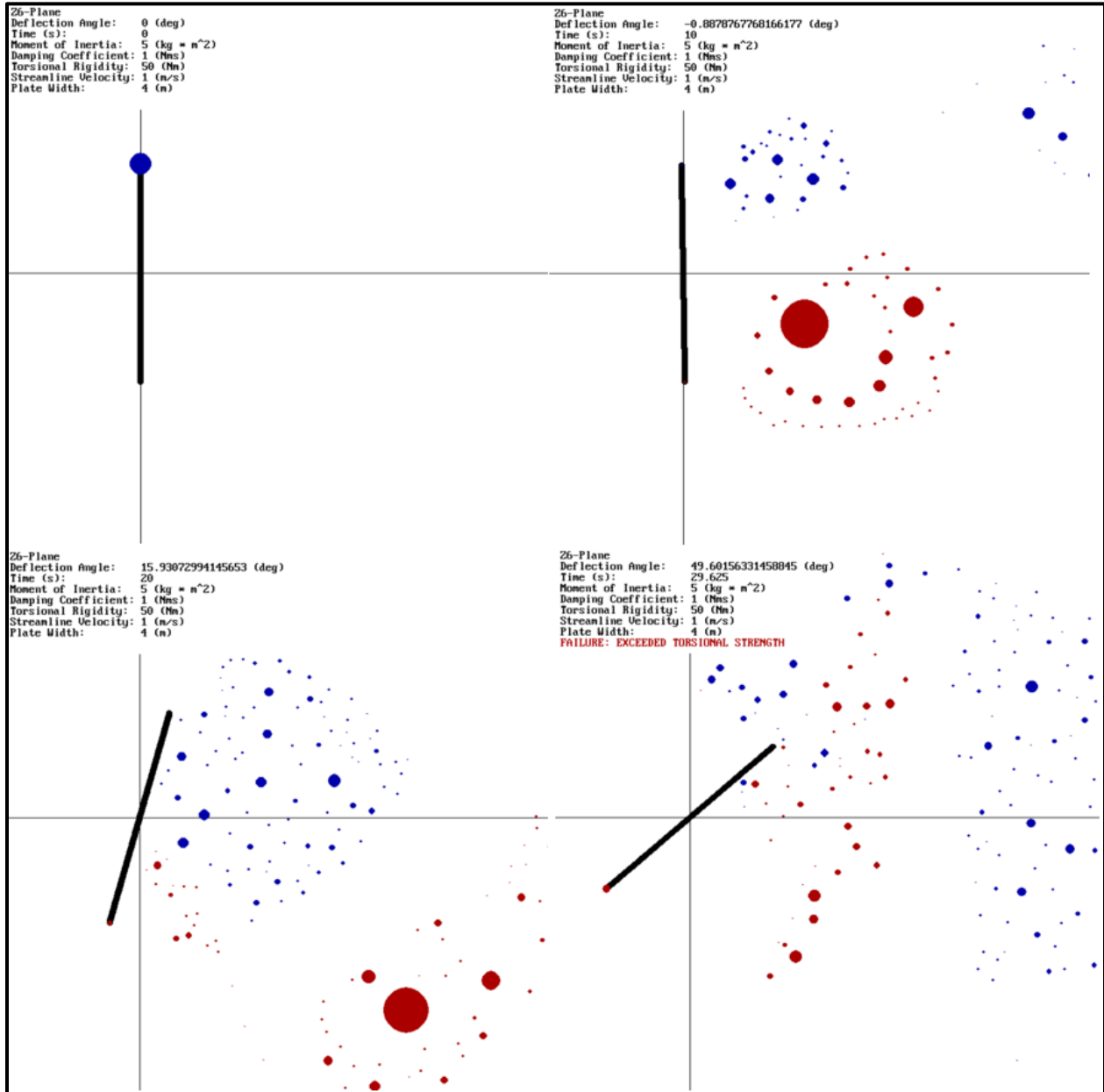
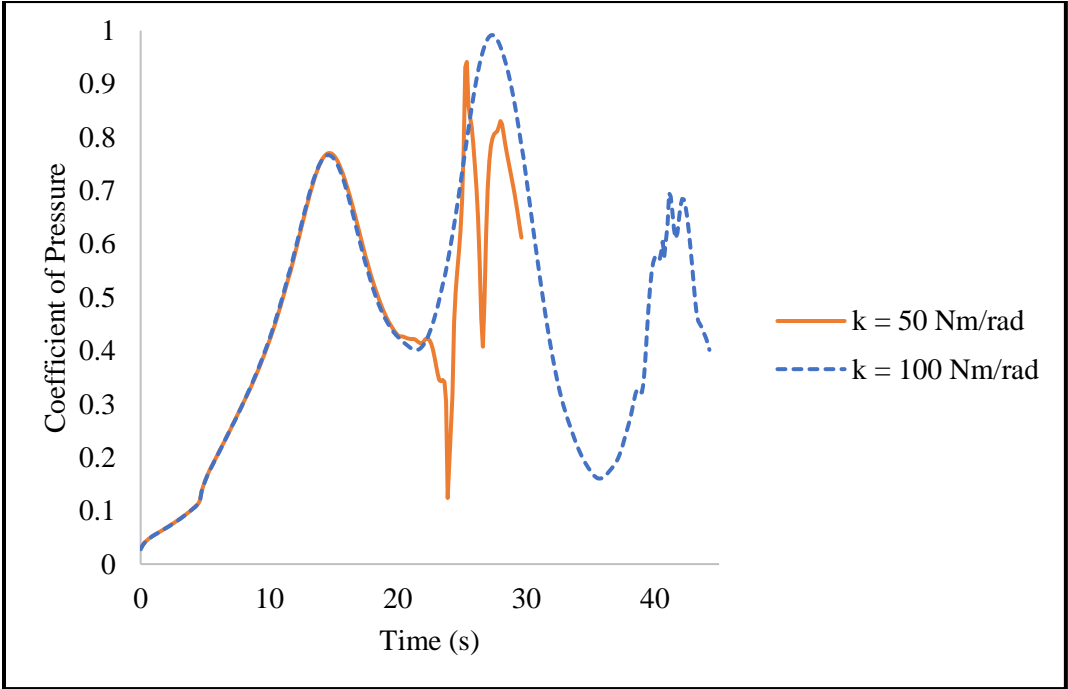


Figure 20 Flutter simulation

Two cases of different torsional stiffness values are compared below while all other properties of the plate are held constant. Compared with the baseline test case, a specimen with a torsional stiffness of 100 Nm (twice the original value of 50 Nm) fails at 44.25 seconds, surviving almost twice as long as the original specimen did in the same flow conditions. The same pressure probe is used to record the  $C_p$  in the wake behind the two plates. At the beginning of the simulation, the two cases yield identical results. However, when the plate with lower stiffness begins to undergo significant deflections of more than ten degrees, it interacts with the flow and creates anomalous behavior of the fluid in the wake. Shortly after that, the specimen fails which signals for the end of data collection. The case with the higher stiffness eventually sees the same effect of the plate motion on the flow resulting in rough, choppy behavior in the wake just before specimen failure.

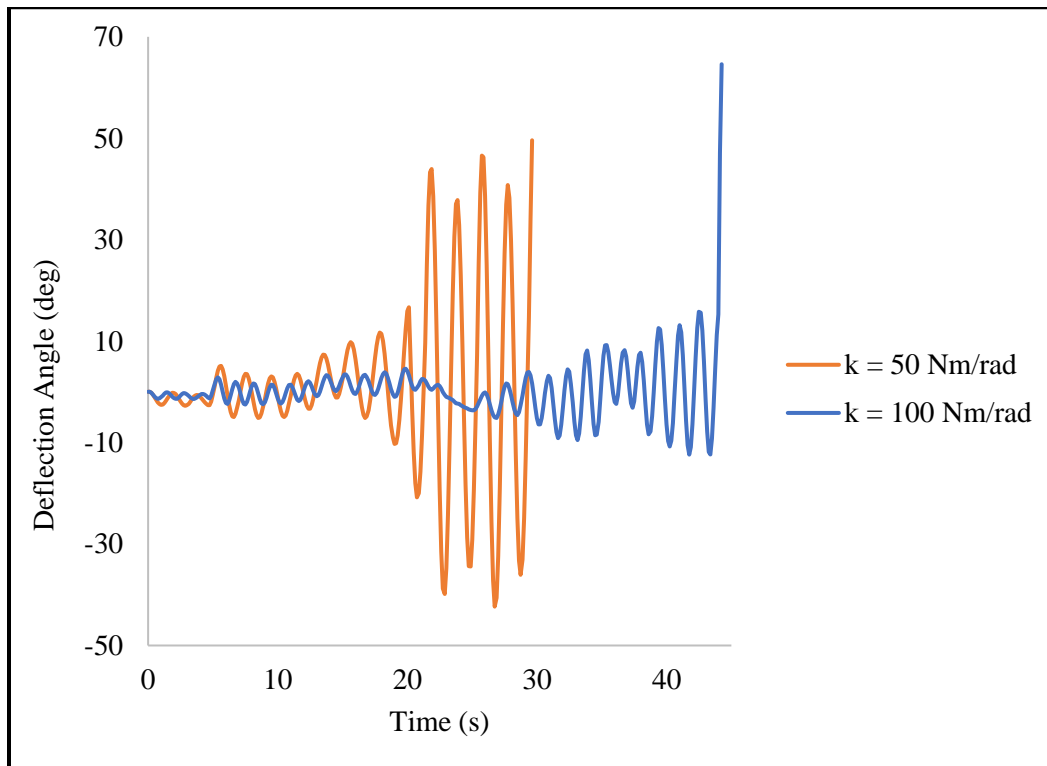


**Figure 21 Pressure comparison for varying torsional stiffness**

Both plots for angular deflection over time shown below resemble operation of a second order system near resonance; these results support proper implementation of the dynamic motion equation:

$$I_{zz}\ddot{\theta} + b\dot{\theta} + \kappa\theta = M_z(t) \quad (25)$$

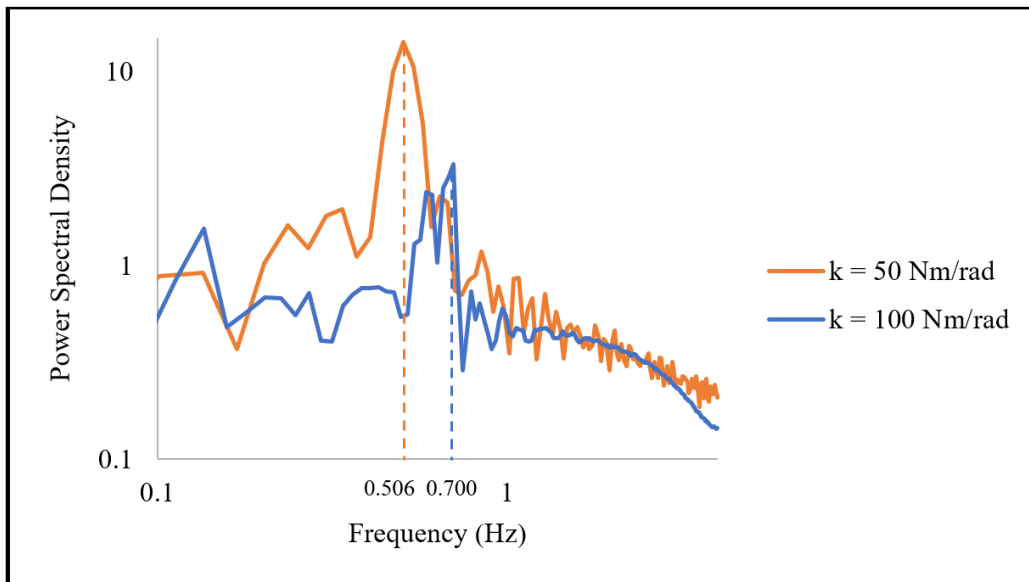
The case with the higher torsional rigidity deflected less under the same aerodynamic forces and took more time to reach failure. Additionally, the frequency at which the plate oscillates just before failure is an indication of the plate's damped frequency which is higher than the specimen with lower torsional rigidity. These results match the predictions for this simulation because the deflection angle curve of the specimen with the lower torsional rigidity demonstrates oscillations of greater amplitude, a shorter time to failure, and a lower damped frequency than the specimen with a higher torsional rigidity. These comparisons align with the logical predictions for a stiffer specimen in Equations 23 and 27 and support the proper execution of dynamic motion in the program to simulate flutter.



**Figure 22 Deflection angle comparison for varying torsional stiffness**



To find the damped frequency of each specimen, the two deflection angle curves were analyzed with a FFT shown below. The peaks of the power spectra occur at approximately 0.506 Hz for the 50 N\*m/rad case and 0.700 Hz for the 100 Nm/rad case (see the dashed lines in Figure 23); these values are within one frequency increment of the expected values on the autospectra indicating the accuracy of the dynamic motion equation. This accuracy is also represented by Figures 30 and 32 in which the calculated values for damped frequency of several specimens align closely with their theoretical counterparts.



**Figure 23 Autospectra of deflection angle for varying torsional stiffness**

Several other combinations of plate properties have been tested and their results examined. Their predicted values for the plate's resonant frequency and the wake's Strouhal frequency are given in the following tables. The actual results found after running the simulation are also provided as the DVM Model values to be compared with the theoretical values. The time at which each specimen failed and the datasets were examined for trends relating each property to the mechanical properties of the plate. For all these cases, the uniform velocity was maintained at 1 m/s and the plate width at 4 m. The baseline values for torsional stiffness, damping and mass moment of inertia were 50 Nm, 1 Nms, and 5 kgm<sup>2</sup> and each parameter was varied while keeping all else constant. For data sets which failed rather quickly, there were

not many points in the data set; as a result, the frequency interval on the FFT plots to find Strouhal frequency was so large that no peak could be discerned and the Strouhal number was deemed inconclusive.

**Table 10 Summary of Results**

	Mass Moment of Inertia (kg*m <sup>2</sup> )	Damping Coefficient (N*m*s)	Torsional Spring Constant (N*m)	Time to Failure (s)	Damped Resonant Frequency (Hz)		Strouhal Number	
					Theo.	DVM	Theo.	DVM
κ Variation	5	1	10	13.875	0.2245	0.2540	0.2004	inconclusive
	5	1	12	17.500	0.2460	0.2520	0.2004	inconclusive
	5	1	15	12.750	0.2752	0.2540	0.2004	inconclusive
	5	1	25	21.375	0.3555	0.3150	0.2004	inconclusive
	5	1	37	20.500	0.4327	0.4409	0.2004	inconclusive
	5	1	50	29.625	0.5030	0.5039	0.2004	inconclusive
	5	1	75	44.375	0.6162	0.5961	0.2004	0.2510
	5	1	100	59.625	0.7116	0.7216	0.2004	0.7559
	5	1	125	70.625	0.7956	0.7671	0.2004	0.3131
	5	1	150	46.125	0.8716	0.8471	0.2004	0.2510
	5	1	175	71.000	0.9414	0.9237	0.2004	0.3131
	5	1	200	53.375	1.0065	0.9412	0.2004	0.5039
b Variation	5	0.5	50	15.750	0.5032	0.5079	0.2004	inconclusive
	5	1	50	29.625	0.5030	0.5039	0.2004	inconclusive
	5	2	50	41.875	0.5023	0.4706	0.2004	0.2510
	5	5	50	44.250	0.4970	0.4409	0.2004	0.2510
	5	7	50	43.625	0.4908	0.3150	0.2004	0.2510
	5	10	50	102.125	0.4775	0.3765	0.2004	0.3131
	5	12	50	172.250	0.4656	0.3206	0.2004	0.2502
	5	15	50	134.250	0.4431	0.4457	0.2004	0.2815
	5	20	50	140.250	0.3898	0.2893	0.2004	0.2815
	5	25	50	284.000	0.3082	0.1133	0.2004	inconclusive
	5	30	50	141.125	0.1592	0.0313	0.2004	0.2815
	5	31.622777	50	157.250	overdamped	0.0391	0.2004	0.2502
	5	33	50	136.375	overdamped	0.0313	0.2004	0.2815
	5	35	50	69.250	overdamped	0.0470	0.2004	0.3131
	5	40	50	352.250	overdamped	0.0743	0.2004	0.1094

**Table 11 Summary of Results (continued)**

	Mass Moment of Inertia (kg*m <sup>2</sup> )	Damping Coefficient (N*m*s)	Torsional Spring Constant (N*m)	Time to Failure (s)	Damped Resonant Frequency (Hz)		Strouhal Number	
					Theo.	DVM	Theo.	DVM
I <sub>zz</sub> Variation	1	1	50	25.875	1.1226	1.1969	0.2004	inconclusive
	2	1	50	41.875	0.7948	0.7529	0.2004	0.2510
	3	1	50	40.750	0.6492	0.5961	0.2004	0.2510
	4	1	50	24.375	0.5623	0.5039	0.2004	inconclusive
	5	1	50	29.625	0.5030	0.5039	0.2004	inconclusive
	6	1	50	19.500	0.4592	0.4409	0.2004	inconclusive
	7	1	50	40.875	0.4252	0.4392	0.2004	0.3765
	8	1	50	31.875	0.3978	0.4392	0.2004	0.2510
	9	1	50	40.625	0.3750	0.3765	0.2004	0.2510
	10	1	50	21.125	0.3558	0.3150	0.2004	inconclusive

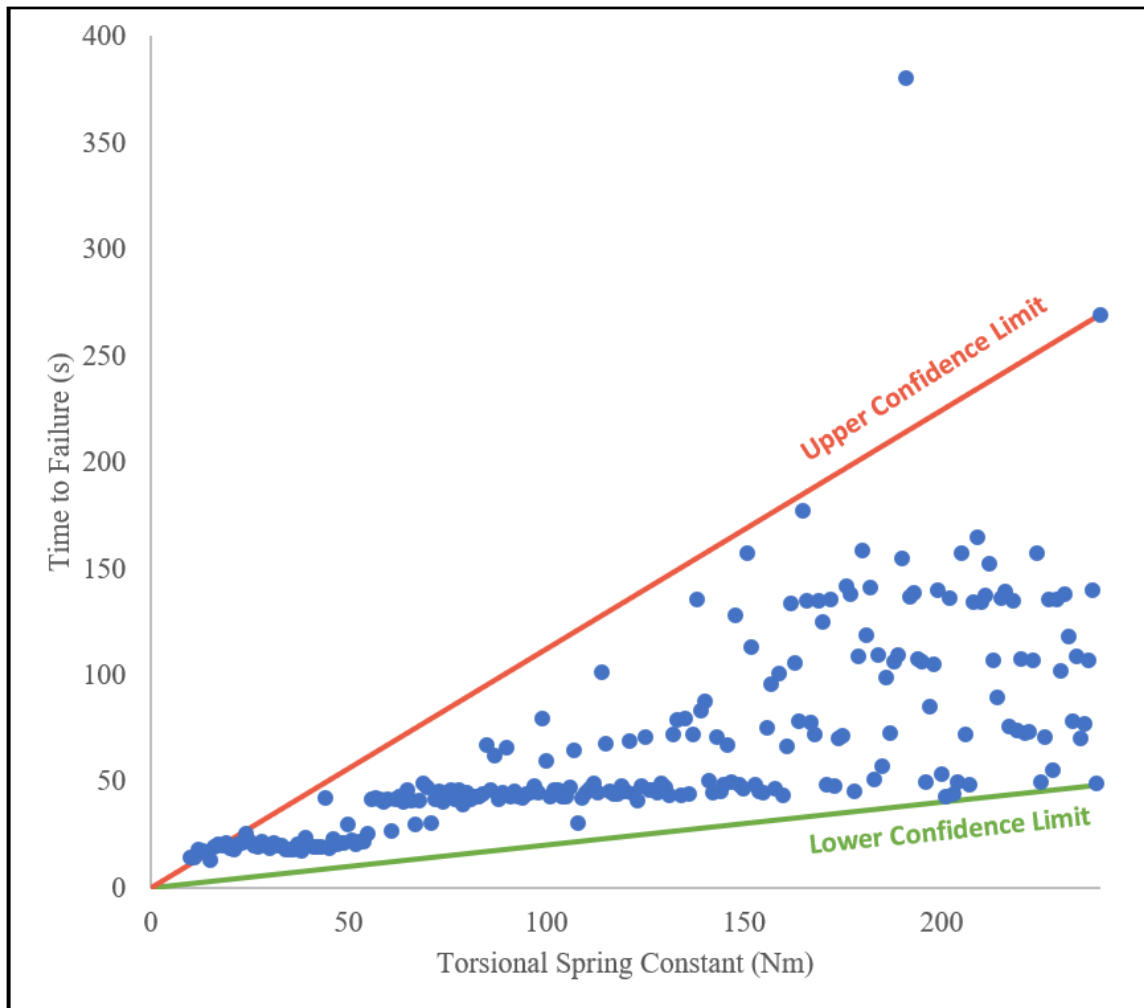
To observe the correlations between each mechanical property and the specimen's failure time, several more test cases were completed to obtain more conclusive graphs with more data points; the resulting graphs are shown in Figures 24, 25, and 26. From the following plots, it can be concluded that the torsional stiffness and damping coefficient of the specimen have a positive correlation with its resulting failure time in the program. However, the mass moment of inertia seems to demonstrate no clear correlation with the failure time. The performance of the flat plate can be expressed in terms of a confidence level. One can be most confident in the integrity of the flat plate below the lower confidence limit and least confident above the upper confidence limit. For changes in failure times due to the torsional spring constant, the minimum time to failure is given by Equation 31. If operated below the lower confidence limit (green line in Figure 24), it is highly unlikely that the flat plate will fail.

$$\Delta t_{f,LCL} \cong \left(0.20 \frac{s}{Nm}\right) * \kappa \quad (31)$$

On the other hand, failure is almost guaranteed to occur for time intervals greater than the failure time predicted by Equation 32. These results indicate that the flat plate must be operated below the red line

in Figure 24 to avoid failure; ideally, the flat plate should be operated below the green line for optimal safety measures.

$$\Delta t_{f,UCL} \cong \left(1.12 \frac{s}{Nm}\right) * \kappa \quad (32)$$



**Figure 24 Failure time for varying torsional stiffness**

Similarly, by varying the damping coefficient of the flat plate, the minimum time to failure clearly has upper and lower confidence limits shown by the red and green lines in Figure 25, respectively. If operated below the lower confidence limit, the flat plate will not fail. This is the safest time limit during which the flat plate should be subject to flutter.

$$\Delta t_{f,LCL} \cong 1.4 (Nm)^{-1} * b \quad (33)$$

The upper confidence limit represents the threshold after which the flat plate is extremely likely to fail for any given damping coefficient (at the given torsional spring constant and mass moment of inertia). This curve can be approximated by the linear equation below. It is possible that the upper confidence limit is nonlinear, but more data points should be gathered to draw a conclusion.

$$\Delta t_{f,UCL} \cong 14.0 (Nm)^{-1} * b \quad (34)$$

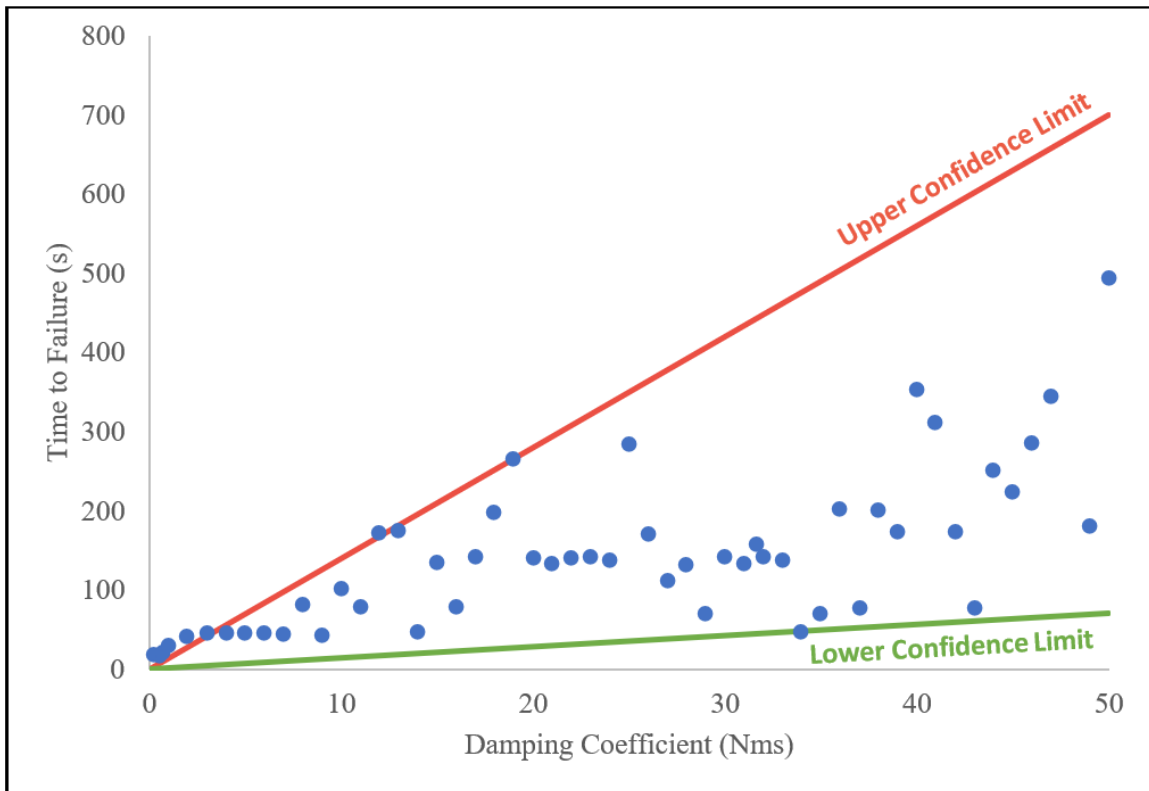


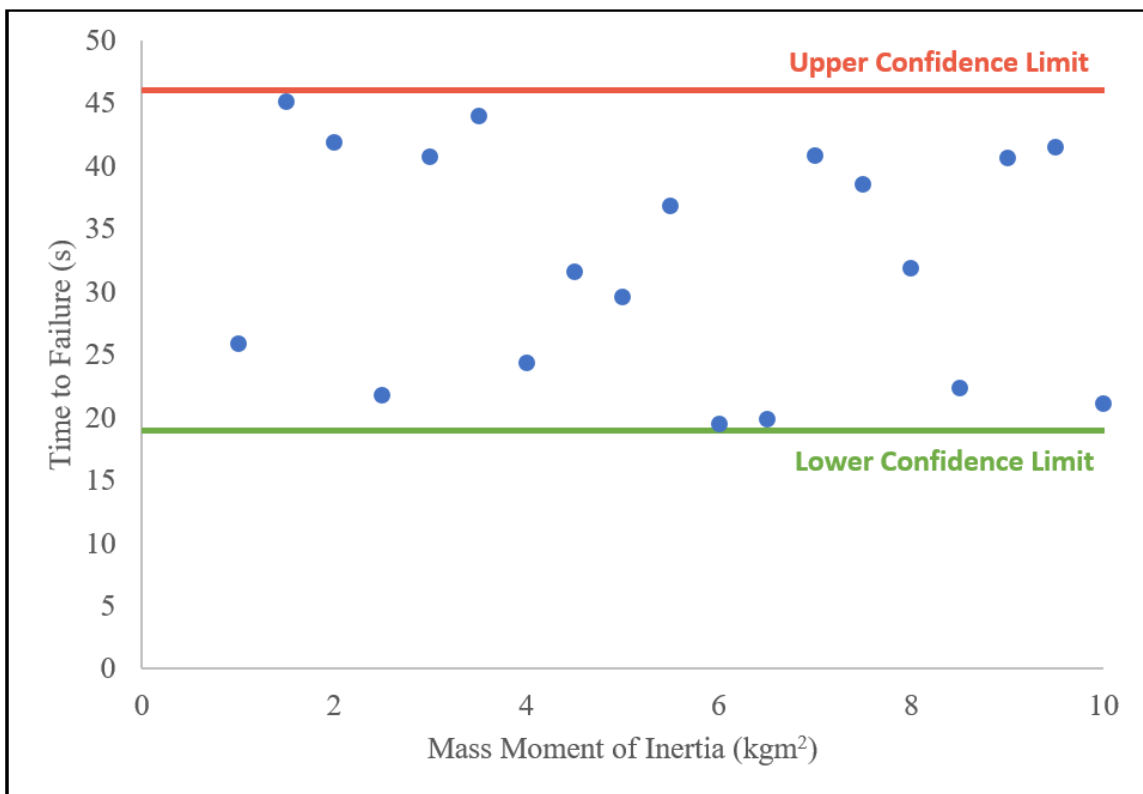
Figure 25 Failure times for varying damping coefficient

Finally, by varying the mass moment of inertia of the flat plate, the minimum time to failure has horizontal upper and lower confidence limits since there is no clear trend in the data points. The lower and upper confidence limits are described by Equations 35 and 36, respectively. With a torsional spring constant of 50 Nm and a damping coefficient of 1 Nms, the flat plate can be safely operated up to 19 seconds regardless of the mass moment of inertia but will never operate beyond 46 seconds.

$$\Delta t_{f,LCL} \cong 19 \text{ s} \quad (35)$$

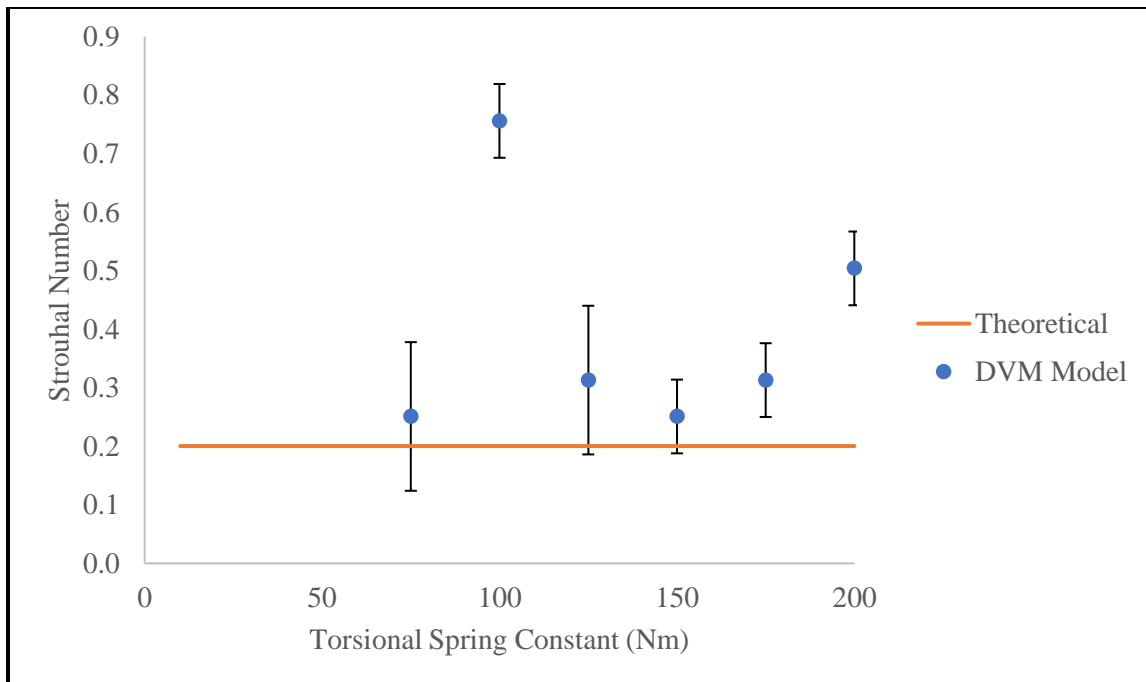
$$\Delta t_{f,UCL} \cong 46 \text{ s} \quad (36)$$

The confidence lines developed from these results indicate the possibility of utilizing a DVM program to provide useful information about an aircraft's operating envelope, for example. This application of results supports the significance and practicality of this simulation technique.

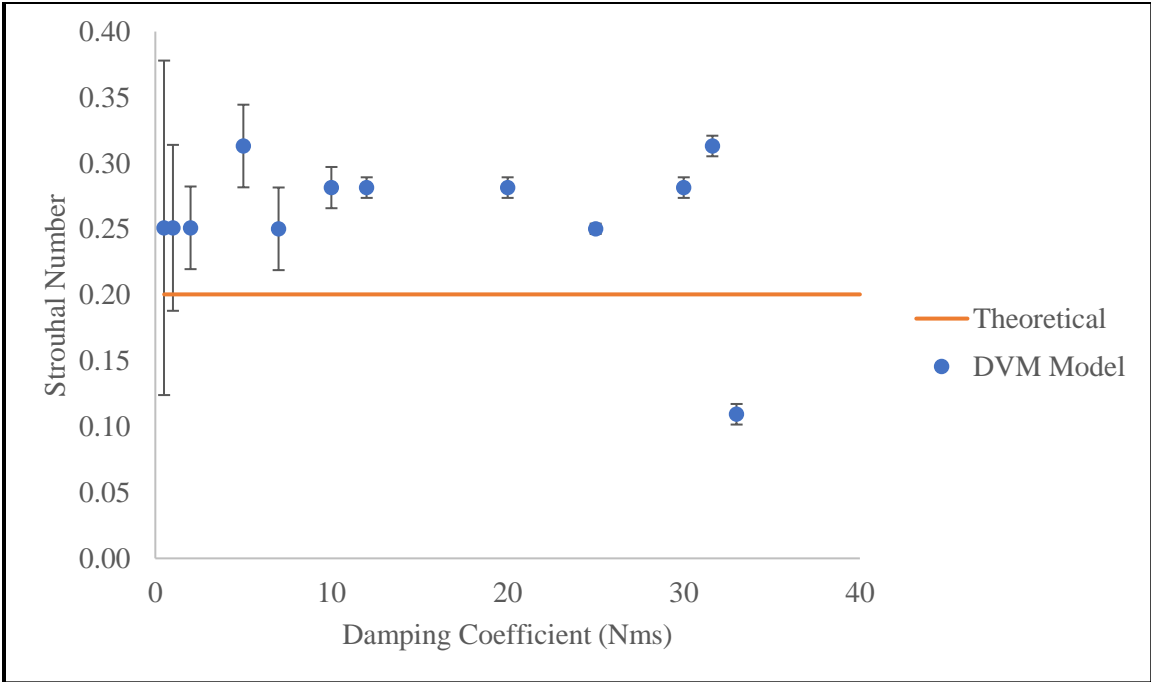


**Figure 26 Failure times for varying mass moment of inertia**

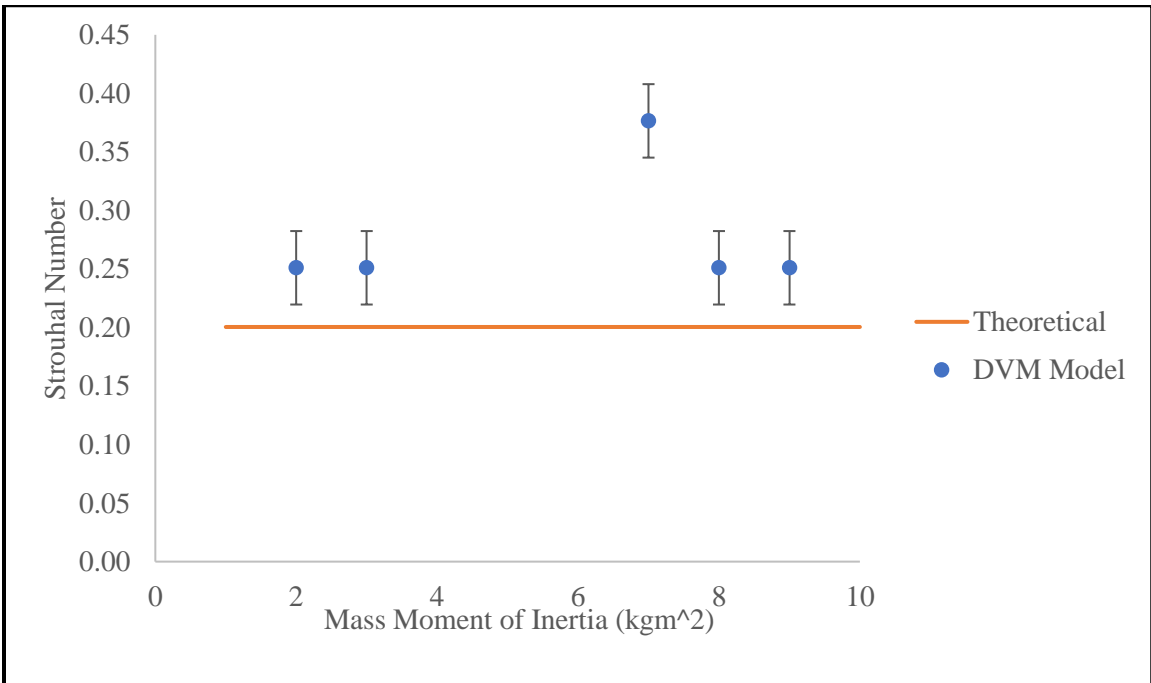
For the test cases summarized in Tables 10 and 11, the wake pressure and deflection angle data were analyzed for the Strouhal number of the wake and damped frequency, respectively. Before the dynamic motion equation was added into the program, the Strouhal number of the wake was found to be between 0.21 and 0.25 (depending on the time steps used to trigger asymmetry and other minor modifications to the code). Now that the plate is free to rotate but the velocity and plate width stay the same, the Strouhal frequency of the wake is expected to remain constant. However, the motion of the plate interacts with the wake and disrupts the flow. This is evident from direct observation of the  $z_6$ -plane while the program is running. The movement of the plate shows a slight effect on the Strouhal number due to the variation of any parameter. However, the graphs reveal no significant trend to the effect of each parameter on the Strouhal number of the wake. As with the data taken previously, it was very difficult to ascertain the true peak in the autospectra and may have been clearer had more resources and time been dedicated to investigating the pressure in the wake behind a flat plate in flutter.



**Figure 27 Strouhal numbers for varying torsional stiffness**



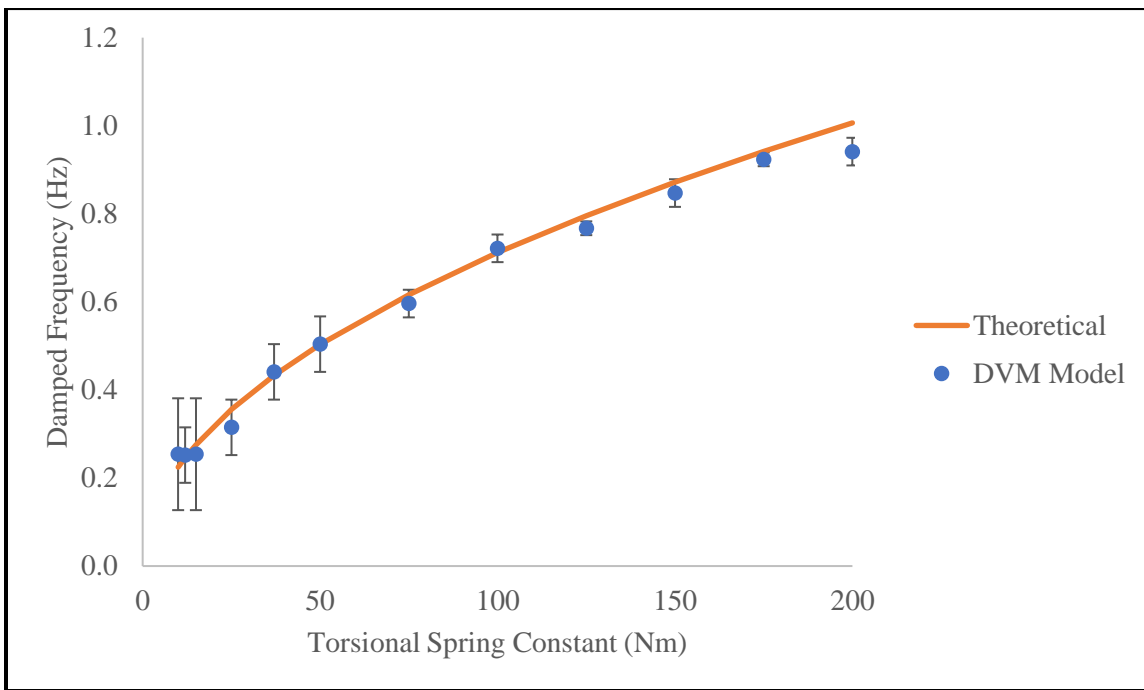
**Figure 28 Strouhal numbers for varying damping coefficient**



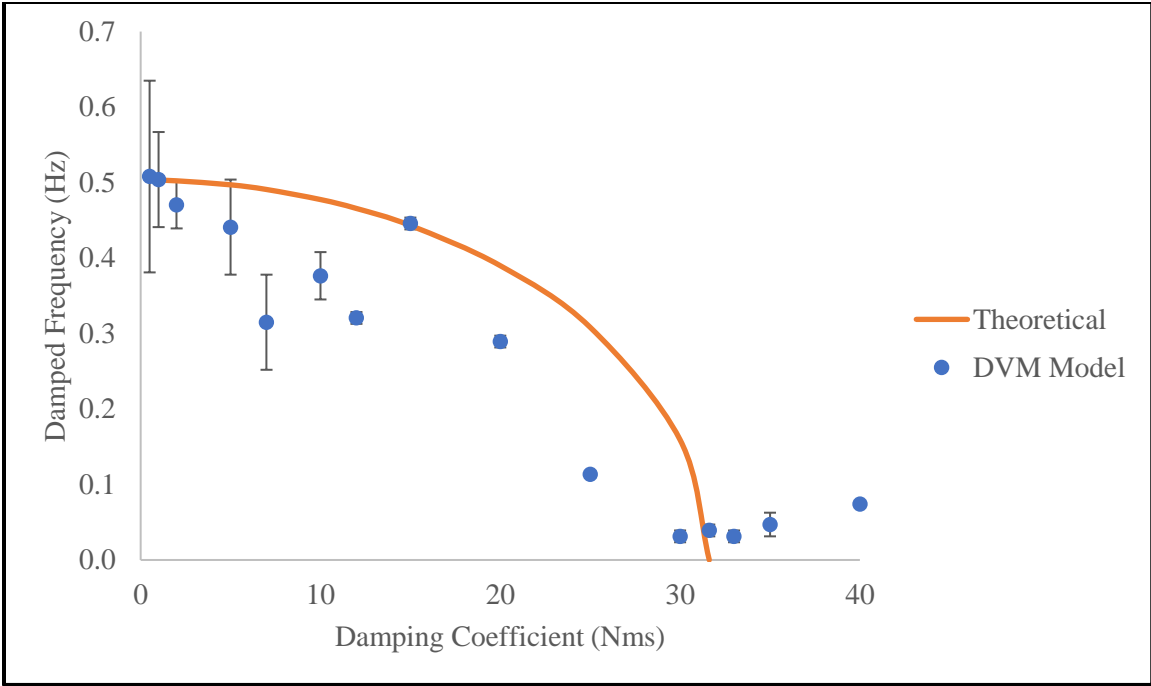
**Figure 29 Strouhal numbers for varying mass moment of inertia**



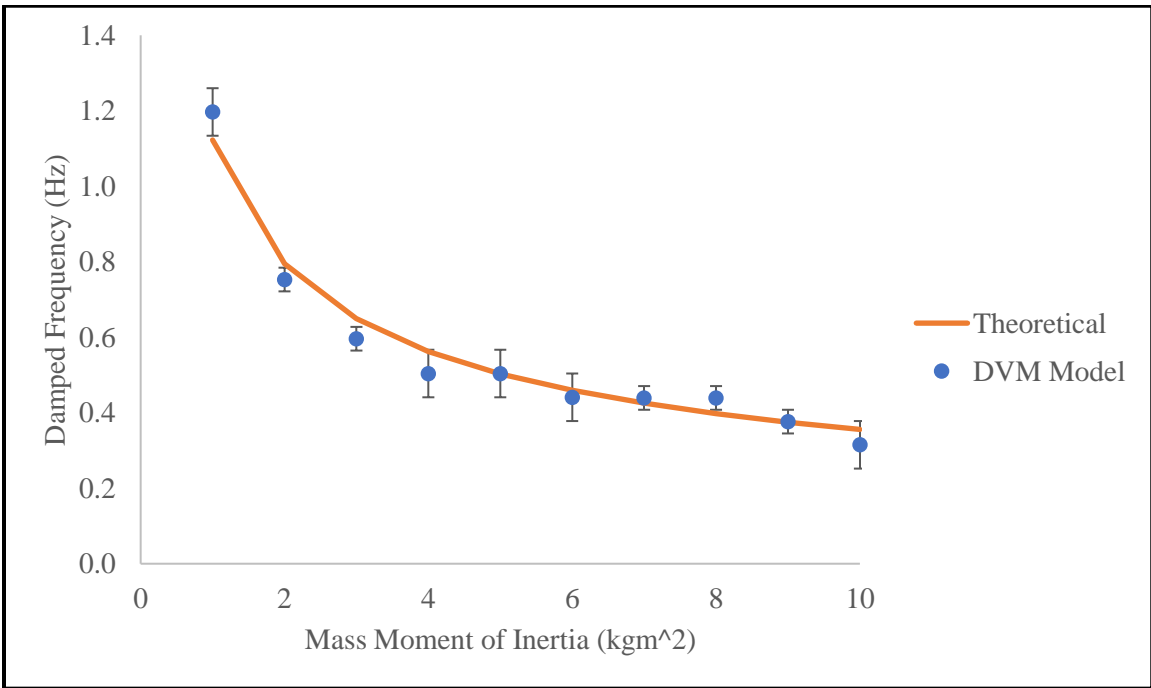
The damped frequencies of each of the test cases are plotted below. For the torsional stiffness and mass moment of inertia parametric studies, the results align with the theoretical predictions based on Equation 29. This correlation supports the proper use of the dynamic motion equation in a flutter simulation and provides predictable results. However, the damping coefficient variation plot in Figure 31 shows a slight deviation of the values obtained in the program from the expected trajectory. It is unlikely that the peaks in the autospectra found for these cases, particularly those with higher damping coefficients, are true indicators of the damped frequency of the flat plate. For the cases with damping coefficients beyond the critical damping coefficient, there is no real value predicted using Equation 29; the autospectral peaks might reflect some frequency associated with the interaction of the discrete vortices creating noise unrelated to the damping frequency.



**Figure 30 Damped frequency for varying torsional stiffness**



**Figure 31 Damped frequency for varying damping coefficient**



**Figure 32 Damped frequency for varying mass moment of inertia**

Introducing motion of the plate into the program results in a realistic visualization of the movement of a flat plate as it rotates about its center due to alternating aerodynamic forces on the plate induced by the wake generated behind the plate. The motion of the plate serves as a plausible prediction of flutter. Various combinations of mechanical properties were tested in the program and the resulting time to failure, Strouhal number, and damped frequency were analyzed. A positive correlation between damping coefficient and torsional stiffness with time to failure met the predictions previously stated. However, a lack of correlation between mass moment of inertia and time to failure provided insight on the arbitrary nature of this property on flutter-induced effects. The Strouhal number of the wake seemed to be rather consistent with the results generated with a fixed plate; however, the large frequency interval and the subjectivity in identifying the correct peak in the FFT indicates that the results for Strouhal number are inconsistent and inaccurate. Finally, the damped frequency found for each specimen correlated well with the results obtained from the equation for damped frequency, Equation 29, particularly when varying the torsional stiffness and mass moment of inertia.

The conclusions drawn from the trends observed in the preceding figures indicate that the motion of the plate is realistic and corresponds with the physical expectations based on theoretical equations. As with the previous results supporting the use of conformal mapping and DVM, the use of dynamic motion in this program is a valid technique for flutter simulation. The combination of these techniques provides a potentially revolutionary method of flutter simulation illustrated by the program developed and studied in this work.

## CHAPTER 7

### CONCLUSION

Flutter-induced effects can be ultimately catastrophic in many engineering applications such as bridges, buildings, and aircraft. Understanding and predicting the behavior of a structure under aerodynamic loads is essential to preventing structural failure. The objective of this study was to develop a viable model of aerodynamic flutter using Discrete Vortex Method (DVM) to support a proposition for a new, more efficient technique to replace current flutter models used in the aerospace industry. Using a FreeBASIC compiler, a program was written that utilized the principles of inviscid flow, DVM, and dynamic motion to implement a time iterative display representing flow past a flat plate. Complex positions are translated between different flow fields using a series of six conformal mappings of  $z$ -planes. Viscous effects are introduced into the model using DVM. By numerically separating the front and back surfaces of the plate into short line segments, the force distribution across each surface and net moment acting on the plate by the fluid around it are calculated. Motion of the plate is created by a dynamic motion equation in which the mass moment of inertia, damping coefficient, and torsional stiffness of the plate rotating about its center dictate the change in displacement of the plate. Flow visualization is accomplished by plotting the locations of each of the discrete vortices in the flow field generating a wake behind the plate. The plate oscillates about its center due to a fluctuating moment acting on the plate induced by the flow field and calculated using inviscid flow theory. By updating the angular displacement of the plate in each time step of the program, the movement of the specimen can be visually observed and its failure can be predicted at a certain torsional stress.

The program was built and tested for indicators of accuracy and realism along the way. The inviscid flow equations used were confirmed by plotting points along the inviscid streamlines in each of the conformal mappings used to obtain the physical flow field. Summation of the total forces acting on plates at different angles to the streamline flow resulted in zero net forces as predicted by the assumptions of

inviscid flow. DVM was checked by first observing the wake and comparing it to the predicted shape of a von Kármán vortex street and then by measuring the Strouhal number of the wake and comparing it with experimental results based on the well-established correlation between Strouhal and Reynolds numbers. Finally, many flutter simulations were run for cases of varying combinations of mechanical properties resulting in damped oscillations of the plate rotating about its center. The flutter predictions were supported by finding the damped frequencies of the different specimens and matching the values with predicted frequencies based on the plate's mechanical properties. The computational results for damped frequency matched up nearly perfectly with the theoretical values when torsional stiffness and mass moment of inertia were varied; however, variation of the damping coefficient resulted more aberrant data. Furthermore, the failure times of varying specimens provided insight into trends in the relationship between each property and when a specimen might fail; this procedure may be extremely applicable in the aerospace industry in the study of aerodynamic flutter. For the simple case of a stop sign rotating about its post, the results demonstrated that the life span of a flat plate (stop sign) increased with torsional stiffness and damping coefficient but showed no distinct correlation with the sign's mass moment of inertia. The Strouhal number seemed to vary slightly but the results for this property were not very conclusive in this study due to the difficulty in identifying the correct peaks in the Fast Fourier Transform (FFT) autospectra. Overall, the results indicated that the behavior of the specimen is realistic and that the program serves as a valid model for a flat plate in aerodynamic flutter. The parametric studies completed in Chapter 6 of this thesis provide a small sample of potential simulations that could be performed to aid and support the design of future aircraft, for example. Each simulation only took a matter of minutes to run on a standard laptop computer providing evidence of the speed and efficiency achieved with the DVM technique. Compared with existing flutter models which used FEA and is extremely slow, this new program shows potential for a very desirable method of flutter simulation.

Because this technique has not previously been used for the purpose of studying flutter, the model built is very rudimentary and provides merely a simple demonstration of the potential for employing DVM in a simulation of aerodynamic flutter. There are many other factors to be considered and added to the

program in future studies. The program should continue to be validated through other indicators of realistic results. The wake generated behind the plate using DVM can be compared with other theoretical models and other experimental results [12]. To provide some real-world data, a flat plate of known mechanical properties can be set up in a wind tunnel, fixed to rotate about its center, and observed as air flows past the plate. Results produced by the program can then be compared with the theoretical behavior of the plate predicted in the program to further validate the reliability of the simulation's results.

One major aspect of structural dynamics neglected in this study was the influence of fatigue in structures under repeated loads. Fatigue considerations can be implemented in the program through the application of the rainfall technique to the plot of deflection angle over time, a case of variable amplitude torsional stress on the specimen [18]. For cases in which the specimen does not reach its torsional stress limit for a long period of time, it is likely that it will fail due to fatigue. An algorithm much more complex than a simple calculation of torsional stress is required to realistically determine when failure will occur. This failure time can be predicted using the rainfall technique and is essential to studies on flutter-induced failure in the structural dynamics community.

Another application to the existing program that will intrigue aerospace engineers for one is the consideration of conformal mappings to profiles other than a flat plate. While flat plates are impractical surfaces on aircraft, streamlined struts and cambered airfoils are rather prominent. Both of these profiles can be obtained through a series of conformal transformations very similar to those discussed in Chapter 2 of this study with slight modifications in the equation used to transform the  $z_4$ -plane to the  $z_5$ -plane. Additionally, the separation of the boundary layer can no longer be assumed at the tips of the flat plate. Instead, they can be found on the top and bottom surface of the airfoil using Thwaites' method. The rest of the program can be executed in the same manner as with the flat plate. Flutter of an airfoil's profile may be of more interest than a flat plate profile to those studying aerodynamic flutter in aircraft. Moreover, various profiles can be plotted in the same flow field and both considered for flutter-induced behavior in the same stream of fluid. Since many aircraft have a horizontal stabilizer aft of the wing, an interesting study may

include the positioning of two airfoils in the same flow field, one in the other's wake to observe the resulting behavior of the two structures and potential flutter-induced failure.

The methods used in this study introduce a world of possibility and potential for the future of flutter simulations in aerospace engineering. By introducing a method of modeling flutter that is less computationally expensive and time consuming, flutter analysis can be employed earlier on in the design process of aircraft and other engineering designs. Consideration of flutter-induced effects at an early stage in a project's timeline reduces the likelihood of expensive design overhauls or catastrophic failures in the future. A disastrous and common phenomenon, aerodynamic flutter must and can be better understood through the integration of discrete vortex method into the field of study.

## APPENDIX A

### FREEBASIC PROGRAM

Below is a copy of the program used to obtain the results in Chapter 6 of this study. To complete operations with complex number, the program references a BI file which is provided at the end of this Appendix. This configuration of the program is set up to freeze when the flat plate fails. However, this can be modified for processing batches of test cases by changing the phrase “sleep” to “end” under “if abs(torsional\_stress) > torsional\_strength then”. This will automatically end the program when the plate fails.

#### A.1 Main Program

```
-----  
'  
' Program: STOPSIGN24.bas  
'  
' Purpose: Compute the flutter of an idealized "stopsign" exposed  
' to a fluid flow resulting in flutter using the Discrete  
' Vortex Method (DVM).  
'  
' Input: a) Text file: "input_stopsign24.txt"  
'  
' Output: a) Output file "output.dat"  
' Version: ecc, 10/16/19  
'  
-----  
  
#include "complex.bi"  
  
Declare Function real_color(mycolor as integer) as string  
Declare Function flipFlop(x as integer) as integer  
Declare Function C_distance(ZA as complex, ZB as complex) as double  
  
dim as integer NN = 10000  
dim as double PI = 3.14159265358  
  
dim as double uniform_velocity, angle_initial, angle_increment, stopsign_width  
dim as integer fluid_choice, output_choice  
dim as double p_ambient, T_ambient  
dim as double torsional_strength, torsional_stress, torsional_stiffness, torsional_damping,  
stopsign_MOI,  
dim as integer N_streamlines  
dim as double N_radii_height, N_radii_width, N_radii_width_mult  
dim as integer color_streamlines, color_cylinder_wall, color_background  
dim as integer color_RT, color_LT, color_RRP, color_LRP  
dim as double t_max, t_increment  
dim as integer N_points, j, k, m, n, v  
dim as double a, phi_min, phi_max, psi_min, psi_max  
dim as double phi, psi(100)
```



```

dim as double x_min, x_max, y_min, y_max
dim as double streamfunction
dim as double scale_of_points
dim as double deflection_angle, deflection_angle_old, deflection_angle_older, deflection_degrees
dim as double angle_RT, angle_LT, surface_angle

dim as string input_line
dim as string plot_label(10,5)

dim as complex F(10), R(10), T(10), B(10), LT(10), RT(10), LRP(10), RRP(10)
dim as complex FRONT(10, NN), BACK(10, NN)
dim as complex argument
dim as complex streamline(10, 100, NN), w, angle

dim as integer plane_choice, plot_choice, axis_choice
dim as string key_choice
dim as integer F_choice, R_choice, T_choice, B_choice
dim as integer LT_choice, RT_choice, LRP_choice, RRP_choice
dim as integer FRONT_choice, BACK_choice, S_choice, V_choice, V_onoff_choice
dim as integer Q_choice

dim as double FRONT_delta_s(NN), BACK_delta_s(NN), FRONT_dFx_dr(NN), FRONT_dFy_dr(NN)
dim as double BACK_dFx_dr(NN), BACK_dFy_dr(NN), FRONT_dF_dr(NN), BACK_dF_dr(NN)
dim as double Z1x, Z1y, Z4x, Z4y, denominator
dim as complex value, value1, value2, value3, value4, value5

dim as complex FRONT_V(NN), BACK_V(NN), FORCE
dim as double FRONT_Cp(NN), BACK_Cp(NN), TOTAL_Cp(NN), MOMENT
dim as double radius, dF, p, density, V2, dM, nu, Re
dim as double x, y, dFx, dFy
dim as double max_abcissa, max_ordinate, min_ordinate, old_abcissa, old_ordinate
dim as double tiempo

dim as complex vortex_position(10, NN), dw_dz(10, NN), vortex_velocity(10, NN), vortex_strength(NN)
dim as integer N_vortices
dim as complex z, C_sum, one, a2, negone, C_uniform_velocity, velocity
dim as double U_s, circulation, total_circulation, trigger, dissipation, distant, amalg_dist, offset

dim as integer zero_vortices_index(NN), number_of_zero_vortices
dim as complex probe_position(10), probe_velocity(10)
dim as double probe_Cp(NN)

```

```

'-----
'
' A. Obtain User Input.
'-----

'-----
'
' A.1 Read in user data from 'input-stopsign.txt.'
'-----

open "input-stopsign24.txt" for input as #1

do while(eof(1) = 0)

line input #1, input_line

if mid(input_line,8,3) = "A.1" then uniform_velocity = val(mid(input_line,70,10))
if mid(input_line,8,3) = "A.2" then angle_initial =val(mid(input_line,70,10))*PI/180

if mid(input_line,8,3) = "B.1" then fluid_choice = int(val(mid(input_line,70,10)))
if mid(input_line,8,3) = "B.2" then p_ambient = val(mid(input_line,70,10))
if mid(input_line,8,3) = "B.3" then T_ambient = val(mid(input_line,70,10))

if mid(input_line,8,3) = "C.1" then torsional_strength = val(mid(input_line,70,10))
if mid(input_line,8,3) = "C.2" then stopsign_MOI = val(mid(input_line,70,10))
if mid(input_line,8,3) = "C.3" then torsional_damping = val(mid(input_line,70,10))
if mid(input_line,8,3) = "C.4" then torsional_stiffness = val(mid(input_line,70,10))
if mid(input_line,8,3) = "C.5" then stopsign_width = val(mid(input_line,70,10))

if mid(input_line,8,3) = "D.1" then N_streamlines = int(val(mid(input_line,70,10)))
if mid(input_line,8,3) = "D.2" then N_radial_height = val(mid(input_line,70,10))
if mid(input_line,8,3) = "D.3" then N_radial_width = val(mid(input_line,70,10))

```

```

if mid(input_line,8,3) = "D.4" then N_radii_width_mult = val(mid(input_line,70,10))
if mid(input_line,8,3) = "D.5" then color_streamlines = int(val(mid(input_line,70,10)))
if mid(input_line,8,3) = "D.6" then color_cylinder_wall = int(val(mid(input_line,70,10)))
if mid(input_line,8,3) = "D.7" then color_background = int(val(mid(input_line,70,10)))
if mid(input_line,8,3) = "D.8" then N_points = int(val(mid(input_line,70,10))

if mid(input_line,8,3) = "E.1" then t_max = val(mid(input_line,70,10))
if mid(input_line,8,3) = "E.2" then t_increment = val(mid(input_line,70,10))
if mid(input_line,8,3) = "E.3" then output_choice = int(val(mid(input_line,70,10)))
if mid(input_line,8,3) = "E.4" then plane_choice = int(val(mid(input_line,70,10)))
if mid(input_line,8,3) = "E.5" then V_onoff_choice = int(val(mid(input_line,70,10))

if mid(input_line,8,3) = "F.1" then trigger = val(mid(input_line,70,10))
if mid(input_line,8,3) = "F.2" then dissipation = val(mid(input_line,70,10))
if mid(input_line,8,3) = "F.3" then distant = val(mid(input_line,70,10))
if mid(input_line,8,3) = "F.4" then amalg_dist = val(mid(input_line,70,10))
if mid(input_line,8,3) = "F.5" then offset = val(mid(input_line,70,10))

loop

close #1

-----
'
' A.2 Compute preliminary constants.
'
' a = cylinder radius (m).
' x_min, x_max = range of the plot (m) in the x-direction.
' y_min, y_max = range of the plot (m) in the y-direction.
' Re = Reynolds number (-).
' trigger = distance between nascent vortices and tips (m).
'
-----

a = stopsign_width / 4 ' Cylinder radius (m).

x_min = -N_radii_width * a *.5
x_max = N_radii_width * a *(N_radii_width_mult-.5)
y_min = -N_radii_height * a *(N_radii_width_mult/2)
y_max = N_radii_height * a *(N_radii_width_mult/2)

if (fluid_choice = 0) then
nu = 15.06e-6 'kinematic viscosity of air at 20C (m^2/s)
end if

Re = uniform_velocity * 4 * a / nu

trigger = trigger * a

-----
'
' A.3 Print out the results to the screen or to a disk file.
'
-----

for output_choice = 1 to 2

if (output_choice = 1) then open Cons for output as #2
if (output_choice = 2) then open "output.dat" for output as #2

print #2, "*****"
print #2, "*" * 60
print #2, "*" Output for Program STOPSIGN.bas *
print #2, "*" * 60
print #2, "*****"
print #2, " "
print #2, "I. Input "
print #2, " "
print #2, " A. Fluid Flow Specifications "
print #2, " 1. Uniform flow velocity (m/s): ", uniform_velocity
print #2, " 2. Initial angle of flow (degrees): ", angle_initial * 180 / PI
print #2, " "
print #2, " B. Ambient Fluid Properties "
print #2, " 1. Fluid (0 = air, 1 = water): ", fluid_choice
print #2, " 2. Ambient pressure (Pa): ", p_ambient
print #2, " 3. Ambient temperature (K): ", T_ambient

```

```

print #2, "
print #2, " C. Stopsign Mechanical Properties
print #2, " 1. Torsional strength (Pa): ", torsional_strength
print #2, " 2. Moment of inertia (kgm^2): ", stopsign_MOI
print #2, " 3. Torsional damping (Nms): ", torsional_damping
print #2, " 4. Torsional stiffness (Nm): ", torsional_stiffness
print #2, " 5. Stopsign width (m): ", stopsign_width
print #2, "
print #2, " D. Plotting Specifications
print #2, " 1. Number of streamline points to plot: ", N_points
print #2, "
print #2, " E. Simulation Time
print #2, " 1. Maximum time for simulation (s): ", t_max
print #2, " 2. Time increment for simulation (s): ", t_increment
print #2, " 3. Print to screen (1) or output.dat (2): ", output_choice
print #2, " 4. Choice of plane to display: ", plane_choice
print #2, " 5. Choice to show vortices ( 1 for yes): ", V_onoff_choice
print #2, " F. Vortex Properties
print #2, " 1. Number of vortices to trigger asymmetry: ", trigger
print #2, " 2. Vortex dissipation factor: ", dissipation
print #2, " 3. Distance at which to eliminate vortices: ", distant
print #2, " 4. Amalgamation distance: ", amalg_dist
print #2, " 5. Initial guess for release point: ", offset

print #2, "
print #2, "II. Output
print #2, "
print #2, " A. Preliminary Values
print #2, " 1. Cylinder radius (m): ", a
print #2, " 2. Horizontal plot range (m): ", x_min, x_max
print #2, " 3. Vertical plot range (m): ", y_min, y_max
print #2, " 4. Reynolds Number: ", Re'

close #2

next output_choice

print "
print " <Hit any key to obtain the plot on the screen!> "

'-----
'
' B. Conduct initial calculations..
'
'-----

'-----
'
' B.1 Define the scale of points shown on the surface of the
' cylinder or on the flat plate. This defines the number
' of pixels used to make circles to document the position
' of stagnation and separation points on the plots.
'
'-----

scale_of_points = 20

'-----
'
' B.2 Make the streamlines. Define the minimum and maximum values for
' the stream function and the equipotential function at the edges
' of the display box.
'
'-----

n = N_radii_width
m = N_radii_height

phi_min = -a * (m + 1/m)
phi_max = a * (m + 1/m)

psi_min = -a * (n - 1/n)

```

```

psi_max = a * (n - 1/n)

k = 1
for streamfunction = psi_min to psi_max step (psi_max - psi_min) / N_streamlines
  psi(k) = streamfunction
  k += 1
next streamfunction

if (fluid_choice = 2) then
  density = 1.0
  uniform_velocity = 1.0
end if

'-----
'
' C. Begin the iteration in time.
'-----

N_vortices = 0:          v = 0          'set number of vortices and vortex index
vortex_strength(1).x = 0: vortex_strength(1).y = 0

'-----
'
' C.1 Initially set the display of all points on the plot to off.
'-----

F_choice = 0
R_choice = 0
T_choice = 0
B_choice = 0
LT_choice = 0
RT_choice = 0
LRP_choice = 0
RRP_choice = 0

FRONT_choice = 1
BACK_choice = 0
S_choice = 0
axis_choice = 1

Q_choice = 0

'-----
'
' C.2 Begin the loop in time for stopsign flutter.
'-----

Screen 20, 2

deflection_degrees = 0
deflection_angle = deflection_degrees * PI / 180

for tiempo = 0 to t_max step t_increment

  key_choice = inkey() 'Set key commands for displaying certain planes and points.

  if (key_choice = "1") then plane_choice = 1
  if (key_choice = "2") then plane_choice = 2
  if (key_choice = "3") then plane_choice = 3
  if (key_choice = "4") then plane_choice = 4
  if (key_choice = "5") then plane_choice = 5
  if (key_choice = "6") then plane_choice = 6
  if (key_choice = "a") then axis_choice = flipflop(axis_choice)
  if (key_choice = "s") then sleep
  if (key_choice = "f") then F_choice = flipflop(F_choice)
  if (key_choice = "r") then R_choice = flipflop(R_choice)
  if (key_choice = "t") then T_choice = flipflop(T_choice)
  if (key_choice = "b") then B_choice = flipflop(B_choice)
  if (key_choice = "L") then LT_choice = flipflop(LT_choice)
  if (key_choice = "R") then RT_choice = flipflop(RT_choice)
  if (key_choice = "P") then LRP_choice = flipflop(LRP_choice)

```

```

if (key_choice = "Z") then RRP_choice = flipflop(RRP_choice)

if (key_choice = "A") then FRONT_choice = flipflop(FRONT_choice)
if (key_choice = "K") then BACK_choice = flipflop(BACK_choice)
if (key_choice = "S") then S_choice = flipflop(S_choice)
if (key_choice = "V") then V_choice = flipflop(V_choice)

if (key_choice = "q") then Q_choice = 1

if (Key_choice = "C") then      'Set key commands for displaying Cp and Force vs. r plots

    plane_choice = 0
    plot_choice = 1
end if

if (Key_choice = "F") then
    plane_choice = 0
    plot_choice = 2
end if

'-----
'
' C.3 Update deflection angle based on plate dynamics and compute stress.
'-----
deflection_angle_older = deflection_angle_old
deflection_angle_old = deflection_angle
denominator = stopsign_MOI + torsional_damping*t_increment

value1.x = 2*stopsign_MOI - torsional_stiffness*t_increment^2 + torsional_damping*t_increment
value2.x = stopsign_MOI
value3.x = t_increment^2

deflection_angle = (deflection_angle_old*value1.x - deflection_angle_older*value2.x -
MOMENT*value3.x)/denominator

torsional_stress = 72e9 * a / 3 * deflection_angle 'tau=theta*G*c/H

open "deflection-angle" for append as #5      'Deflection angle data file
print #5, "", stopsign_width, uniform_velocity, stopsign_MOI, torsional_damping,
torsional_stiffness, tiempo, deflection_angle
close #5

'-----
'
' C.3 Generate points for the graphs at the tips of the stopsign.
' These points are defined in the z(5) plane (because the
' plate is lying flat and it's easy to do). We have to
' then compute back through the previous four planes and finish
' by calculating their positions in z(6).
'
' LT = Left tip (bottom) of the plate.
' RT = Right tip (top) of the plate.
'-----

LT(5).x = -2 * a:          LT(5).y = 0
RT(5).x = 2 * a:          RT(5).y = 0

LT(4).x = -a:             LT(4).y = 0
RT(4).x = a:              RT(4).y = 0

LT(3) = C_mult(i, LT(4))
RT(3) = C_mult(i, RT(4))

angle.x = deflection_angle:      angle.y = 0

LT(2) = C_mult( LT(3), C_exp( C_mult( C_neg(i), angle) ) )
RT(2) = C_mult( RT(3), C_exp( C_mult( C_neg(i), angle) ) )

angle.x = angle_initial:      angle.y = 0

```

```

LT(1) = C_mult( LT(2), C_exp( C_mult( C_neg(i), angle) ) )
RT(1) = C_mult( RT(2), C_exp( C_mult( C_neg(i), angle) ) )

angle.x = (PI/2 - deflection_angle):      angle.y = 0

LT(6) = C_mult( LT(5), C_exp( C_mult( i, angle) ) )
RT(6) = C_mult( RT(5), C_exp( C_mult( i, angle) ) )

'-----
'
' C.4 Introduce the discrete vortices. In each time step, introduce
' two vortices, one near the left tip and right tip in the Z5 plane.
' The vortex strength is equal to 0.5 * U_s^2 * t where U_s is the
' velocity in the flow at the surface of the plate (?). The
' vortices are released at a distance "m" away from the
' stagnation point normal to the surface of the cylinder in the
' Z1 plane.
'-----

v += 1

'-----
'
' C.4.1 Update positions of all existing vortices.
'-----

'
' C.4.1.1 Find the velocity at each vortex's position in the
' z-6 plane and update its position.
'-----

for k = 1 to N_vortices
    C_sum.x = 0:          C_sum.y = 0
    one.x = 1:          one.y = 0
    a2.x = a*a:        a2.y = 0
C_uniform_velocity.x = uniform_velocity:    C_uniform_velocity.y = 0

    for j = 1 to N_vortices
        if j <> k then
            value1 = C_sub(vortex_position(1,k), vortex_position(1,j))
            value1 = C_div(one, value1)

            value2 = C_div(a2, C_conj(vortex_position(1,j)))
            value2 = C_sub(vortex_position(1,k), value2)
            value2 = C_div(one, value2)

            value3 = C_sub(value1, value2)
            value3 = C_mult(vortex_strength(j), value3)

            C_sum = C_add(C_sum, value3)
        end if
    next j

    value4.x = 2*PI:          value4.y = 0
    value4 = C_div(i, value4)
    value4 = C_neg(C_mult(value4, C_sum))          'discrete vortex portion of dw/dz1

    value5 = C_mult(vortex_position(1,k),vortex_position(1,k))
    value5 = C_div(a2, value5)
    value5 = C_sub(one, value5)
    value5 = C_mult(C_uniform_velocity, value5)    'inviscid flow portion of dw/dz1

    dw_dz(1,k) = (C_add(value5, value4))

    angle.x = angle_initial:          angle.y = 0 'dw/dz2 = dw/dz1 * dz1/dz2
    dw_dz(2,k) = C_mult(dw_dz(1,k), C_exp( C_mult(C_neg(i), angle) ) )

    angle.x = deflection_angle:          angle.y = 0 'dw/dz3 = dw/dz2 * dz2/dz3
    dw_dz(3,k) = C_mult(dw_dz(2,k), C_exp( C_mult(C_neg(i), angle) ) )

    dw_dz(4,k) = C_mult(i, dw_dz(3,k) )          'dw/dz4 = dw/dz3 * dz3/dz4

```

```

value1.x = a * a:                               value1.y = 0 `dw/dz5 = dw/dz4 * dz4/dz5
dw_dz(5,k) = C_mult(dw_dz(4,k), C_div(one, C_sub(one, C_div( value1,
C_mult(vortex_position(4,k),vortex_position(4,k))))))

angle.x = deflection_angle:   angle.y = 0       `dw/dz6 = dw/dz5 * dz5/dz6
dw_dz(6,k) = C_mult(dw_dz(5,k), C_mult(C_neg(i), C_exp(C_mult(i, angle))))

vortex_velocity(6,k) = C_conj(dw_dz(6,k)) `velocity = complex conjugate of (dw/dz6=u-iv)

vortex_position(6,k).x += vortex_velocity(6,k).x * t_increment
vortex_position(6,k).y += vortex_velocity(6,k).y * t_increment

next k

'-----
'
' C.4.1.2 Transform positions of all vortices through all planes.
'-----

for k = 1 to N_vortices                          `Transform the vortex_positions from z-6 to z-1
angle.x = (PI/2 - deflection_angle):             angle.y = 0
w.x      = vortex_position(5,k).x:               w.y      = vortex_position(5,k).y
vortex_position(5,k) = C_mult(vortex_position(6,k), C_exp(C_mult(C_neg(i),angle)))
if k < N_vortices and abs(vortex_position(5,k).x) < 2 * a then
    if (vortex_position(5,k).y * w.y) < 0 then vortex_strength(k).x = 0 ')>0 then
vortex_strength(k).x = 0
end if                                           `Prevent vortices from crossing the plate, eliminate

w.x      = vortex_position(5,k).x: w.y      = vortex_position(5,k).y
value.x  = 4 * a * a:                          value.y = 0
argument = C_sqr( C_sub( C_mult(w, w), value) )
value.x  = 2:                                  value.y = 0
if vortex_position(5,k).x > 0 then
    vortex_position(4,k) = C_div( C_add( w, argument), value)      `Plus argument
else
    vortex_position(4,k) = C_div( C_sub( w, argument), value)      `Minus argument
end if

vortex_position(3,k) = C_mult(i, vortex_position(4,k))

angle.x = deflection_angle:                     angle.y = 0
vortex_position(2,k) = C_mult(vortex_position(3,k), C_exp( C_mult( C_neg(i), angle) ) )

angle.x = angle_initial:                       angle.y = 0
vortex_position(1,k) = C_mult(vortex_position(2,k), C_exp( C_mult( C_neg(i), angle) ) )

next k

'-----
'
' C.4.1.1 Amalgamation of adjacent vortices.
'-----

for k = 1 to N_vortices
    for j = k + 1 to N_vortices

        if C_distance(vortex_position(6,k), vortex_position(6,j)) < a/10 then
            vortex_strength(k).x += vortex_strength(j).x
            vortex_strength(j).x = 0
        end if

    next j
next k

'-----
'
' C.4.1.2 Dissipate vortices with time.
'-----

for k = 1 to N_vortices
    vortex_strength(k).x *= dissipation
next k

```

```

-----
'
' C.4.1.2 Zero out distant vortices.
'
-----

for k = 1 to N_vortices
  if vortex_position(6,k).x > a * distant then
    vortex_strength(k).x = 0
  end if
next k

-----
'
' C.4.1.3 Garbage collection of zero-strength vortices.
' 1) Identify the indices of each zero-strength vortex
' 2) Create placeholder matrices and fill them with nonzero vortices
' 3) Redefine vortex_position, velocity, and strength
'
-----

j = 1
for k = N_vortices to 1 step -1
  if abs(vortex_strength(k).x) < 1e-5 then
    zero_vortices_index(j) = k
    j += 1
  end if
next k
number_of_zero_vortices = j - 1

if number_of_zero_vortices > 100 then
  for k = 1 to number_of_zero_vortices
    for j = 1 to N_vortices - k
      for n = 1 to 6
        if j > zero_vortices_index(k) - 1 then
          vortex_position(n,j) = vortex_position(n,j+1)
          vortex_velocity(n,j) = vortex_velocity(n,j+1)
          vortex_strength(j) = vortex_strength(j+1)
        end if
      next n
    next j
  next k

  for j = N_vortices - number_of_zero_vortices + 1 to N_vortices
    for n = 1 to 6
      vortex_position(n,j).x = 0: vortex_position(n,j).y = 0
      vortex_velocity(n,j).x = 0: vortex_velocity(n,j).y = 0
      vortex_strength(j).x = 0: vortex_strength(j).y = 0
    next n
  next j

  N_vortices -= number_of_zero_vortices
  v = N_vortices + 1
end if

-----
'
' C.4.2 Solve for the position and velocity of the LT nascent vortex.
'
-----

vortex_position(1,v) = LT(1) 'Initial guess
vortex_position(4,v).x = LT(4).x -offset*a : vortex_position(4,v).y = LT(4).y

-----
'
' C.4.2.1 Solve for the complex velocity at the release point
' of the LT nascent vortex in the Z1 plane.
'
-----

C_sum.x = 0: C_sum.y = 0
one.x = 1: one.y = 0
a2.x = a*a: a2.y = 0
C_uniform_velocity.x = uniform_velocity: C_uniform_velocity.y = 0

for j = 1 to N_vortices

```



```

value1 = C_sub(LT(1), vortex_position(1,j))
value1 = C_div(one, value1)

value2 = C_div(a2, C_conj(vortex_position(1,j)))
value2 = C_sub(LT(1), value2)
value2 = C_div(one, value2)

value3 = C_sub(value1, value2)
value3 = C_mult(vortex_strength(j), value3)

C_sum = C_add(C_sum, value3)

next j

value4.x = 2*PI:          value4.y = 0
value4 = C_div(i, value4)
value4 = C_neg(C_mult(value4, C_sum) )          'discrete vortex portion of dw/dz1

value5 = C_mult(vortex_position(1,v),vortex_position(1,v))

value5 = C_div(a2, value5)
value5 = C_sub(one, value5)
value5 = C_mult(C_uniform_velocity, value5)    'inviscid flow portion of dw/dz1

dw_dz(1,v) = (C_add(value5, value4))          'assume velocity at release point is
equal to the velocity at the surface (ignore no-slip condition)

'-----
'
' C.4.2.2 Transform it to the Z6 plane to solve for U_s. (dw/dz6)
'
'     1. Calculate velocity of the flow at the release point (U_s)
'     2. Calculate strength of nascent vortex (circulation)
'-----

angle.x = angle_initial:          angle.y = 0 'dw/dz2 = dw/dz1 * dz1/dz2
dw_dz(2,v) = C_mult(dw_dz(1,v), C_exp( C_mult(C_neg(i), angle) ) )

angle.x = deflection_angle:       angle.y = 0 'dw/dz3 = dw/dz2 * dz2/dz3
dw_dz(3,v) = C_mult(dw_dz(2,v), C_exp( C_mult(C_neg(i), angle) ) )

dw_dz(4,v) = C_mult(i, dw_dz(3,v) )          'dw/dz4 = dw/dz3 * dz3/dz4

value1.x = a * a:                  value1.y = 0 'dw/dz5 = dw/dz4 * dz4/dz5
dw_dz(5,v) = C_mult(dw_dz(4,v), C_div(one, C_sub(one, C_div( value1,
C_mult(vortex_position(4,v),vortex_position(4,v))))))

angle.x = deflection_angle:       angle.y = 0          'dw/dz6 = dw/dz5 * dz5/dz6
dw_dz(6,v) = C_mult(dw_dz(5,v), C_mult(C_neg(i), C_exp(C_mult(i, angle))))

vortex_velocity(6,v) = C_conj(dw_dz(6,v))    'velocity = complex conjugate of (dw/dz6=u-iv)

U_s = sqr(vortex_velocity(6,v).x * vortex_velocity(6,v).x + vortex_velocity(6,v).y *
vortex_velocity(6,v).y)          'U_s = magnitude of velocity

circulation = 0.5 * U_s * U_s * t_increment 'circulation = strength of a nascent vortex

if V_onoff_choice = 1 then
    if (LT(6).x*vortex_velocity(6,v).y-LT(6).y*vortex_velocity(6,v).x) > 0 then
'vortex strength will remain constant with time and displacement of vortex (in all planes)
        vortex_strength(v).x = circulation:    vortex_strength(v).y = 0          'ccw
    else
        vortex_strength(v).x = -circulation:   vortex_strength(v).y = 0          'cw
    endif

    if v < trigger then                'trigger asymmetry
        vortex_strength(v).x = 0:          vortex_strength(v).y = 0
    endif

'replace the big vortices with realistically-sized ones
if abs(vortex_strength(v).x) > 5 then vortex_strength(v).x = vortex_strength(v-2).x

else
    vortex_strength(v).x = 0:            vortex_strength(v).y = 0
endif

```

```

-----
'
' C.4.2.3 Calculate release distance of the nascent vortex.
'
-----

one.x = 1: one.y = 0
angle.x = atan2(LT(1).y , LT(1).x): angle.y = 0
value.x = abs(circulation) / (2 * PI * U_s): value.y = 0 'value = |circulation j| /
(2*PI*U_s j) = velocity of a flow field due to vortex
value = C_div(C_add(one, value), C_sub(one, value))'1 + mj = (1 + value) / (1 - value)

vortex_position(1,v) = C_mult(value, C_exp(C_mult(i, angle)))
LRP(1) = vortex_position(1,v)
-----
'
' C.4.3 Repeat for the right tip nascent vortex.
'
-----
N_vortices += 1: v += 1
vortex_position(1,v) = RT(1): vortex_position(4,v).x = RT(4).x + offset*a
vortex_position(4,v).y = RT(4).y 'Initial guess
-----
'
' C.4.3.1 Solve for the complex velocity at the release point
' of the LT nascent vortex in the Z1 plane.
'
-----

C_sum.x = 0: C_sum.y = 0
one.x = 1: one.y = 0
a2.x = a*a: a2.y = 0
C_uniform_velocity.x = uniform_velocity: C_uniform_velocity.y = 0

for j = 1 to N_vortices

value1 = C_sub(RT(1), vortex_position(1,j))
value1 = C_div(one, value1)

value2 = C_div(a2, C_conj(vortex_position(1,j)))
value2 = C_sub(RT(1), value2)
value2 = C_div(one, value2)

value3 = C_sub(value1, value2)
value3 = C_mult(vortex_strength(j), value3)

C_sum = C_add(C_sum, value3)

next j

value4.x = 2*PI: value4.y = 0
value4 = C_div(i, value4)
value4 = C_neg(C_mult(value4, C_sum) ) 'discrete vortex portion of dw/dz1

value5 = C_mult(vortex_position(1,v),vortex_position(1,v))
value5 = C_div(a2, value5)
value5 = C_sub(one, value5)
value5 = C_mult(C_uniform_velocity, value5) 'inviscid flow portion of dw/dz1

dw_dz(1,v) = (C_add(value5, value4)) 'assume velocity at release point is equal to
the velocity at the surface (ignore no-slip condition)
-----
'
' C.4.3.2 Transform it to the Z6 plane to solve for U_s. (dw/dz6)
'
' 1. Calculate velocity of the flow at the release point (U_s)
' 2. Calculate strength of nascent vortex (circulation)
'
-----

angle.x = angle_initial: angle.y = 0 'dw/dz2 = dw/dz1 * dz1/dz2
dw_dz(2,v) = C_mult(dw_dz(1,v), C_exp( C_mult(C_neg(i), angle) ) )

angle.x = deflection_angle: angle.y = 0 'dw/dz3 = dw/dz2 * dz2/dz3
dw_dz(3,v) = C_mult(dw_dz(2,v), C_exp( C_mult(C_neg(i), angle) ) )

```

```

dw_dz(4,v) = C_mult(i, dw_dz(3,v) )           `dw/dz4 = dw/dz3 * dz3/dz4

value1.x = a * a:                             value1.y = 0 `dw/dz5 = dw/dz4 * dz4/dz5
dw_dz(5,v) = C_mult(dw_dz(4,v), C_div(one, C_sub(one, C_div( value1,
C_mult(vortex_position(4,v),vortex_position(4,v))))))

angle.x = deflection_angle:                   angle.y = 0 `dw/dz6 = dw/dz5 * dz5/dz6
dw_dz(6,v) = C_mult(dw_dz(5,v), C_mult(C_neg(i), C_exp(C_mult(i, angle))))

vortex_velocity(6,v) = C_conj(dw_dz(6,v)) `velocity = complex conjugate of (dw/dz6=u-iv)

U_s = sqr(vortex_velocity(6,v).x * vortex_velocity(6,v).x + vortex_velocity(6,v).y *
vortex_velocity(6,v).y)           `U_s = magnitude of velocity

circulation = 0.5 * U_s * U_s * t_increment   `circulation = strength of a nascent vortex

if V_onoff_choice = 1 then
  if (RT(6).x*vortex_velocity(6,v).y-RT(6).y*vortex_velocity(6,v).x) > 0 then
'vortex strength will remain constant with time and displacement of vortex (in all planes)
    vortex_strength(v).x = circulation:       vortex_strength(v).y = 0           `ccw
  else
    vortex_strength(v).x = -circulation:     vortex_strength(v).y = 0           `cw
  endif

  'replace the big vortices with realistically-sized ones
  if abs(vortex_strength(v).x) > 5 then vortex_strength(v).x = vortex_strength(v-2).x
else
  vortex_strength(v).x = 0:                   vortex_strength(v).y = 0
endif

'-----
'|
'| C.4.3.3 Calculate release distance of the nascent vortex.
'|
'-----

one.x = 1:                                   one.y = 0
angle.x = atan2(RT(1).y, RT(1).x):           angle.y = 0
value.x = abs(circulation) / (2 * PI * U_s):  value.y = 0
value = C_div(C_add(one, value), C_sub(one, value)) `1 + mj = (1 + value) / (1 - value)

vortex_position(1,v) = C_mult(value, C_exp(C_mult(i, angle)))
RRP(1) = vortex_position(1,v)

'-----
'|
'| C.4.4 Transform both points through all planes.
'|
'-----

for j = v-1 to v
  angle.x = angle_initial:                   angle.y = 0           `z1 to z2
  vortex_position(2,j) = C_mult(vortex_position(1,j), C_exp( C_mult( i, angle) ) )

  angle.x = deflection_angle:               angle.y = 0           `z2 to z3
  vortex_position(3,j) = C_mult(vortex_position(2,j), C_exp( C_mult( i, angle) ) )

  vortex_position(4,j) = C_mult( C_neg(i),vortex_position(3,j) )

  value.x = a * a:                           value.y = 0           `z4 to z5
  vortex_position(5,j) = C_add( vortex_position(4,j), C_div( value, vortex_position(4,j)))

  angle.x = PI/2 - deflection_angle:        angle.y = 0           `z5 to z6
  vortex_position(6,j) = C_mult(vortex_position(5,j), C_exp(C_mult(i, angle)) )
next j

N_vortices += 1

total_circulation = 0           `Calculalte the total circulation in the flow field - should be 0
for j = 1 to N_vortices
  total_circulation += vortex_strength(j).x
next j

'-----
'|
'| C.5 Next, figure out the positions of the position of points on
'| the original circle in the Z(1) plane. This includes:
'|
'-----

```

```

'      T = Top of the cylinder.
'      B = Bottom of the cylinder.
'      F = Forward stagnation point.
'      R = Rearward stagnation point.
'      LRP = Left release point.
'      RRP = Right release point.
'
'-----

```

```

'-----
'
' C.5.1 Z1-Plane.
'-----

```

```

F(1).x = -a:   F(1).y = 0
R(1).x =  a:   R(1).y = 0
T(1).x =  0:   T(1).y = a
B(1).x =  0:   B(1).y = -a

```

```

'-----
'
' C.5.2 Z2-Plane.
'-----

```

```

angle.x = angle_initial:      angle.y = 0

F(2) = C_mult( F(1), C_exp( C_mult( i, angle) ) )
R(2) = C_mult( R(1), C_exp( C_mult( i, angle) ) )
T(2) = C_mult( T(1), C_exp( C_mult( i, angle) ) )
B(2) = C_mult( B(1), C_exp( C_mult( i, angle) ) )
LRP(2) = C_mult(LRP(1), C_exp( C_mult( i, angle) ) )
RRP(2) = C_mult(RRP(1), C_exp( C_mult( i, angle) ) )

```

```

'-----
'
' C.5.3 Z3-Plane.
'-----

```

```

angle.x = deflection_angle:    angle.y = 0

F(3) = C_mult( F(2), C_exp( C_mult( i, angle) ) )
R(3) = C_mult( R(2), C_exp( C_mult( i, angle) ) )
T(3) = C_mult( T(2), C_exp( C_mult( i, angle) ) )
B(3) = C_mult( B(2), C_exp( C_mult( i, angle) ) )
LRP(3) = C_mult(LRP(2), C_exp( C_mult( i, angle) ) )
RRP(3) = C_mult(RRP(2), C_exp( C_mult( i, angle) ) )

```

```

'-----
'
' C.5.4 Z4-Plane.
'-----

```

```

F(4) = C_mult( C_neg(i), F(3) )
R(4) = C_mult( C_neg(i), R(3) )
T(4) = C_mult( C_neg(i), T(3) )
B(4) = C_mult( C_neg(i), B(3) )
LRP(4) = C_mult( C_neg(i), LRP(3) )
RRP(4) = C_mult( C_neg(i), RRP(3) )

```

```

'-----
'
' C.5.5 Z5-Plane.
'-----

```

```

value.x = a * a:              value.y = 0

F(5) = C_add( F(4), C_div( value, F(4) ) )
R(5) = C_add( R(4), C_div( value, R(4) ) )
T(5) = C_add( T(4), C_div( value, T(4) ) )
B(5) = C_add( B(4), C_div( value, B(4) ) )
LRP(5) = C_add(LRP(4), C_div( value,LRP(4) ) )
RRP(5) = C_add(RRP(4), C_div( value,RRP(4) ) )

```

```

-----
'
' C.5.6 Z6-Plane.
'
-----

angle.x = PI/2 - deflection_angle:    angle.y = 0

F(6)  = C_mult( F(5), C_exp(C_mult(i, angle)) )
R(6)  = C_mult( R(5), C_exp(C_mult(i, angle)) )
T(6)  = C_mult( T(5), C_exp(C_mult(i, angle)) )
B(6)  = C_mult( B(5), C_exp(C_mult(i, angle)) )
LRP(6) = C_mult(LRP(5), C_exp(C_mult(i, angle)) )
RRP(6) = C_mult(RRP(5), C_exp(C_mult(i, angle)) )

-----
'
' C.6 Identify labels for the plots in each plane.
'
-----

plot_label(1, 1) = "Z1-Plane"
plot_label(2, 1) = "Z2-Plane"
plot_label(3, 1) = "Z3-Plane"
plot_label(4, 1) = "Z4-Plane"
plot_label(5, 1) = "Z5-Plane"
plot_label(6, 1) = "Z6-Plane"

plot_label(1, 2) = "Deflection Angle:    " & (deflection_angle * 180 / PI) & " (deg)"
plot_label(1, 3) = "Moment of Inertia:  " & (stopsign_MOI) & " (kg * m^2)"
plot_label(1, 4) = "Damping Coefficient: " & (torsional_damping) & " (Nms)"
plot_label(1, 5) = "Torsional Rigidity:  " & (torsional_stiffness) & " (Nm)"
plot_label(1, 6) = "Streamline Velocity: " & (uniform_velocity) & " (m/s)"
plot_label(1, 7) = "Plate Width:        " & (stopsign_width) & " (m)"
plot_label(1, 8) = "Time (s):          " & (tiempo)

-----
'
' C.7 Identify points on the front and back of the stopsign. Define
' these points in the z(1) plane and transform them to the other
' planes.
'
-----

' -----
'
' C.7.1 Z1-Plane.
'
-----

angle_RT = C_arg(RT(1))
angle_LT = C_arg(LT(1))

    angle_increment = ( angle_LT - angle_RT ) / N_points
    surface_angle   = angle_LT
    for j = 1 to N_points
        value.x = a:                value.y = 0
        angle.x = surface_angle:    angle.y = 0
        FRONT(1, j) = C_mult(value, C_exp( C_mult( i, angle) ) )
        surface_angle -= angle_increment
    next j
    surface_angle   = angle_LT
    for j = 1 to N_points
        value.x = a:                value.y = 0
        angle.x = surface_angle:    angle.y = 0
        BACK(1, j) = C_mult(value, C_exp( C_mult( i, angle) ) )
        surface_angle += angle_increment
    next j

-----
'
' C.7.2 Z2-Plane.
'
-----

```

```

angle.x = angle_initial:          angle.y = 0

for j = 1 to N_points
  FRONT(2,j) = C_mult( FRONT(1,j), C_exp( C_mult( i, angle) ) )
  BACK(2,j) = C_mult( BACK(1,j), C_exp( C_mult( i, angle) ) )
next j

-----
'
' C.7.3 Z3-Plane.
'
-----

angle.x = deflection_angle:        angle.y = 0

for j = 1 to N_points
  FRONT(3,j) = C_mult( FRONT(2,j), C_exp( C_mult( i, angle) ) )
  BACK(3,j) = C_mult( BACK(2,j), C_exp( C_mult( i, angle) ) )
next j

-----
'
' C.7.4 Z4-Plane.
'
-----

for j = 1 to N_points
  FRONT(4,j) = C_mult( C_neg(i), FRONT(3,j) )
  BACK(4,j) = C_mult( C_neg(i), BACK(3,j) )
next j

-----
'
' C.7.5 Z5-Plane.
'
-----

value.x = a * a:                  value.y = 0

for j = 1 to N_points
  FRONT(5,j) = C_add( FRONT(4,j), C_div( value, FRONT(4,j) ) )
  BACK(5,j) = C_add( BACK(4,j), C_div( value, BACK(4,j) ) )
next j

-----
'
' C.7.6 Z6-Plane.
'
-----

angle.x = PI/2 - deflection_angle:  angle.y = 0

for j = 1 to N_points
  FRONT(6,j) = C_mult( FRONT(5,j), C_exp( C_mult( i, angle) ) )
  BACK(6,j) = C_mult( BACK(5,j), C_exp( C_mult( i, angle) ) )
next j

-----
'
' C.7.7 In the Z6-plane, what are the distances between each
' point in FRONT and BACK and what is the velocity and
' pressure in each of these point regions.
'
' FRONT_delta_s(j) = distance between point "j" and
' "j+1" on the front of the flat
' plate (stop sign).
' BACK_delta_s(j) = distance between point "j" and
' "j+1" on the back of the flat
' plate (stop sign).
'
-----

```

```

FRONT_delta_s(1) = 0
BACK_delta_s(1) = 0

for j = 2 to (N_points - 1)
    FRONT_delta_s(j) = C_distance( FRONT(6, j), FRONT(6, j+1) )
    BACK_delta_s(j) = C_distance( BACK(6, j), BACK(6, j+1) )
next j

FRONT_delta_s(N_points) = C_distance( FRONT(6, N_points), LT(6) )
BACK_delta_s(N_points) = C_distance( BACK(6, N_points), LT(6) )

'-----
'
' C.7.8 Print out the points on the front and the back, the
' distances between points, the velocity at each point,
' and the pressure coefficient.
'
' FRONT_V(j) = velocity vector at each point on the front.
' BACK_V(j) = velocity vector at each point on the back.
' FRONT_Cp(j) = pressure coefficient at each front point.
' BACK_Cp(j) = pressure coefficient at each back point.
' FORCE = net force on the stopsign (fx + i fy).
' MOMENT = net moment about the stopsign center.
'-----

open "check" for output as #3

print #3, " "
print #3, "Right Tip Position: x6, y6 = ", LT(6).x, LT(6).y
print #3, "Left Tip Position: x6, y6 = ", RT(6).x, LT(6).y
print #3, " "
print #3, "FRONT "
for j = 1 to N_points
    print #3, " j, x, y, ds: ", j, FRONT(6, j).x, FRONT(6, j).y, FRONT_delta_s(j)
next j

print #3, " "
print #3, "Right Tip Position: x6, y6 = ", LT(6).x, LT(6).y
print #3, "Left Tip Position: x6, y6 = ", RT(6).x, LT(6).y
print #3, " "
print #3, "BACK "
for j = 1 to N_points
    print #3, " j, x, y, ds: ", j, BACK(6, j).x, BACK(6, j).y, BACK_delta_s(j)
next j

'-----
'
' C.7.8.1 Compute the velocity at each point and the pressure coefficient.
'-----

'-----
'
' C.7.8.1.1 Front surface.
'-----

for k = 1 to N_points

    C_sum.x = 0: C_sum.y = 0
    one.x = 1: one.y = 0
    a2.x = a*a: a2.y = 0
C_uniform_velocity.x = uniform_velocity: C_uniform_velocity.y = 0

for j = 1 to N_vortices

    value1 = C_sub(FRONT(1,k), vortex_position(1,j))
    value1 = C_div(one, value1)

    value2 = C_div(a2, C_conj(vortex_position(1,j)))
    value2 = C_sub(FRONT(1,k), value2)
    value2 = C_div(one, value2)

```

```

value3 = C_sub(value1, value2)
value3 = C_mult(vortex_strength(j), value3)
C_sum = C_add(C_sum, value3)

next j

value4.x = 2*PI:          value4.y = 0
value4 = C_div(i, value4)
value4 = C_neg(C_mult(value4, C_sum) )           'discrete vortex portion of dw/dz1

value5 = C_mult(FRONT(1,k),FRONT(1,k))

value5 = C_div(a2, value5)
value5 = C_sub(one, value5)
value5 = C_mult(C_uniform_velocity, value5)     'inviscid flow portion of dw/dz1

dw_dz(1,k) = (C_add(value5, value4))           'assume velocity at release point is
equal to the velocity at the surface (ignore no-slip condition)

angle.x = angle_initial:          angle.y = 0  'dw/dz2 = dw/dz1 * dz1/dz2
dw_dz(2,k) = C_mult(dw_dz(1,k), C_exp( C_mult(C_neg(i), angle) ) )

angle.x = deflection_angle:       angle.y = 0  'dw/dz3 = dw/dz2 * dz2/dz3
dw_dz(3,k) = C_mult(dw_dz(2,k), C_exp( C_mult(C_neg(i), angle) ) )

dw_dz(4,k) = C_mult(i, dw_dz(3,k) )           'dw/dz4 = dw/dz3 * dz3/dz4

value1.x = a * a:                 value1.y = 0  'dw/dz5 = dw/dz4 * dz4/dz5
dw_dz(5,k) = C_mult(dw_dz(4,k), C_div(one, C_sub(one, C_div( value1,
C_mult(FRONT(4,k),FRONT(4,k)))))) 'dw_dz(4,k),dw_dz(4,k))))

angle.x = deflection_angle:       angle.y = 0  'dw/dz6 = dw/dz5 * dz5/dz6
dw_dz(6,k) = C_mult(dw_dz(5,k), C_mult(C_neg(i), C_exp(C_mult(i, angle))))

FRONT_V(k) = C_conj(dw_dz(6,k))

V2 = (FRONT_V(k).x)^2 + (FRONT_V(k).y)^2

' Take care of infinite velocities predicted near singularities.

if (V2 > 100 * uniform_velocity^2) then
  FRONT_V(k).x = 0.0
  FRONT_V(k).y = 0.0
  V2 = 0.0
end if

FRONT_Cp(k) = 1.0 - V2 / uniform_velocity^2

'-----
'
' C.7.8.1.1 Back surface.
'
'-----

C_sum.x =          0:          C_sum.y = 0
one.x =          1:          one.y = 0
a2.x =          a*a:          a2.y = 0
C_uniform_velocity.x = uniform_velocity:  C_uniform_velocity.y = 0

for j = 1 to N_vortices

value1 = C_sub(BACK(1,k), vortex_position(1,j))
value1 = C_div(one, value1)

value2 = C_div(a2, C_conj(vortex_position(1,j)))
value2 = C_sub(BACK(1,k), value2)
value2 = C_div(one, value2)

value3 = C_sub(value1, value2)
value3 = C_mult(vortex_strength(j), value3)

C_sum = C_add(C_sum, value3)

next j

```



```

value4.x = 2*PI:                value4.y = 0
value4 = C_div(i, value4)
value4 = C_neg(C_mult(value4, C_sum) )           'discrete vortex portion of dw/dz1
value5 = C_mult(BACK(1,k),BACK(1,k))

value5 = C_div(a2, value5)
value5 = C_sub(one, value5)
value5 = C_mult(C_uniform_velocity, value5)     'inviscid flow portion of dw/dz1
dw_dz(1,k) = (C_add(value5, value4))           'assume velocity at release point
is equal to the velocity at the surface (ignore no-slip condition)

angle.x = angle_initial:        angle.y = 0      'dw/dz2 = dw/dz1 * dz1/dz2
dw_dz(2,k) = C_mult(dw_dz(1,k), C_exp( C_mult(C_neg(i), angle) ) )

angle.x = deflection_angle:     angle.y = 0      'dw/dz3 = dw/dz2 * dz2/dz3
dw_dz(3,k) = C_mult(dw_dz(2,k), C_exp( C_mult(C_neg(i), angle) ) )

dw_dz(4,k) = C_mult(i, dw_dz(3,k) )             'dw/dz4 = dw/dz3 * dz3/dz4

value1.x = a * a:                value1.y = 0    'dw/dz5 = dw/dz4 * dz4/dz5
dw_dz(5,k) = C_mult(dw_dz(4,k), C_div(one, C_sub(one, C_div( value1,
C_mult(BACK(4,k),BACK(4,k))))))

angle.x = deflection_angle:     angle.y = 0      'dw/dz6 = dw/dz5 * dz5/dz6
dw_dz(6,k) = C_mult(dw_dz(5,k), C_mult(C_neg(i), C_exp(C_mult(i, angle))))

BACK_V(k) = C_conj(dw_dz(6,k))

V2 = (BACK_V(k).x)^2 + (BACK_V(k).y)^2

' Take care of infinite velocities predicted near singularities.

if (V2 > 100 * uniform_velocity^2) then
    BACK_V(k).x = 0.0
    BACK_V(k).y = 0.0
    V2 = 0.0
end if

BACK_Cp(k) = (1.0 - V2 / uniform_velocity^2)

TOTAL_Cp(k) = FRONT_Cp(k) - BACK_Cp(k)
next k

'-----
'
' C.7.9 Forces and Moments on the FRONT Surface.
'
'     Next, compute the total force in the x-direction and in the
'     y-direction. In the same loop, compute the moment about the
'     center of the plate.
'-----

FORCE.x = 0.0:                FORCE.y = 0.0
MOMENT = 0.0

x = FRONT(6, 1).x
y = FRONT(6, 1).y
radius = sqrt(x * x + y * y)
max_abscissa = radius

for j = 1 to N_points

x = FRONT(6, j).x
y = FRONT(6, j).y
radius = sqrt( x * x + y * y )

' I assume that c(p) = 1 - V^2/U(inf)^2 = (p - p(inf)) / (0.5 * density * U(inf)^2)
' c(p) = (p - 0) / (1) = p.

dF = FRONT_Cp(j) * FRONT_delta_s(j)
dFx = dF * Cos(deflection_angle)
dFy = - dF * sin(deflection_angle)

FRONT_dFx_dr(j) = dFx / FRONT_delta_s(j)

```

```

FRONT_dFy_dr(j) = dFy / FRONT_delta_s(j)
FRONT_dF_dr(j) = dF / FRONT_delta_s(j)

FORCE.x += dFx
FORCE.y += dFy

dM      = (x / radius) * dFy - (y / radius) * dFx

MOMENT += dM

next j

'-----
'
' C.7.9.1 Repeat for the BACK surface.
'-----

for j = 1 to N_points

x      = BACK(6, j).x
y      = BACK(6, j).y
radius = sqrt( x * x + y * y )

' I assume that c(p) = 1 - V^2/U(inf)^2 = (p - p(inf)) / (0.5 * density * U(inf)^2)
' c(p) = (p - 0) / (1) = p.

dF      = BACK_Cp(j) * BACK_delta_s(j)
dFx     = - dF * cos(deflection_angle)
dFy     = dF * sin(deflection_angle)

BACK_dFx_dr(j) = dFx / BACK_delta_s(j)
BACK_dFy_dr(j) = dFy / BACK_delta_s(j)
BACK_dF_dr(j)  = dF / BACK_delta_s(j)

FORCE.x += dFx
FORCE.y += dFy

dM      = (x / radius) * dFy - (y / radius) * dFx
MOMENT += dM

next j

close #3
open "plate-drag" for append as #6
print #6, "", stopsign_width, uniform_velocity, stopsign_MOI, torsional_damping,
torsional_stiffness, tiempo, FORCE.x, FORCE.y, MOMENT
close #6

'-----
'
' C.7.10 Pressure probe in wake.
'-----

probe_position(6).x = 12 * a:      probe_position(1).y = 0

'-----
'
' C.7.10.1 Position the probe in all 6 planes.
'-----

angle.x = (PI/2 - deflection_angle):      angle.y = 0
probe_position(5) = C_mult(probe_position(6), C_exp(C_mult(C_neg(i), angle)))

w.x      = probe_position(5).x: w.y      = probe_position(5).y
value.x  = 4 * a * a:      value.y = 0
argument = C_sqrt( C_sub( C_mult(w, w), value) )
value.x  = 2:      value.y = 0

'Prevent vortices from transforming to inside the cylinder
if probe_position(5).x > 0 then
    probe_position(4) = C_div( C_add( w, argument), value)      'Plus argument
else

```

```

    probe_position(4) = C_div( C_sub( w, argument), value)           'Minus argument
end if

probe_position(3) = C_mult(i, probe_position(4))

angle.x = deflection_angle:           angle.y = 0
probe_position(2) = C_mult(probe_position(3), C_exp( C_mult( C_neg(i), angle) ) )

angle.x = angle_initial:             angle.y = 0
probe_position(1) = C_mult(probe_position(2), C_exp( C_mult( C_neg(i), angle) ) )

'-----
'
' C.7.10.2 Calculate the velocity at that position in the z-6 plane.
'
'-----

    C_sum.x =           0:           C_sum.y = 0
    one.x =           1:           one.y = 0
    a2.x =           a*a:           a2.y = 0
C_uniform_velocity.x = uniform_velocity:           C_uniform_velocity.y = 0

for j = 1 to N_vortices
if j <> k then
value1 = C_sub(probe_position(1), vortex_position(1,j))
value1 = C_div(one, value1)

value2 = C_div(a2, C_conj(vortex_position(1,j)))
value2 = C_sub(probe_position(1), value2)
value2 = C_div(one, value2)

value3 = C_sub(value1, value2)
value3 = C_mult(vortex_strength(j), value3)

C_sum = C_add(C_sum, value3)
end if
next j

value4.x = 2*PI:           value4.y = 0
value4 = C_div(i, value4)
value4 = C_neg(C_mult(value4, C_sum))           'discrete vortex portion of dw/dz1

value5 = C_mult(probe_position(1), probe_position(1))
value5 = C_div(a2, value5)
value5 = C_sub(one, value5)
value5 = C_mult(C_uniform_velocity, value5)           'inviscid flow portion of dw/dz1

probe_velocity(1) = (C_add(value5, value4)) 'Not technically the velocity here; dw/dz1

angle.x = angle_initial:           angle.y = 0           'dw/dz2 = dw/dz1 * dz1/dz2
probe_velocity(2) = C_mult(probe_velocity(1), C_exp( C_mult(C_neg(i), angle) ) )

angle.x = deflection_angle:           angle.y = 0           'dw/dz3 = dw/dz2 * dz2/dz3
probe_velocity(3) = C_mult(probe_velocity(2), C_exp( C_mult(C_neg(i), angle) ) )

probe_velocity(4) = C_mult(i, probe_velocity(3) )           'dw/dz4 = dw/dz3 * dz3/dz4

value1.x = a * a:           value1.y = 0           'dw/dz5 = dw/dz4 * dz4/dz5
probe_velocity(5) = C_mult(probe_velocity(4), C_div(one, C_sub(one, C_div( value1,
C_mult(probe_position(4),probe_position(4))))))

angle.x = deflection_angle:           angle.y = 0           'dw/dz6 = dw/dz5 * dz5/dz6
probe_velocity(6) = C_mult(probe_velocity(5), C_mult(C_neg(i), C_exp(C_mult(i, angle))))

probe_velocity(6) = C_conj(probe_velocity(6))

j = tiempo / t_increment
probe_Cp(j) = 1.0 - (probe_velocity(6).x^2 + probe_velocity(6).y^2) / uniform_velocity^2

open "pressure-probe" for append as #4
print #4, "", stopsign_width, uniform_velocity, stopsign_MOI, torsional_damping,
torsional_stiffness, tiempo, probe_Cp(j)
close #4

'-----

```

```

'
' C.8 Define the streamlines to plot in each of the Z planes.
'
' streamline(plane, streamline index, number of points to plot)
'
' 1. W(z) plane to the z1-plane. Uniform flow about a circular
' cylinder. Flow is in x1-direction.
'
' z1 = [w +/- sqrt(w^2 - 4a^2)] / 2
'
' w(z1) = z1 + a^2 / z1
'
' 2. Z1-plane to the z2-plane. Set the angle of attack to
' the initial value.
'
' z2 = z1 * e^[i * alpha(initial)]
'
' z1 = z2 * e^[-i * alpha(initial)]
'
' 3. Z2-plane to the z3-plane. Rotate the system through the
' deflection angle.
'
' z3 = z2 * e^[ i * deflection_angle]
'
' z2 = z3 * e^[-i * deflection_angle]
'
' 4. Z3-plane to the z4-plane. Rotate the z3 plane by PI/2 to
' prepare the cylinder to be squished. 4
'
' z4 = z3 * (-i)
'
' z3 = z4 * i
'
' 5. Z4-plane to the z5-plane. Squish the cylinder.
'
' z5 = z4 + a^2 / z4
'
' z4 = [ z5 +/- sqrt(z5^2 - 4a^2) ] / 2
'
' 6. Z5-plane to the z6-plane. Rotate the flow back by the
' angle PI/2 minus the deflection_angle angle..
'
' z6 = z5 * e^[-PI/2 + deflection]*i
'
' z5 = z6 * e^[ PI/2 - deflection]*i
'
'-----
if (S_choice) = 1 then
'-----
' C.8.1 W to Z1-Plane
'-----

for k = 1 to (2 * N_streamlines)
    j = 1

    for phi = phi_min to phi_max step (phi_max - phi_min) / N_points

        w.x      = phi:                w.y      = psi(k)
        value.x  = 4 * a * a:          value.y  = 0 '
        argument = C_sqr( C_sub( C_mult(w, w), value) )
        value.x  = 2:                value.y  = 0

        streamline(1, k, j) = C_div( C_add( w, argument), value) 'Inviscid component ONLY

    j += 1

```

```

next phi

    j = N_points

for phi = phi_min to phi_max step (phi_max - phi_min) / N_points

    w.x      = phi:          w.y      = psi(k)
    value.x  = 4 * a * a:    value.y  = 0
    argument = C_neg( C_sqr( C_sub( C_mult(w, w), value) ) )
    value.x  = 2:          value.y  = 0
    streamline(1, k, j) = C_div( C_add( w, argument), value)

    j += 1

next phi

next k

'-----
'
' C.8.2 Z1 to Z2-Plane
'-----

angle.x = angle_initial:          angle.y = 0

for k = 1 to 2 * N_streamlines
    for j = 1 to 2 * N_points

        streamline(2, k, j) = C_mult( streamline(1, k, j), C_exp( C_mult(i, angle) ) )

        ' Eliminate the streamlines within the cylinder.

        if (C_abs(streamline(2, k, j)) < a) then
            streamline(2, k, j).x = 0
            streamline(2, k, j).y = 0
        end if

    next j
next k

'-----
'
' C.8.3 Z2 to Z3-Plane
'-----

angle.x = deflection_angle:          angle.y = 0

for k = 1 to 2 * N_streamlines
    for j = 1 to 2 * N_points

        streamline(3, k, j) = C_mult( streamline(2, k, j), C_exp( C_mult(i, angle) ) )

        ' Eliminate the streamlines within the cylinder.

        if (C_abs(streamline(3, k, j)) < a) then
            streamline(3, k, j).x = 0
            streamline(3, k, j).y = 0
        end if

    next j
next k

'-----
'
' C.8.4 Z3 to Z4-Plane
'-----

for k = 1 to 2 * N_streamlines
    for j = 1 to 2 * N_points
        streamline(4, k, j) = C_mult( C_neg(i), streamline(3, k, j) )
    next j

```

```

next k

'-----
'
' C.8.5 Z4 to Z5-Plane
'-----

value.x = a * a:                value.y = 0

for k = 1 to 2 * N_streamlines
  for j = 1 to 2 * N_points
    streamline(5, k, j) = C_add( streamline(4, k, j), C_div( value, streamline(4, k, j)))
  next j
next k

'-----
'
' C.8.6 Z5 to Z6-Plane
'-----

angle.x = PI/2 - deflection_angle:  angle.y = 0

for k = 1 to 2 * N_streamlines
  for j = 1 to 2 * N_points
    streamline(6, k, j) = C_mult( streamline(5, k, j), C_exp(C_mult(i, angle)) )
  next j
next k

end if

'-----
'
' C.10 Plot out a screen.
'-----

'-----
'
' C.10.1 Set up the graphics screens to flip them.
'-----

ScreenSet 2,1
cls
view (0,0) - (700,700), color_background,0 'Upper left (0,0), Lower right (700,700) pixels

'-----
'
' C.10.1.1 Plot Cp.
'-----

if (plane_choice = 0) then

if (plot_choice = 1) then
'Plot out a data graph.
  window (-1.1, -1.1) - (1.1, 1.1)

'Plot the axes.

  line (-1, 0) - (1, 0), 0
  line ( 0, -1.2) - (0, 1.2), 0

'Label the axes.

  draw string (-0.95, 0.90), "** Pressure Coefficient, Cp **", 0
  draw string (-0.95, 0.85), "Deflection Angle (deg) = " & deflection_angle * 180 / PI, 0

  max_abcissa = 2.0 * a
  min_ordinate = FRONT_Cp(1)

```

```

max_ordinate = FRONT_Cp(1)

for j = 1 to N_points

    if (min_ordinate > FRONT_Cp(j)) then min_ordinate = FRONT_Cp(j)
    if (max_ordinate < FRONT_Cp(j)) then max_ordinate = FRONT_Cp(j)

    if (min_ordinate > BACK_Cp(j)) then min_ordinate = BACK_Cp(j)
    if (max_ordinate < BACK_Cp(j)) then max_ordinate = BACK_Cp(j)

    if (max_ordinate < abs(min_ordinate) ) then max_ordinate = abs(min_ordinate)

next j

max_ordinate = 15
min_ordinate = 15

draw string (0.95, 0.10), "r (m)", 0:      draw string (0.05, 1.05), "Cp", 0
draw string (-1.0, -0.07), "-" & a,0:   draw string (1.0, -0.07), "" & a,0
draw string (-0.1, 1.05), "" & max_ordinate,0
draw string (-0.1, -1.05), "-" & min_ordinate,0

draw string (-0.95, 0.80), "Net force x (N) = " & FORCE.x, 0
draw string (-0.95, 0.75), "Net force y (N) = " & FORCE.y, 0
draw string (-0.95, 0.70), "Net moment (N-m) = " & MOMENT, 0
draw string (-0.95, 0.65), "FRONT = Cyan, BACK = Red, TOTAL = Green", 0

for j = 1 to N_points
'Plot pressure distribution on front surface
if FRONT_Cp(j)-FRONT_Cp(j-1) <> 0 and abs(FRONT_Cp(j)-FRONT_Cp(j-1)) < 10 then
x = FRONT(6, j).x / max_abcissa
y = FRONT(6, j).y / max_abcissa
radius = sqr(x * x + y * y)
if (j < N_points/2) then radius = -radius
line (radius, FRONT_Cp(j) / max_ordinate) - (radius, FRONT_Cp(j-1) / max_ordinate), 3
end if
'Plot pressure distribution on back surface
if BACK_Cp(j)-BACK_Cp(j-1) <> 0 and abs(BACK_Cp(j)-BACK_Cp(j-1)) < 10 then
x = BACK(6, j).x / max_abcissa
y = BACK(6, j).y / max_abcissa
radius = sqr(x * x + y * y)
if (j < N_points/2) then radius = -radius
line (radius, -BACK_Cp(j) / max_ordinate) - (radius, -BACK_Cp(j-1) / max_ordinate), 4
end if
'Plot total pressure distribution
if abs(TOTAL_Cp(j)-TOTAL_Cp(j-1)) < 10 then
line (radius, TOTAL_Cp(j) / max_ordinate) - (radius, TOTAL_Cp(j-1) / max_ordinate), 2
end if
next j

if deflection_angle + angle_initial > 0 then
if(F_choice= 1) then circle (- sqrt(F(6).x^2+F(6).y^2)/(2*a),0),a/scale_of_points/4, 3,,,,F
if(R_choice= 1) then circle ( sqrt(R(6).x^2+R(6).y^2)/(2*a),0),a/scale_of_points/4, 4,,,,F
else
if(F_choice= 1) then circle ( sqrt(F(6).x^2+F(6).y^2)/(2*a),0),a/scale_of_points/4, 3,,,,F
if(R_choice= 1) then circle (- sqrt(R(6).x^2+R(6).y^2)/(2*a),0),a/scale_of_points/4, 4,,,,F
end if

if abs(torsional_stress) > torsional_strength then
draw string (-0.95, 0.68), "FAILURE: EXCEEDED TORSIONAL STRENGTH",4
sleep
end if

end if

'-----
'
' C.10.1.2 Plot Forces.
'
'-----

if (plot_choice = 2) then
window (-1.1, -1.1) - (1.1, 1.1) ' plot out a data graph.

line (-1, 0) - (1, 0), 0 ' Plot the axes.

```

```

line ( 0, -1.2) - (0, 1.2), 0

draw string (-0.95, 0.90), "* Forces: dFx/dr, dFy/dr (N/m) *", 0 ' Label the axes.
draw string (-0.95, 0.85), "Deflection Angle (deg) = " & deflection_angle * 180 / PI, 0

max_abscissa = 2.0 * a
min_ordinate = FRONT_dFx_dr(1)
max_ordinate = FRONT_dFx_dr(1)

for j = 1 to N_points

  if (min_ordinate > FRONT_dFx_dr(j)) then min_ordinate = FRONT_dFx_dr(j)
  if (max_ordinate < FRONT_dFx_dr(j)) then max_ordinate = FRONT_dFx_dr(j)

  if (min_ordinate > BACK_dFx_dr(j)) then min_ordinate = BACK_dFx_dr(j)
  if (max_ordinate < BACK_dFx_dr(j)) then max_ordinate = BACK_dFx_dr(j)

  if (max_ordinate < abs(min_ordinate) ) then max_ordinate = abs(min_ordinate)

next j

max_ordinate = 15
min_ordinate = 15

draw string (0.95, 0.10), "r (m)", 0:          draw string (0.05, 1.05), "dF/dr (N/m)", 0
draw string (-1.0, -0.07), "-" & a,0:        draw string (1.0, -0.07), "-" & a,0
draw string (-0.1, 1.05), "" & max_ordinate,0: draw string (-0.1, -1.05), "-" & min_ordinate,0

draw string (-0.95, 0.80), "Net force x (N) = " & FORCE.x, 0
draw string (-0.95, 0.75), "Net force y (N) = " & FORCE.y, 0
draw string (-0.95, 0.70), "Net moment (N-m) = " & MOMENT, 0
draw string (-0.95, 0.65), "FRONT = Cyan, BACK = Red, TOTAL = Green", 0

max_ordinate = 15
min_ordinate = 15

for j = 1 to N_points
  if FRONT_dF_dr(j)-FRONT_dF_dr(j-1) <> 0 and abs(FRONT_dF_dr(j)-FRONT_dF_dr(j-1))<10 then
    x = FRONT(6, j).x / max_abscissa
    y = FRONT(6, j).y / max_abscissa
    radius = sqr(x * x + y * y)
    if (j < N_points/2) then radius = -radius
  line (radius, FRONT_dF_dr(j) / max_ordinate) - (radius, FRONT_dF_dr(j-1) / max_ordinate), 3
  end if
  if BACK_dF_dr(j)-BACK_dF_dr(j-1) <> 0 and abs(BACK_dF_dr(j)-BACK_dF_dr(j-1)) <10 then
    x = BACK(6, j).x / max_abscissa
    y = BACK(6, j).y / max_abscissa
    radius = sqr(x * x + y * y)
    if (j < N_points/2) then radius = -radius
  line (radius, -BACK_dF_dr(j) / max_ordinate) - (radius, -BACK_dF_dr(j-1) / max_ordinate), 4
  end if
  if abs((FRONT_dF_dr(j) - BACK_dF_dr(j))-(FRONT_dF_dr(j-1) - BACK_dF_dr(j-1))) <10 then
    line (radius, (FRONT_dF_dr(j) - BACK_dF_dr(j)) / max_ordinate) - (radius, (FRONT_dF_dr(j-1) -
BACK_dF_dr(j-1)) / max_ordinate), 2
  end if
next j

if deflection_angle + angle_initial > 0 then
  if(F_choice=1) then circle (-sqr(F(6).x^2+F(6).y^2)/max_abscissa,0),a/scale_of_points/4, 3,,,,F
  if(R_choice=1) then circle (sqr(R(6).x^2+R(6).y^2)/max_abscissa,0),a/scale_of_points/4, 4,,,,F
else
  if(F_choice=1) then circle (sqr(F(6).x^2+F(6).y^2)/max_abscissa,0),a/scale_of_points/4, 3,,,,F
  if(R_choice=1) then circle (-sqr(R(6).x^2+R(6).y^2)/max_abscissa,0),a/scale_of_points/4, 4,,,,F
end if

if abs(torsional_stress) > torsional_strength then
  draw string (-0.95, 0.60), "FAILURE: EXCEEDED TORSIONAL STRENGTH",4
  sleep
end if

end if

else

```



```

window (x_min, y_min) - (x_max, y_max)

'-----
'
' C.10.2 Plot the cylinder wall.
'-----

if (plane_choice = 1) then circle (0, 0), a, color_cylinder_wall,,,,F

'-----
'
' C.10.3 Plot the front and back surfaces of the stopsign in each Z plane.
'-----

if(FRONT_choice = 1) then
  for j = 1 to N_points
    circle(FRONT(plane_choice, j).x, FRONT(plane_choice, j).y), a/scale_of_points, 0,,,,F
  next j
end if

if(BACK_choice = 1) then
  for j = 1 to N_points
    circle(BACK(plane_choice, j).x, BACK(plane_choice, j).y), a/scale_of_points, 14,,,,F
  next j
end if

'-----
'
' C.10.4 Plot key points on the surface of the stopsign as small circles.
'-----

if(F_choice = 1) then circle ( F(plane_choice).x, F(plane_choice).y), a/scale_of_points, 3,,,,F
if(R_choice = 1) then circle ( R(plane_choice).x, R(plane_choice).y), a/scale_of_points, 4,,,,F
if(T_choice = 1) then circle ( T(plane_choice).x, T(plane_choice).y), a/scale_of_points, 6,,,,F
if(B_choice = 1) then circle ( B(plane_choice).x, B(plane_choice).y), a/scale_of_points, 5,,,,F
if(LT_choice = 1) then circle ( LT(plane_choice).x, LT(plane_choice).y), a/scale_of_points, 10,,,,F
if(RT_choice = 1) then circle ( RT(plane_choice).x, RT(plane_choice).y), a/scale_of_points, 2,,,,F
if(LRP_choice = 1) then circle (LRP(plane_choice).x, LRP(plane_choice).y), a/scale_of_points, 11,,,,F
if(RRP_choice = 1) then circle (RRP(plane_choice).x, RRP(plane_choice).y), a/scale_of_points, 3,,,,F
'circle (probe_position(plane_choice).x, probe_position(plane_choice).y), a/scale_of_points, 14,,,,F

'-----
'
' C.10.5 Plot axes on the reactor drawing.and label the plane.
'-----

if (axis_choice=1) then
  line (0, y_min) - (0, 0.6*y_max), 0
  line (x_min, 0) - (x_max, 0), 0
end if

draw string (0.95*x_min, 0.98*y_max), plot_label(plane_choice, 1),0
draw string (0.95*x_min, 0.94*y_max), plot_label( 1, 2),0
draw string (0.95*x_min, 0.90*y_max), plot_label( 1, 8),0

draw string (0.95*x_min, 0.86*y_max), plot_label( 1, 3),0
draw string (0.95*x_min, 0.82*y_max), plot_label( 1, 4),0
draw string (0.95*x_min, 0.78*y_max), plot_label( 1, 5),0
draw string (0.95*x_min, 0.74*y_max), plot_label( 1, 6),0
draw string (0.95*x_min, 0.70*y_max), plot_label( 1, 7),0

'-----
'
' C.10.6 Plot the inviscid streamlines.
'-----

if(S_choice = 1) then 'Show all inviscid streamlines
  for k = 1 to 2 * N_streamlines
    for j = 1 to 2 * N_points
pset(streamline(plane_choice, k, j).x, streamline(plane_choice, k, j).y), color_streamlines

```

```

        next j
      next k
    end if

    if(S_choice = 0) then 'Show profile streamline only
      line (LT(plane_choice).x,LT(plane_choice).y) - (RT(plane_choice).x,RT(plane_choice).y), 0
    end if
  '-----
  ' C.10.7 Plot the vortices. (+ counter clockwise, - clockwise)
  '-----

  for j = 1 to N_vortices
    if vortex_strength(j).x > 1e-5 then
      if (V_choice = 1) then
        line (vortex_position(plane_choice,j).x - (N_radii_width_mult/2)*a/scale_of_points,
vortex_position(plane_choice,j).y) - (vortex_position(plane_choice,j).x +
(N_radii_width_mult/2)*a/scale_of_points, vortex_position(plane_choice,j).y), 0          'ccw
        line (vortex_position(plane_choice,j).x, vortex_position(plane_choice,j).y -
(N_radii_width_mult/2)*a/scale_of_points) - (vortex_position(plane_choice,j).x,
vortex_position(plane_choice,j).y + (N_radii_width_mult/2)*a/scale_of_points), 0          'ccw
      else
        circle (vortex_position(plane_choice,j).x, vortex_position(plane_choice,j).y),
2*a/scale_of_points*(N_radii_width_mult/2)*vortex_strength(j).x, 4,,,F
      end if
    elseif vortex_strength(j).x < -1e-5 then
      if (V_choice = 1) then
        line (vortex_position(plane_choice,j).x - (N_radii_width_mult/2)*a/scale_of_points,
vortex_position(plane_choice,j).y) - (vortex_position(plane_choice,j).x +
(N_radii_width_mult/2)*a/scale_of_points, vortex_position(plane_choice,j).y), 0          'cw
      else
        circle (vortex_position(plane_choice,j).x, vortex_position(plane_choice,j).y),
2*a/scale_of_points*(N_radii_width_mult/2)*-vortex_strength(j).x, 1,,,F
      end if
    end if
  next j

  if abs(torsional_stress) > torsional_strength then
    draw string (0.95*x_min, 0.66*y_max), "FAILURE: EXCEEDED TORSIONAL STRENGTH",4
    sleep
  end if

  '-----
  ' C.10.9 Flip the graphics screens.
  '-----

  ScreenSet 1,1
  ScreenSync
  Flip 2,1

end if

next tiempo

end

function C_distance(ZA as complex, ZB as complex) as double
'-----
' Function: C_distance
' Purpose: Compute the distance between two points, ZA and ZB, expressed
'          in complex coordinates.
'-----

dim as double distance

distance = sqr( (ZB.x - ZA.x) * (ZB.x - ZA.x) + (ZB.y - ZA.y) * (ZB.y - ZA.y) )

```

```

    return(distance)
end function

function real_color(mycolor as integer) as string
'-----
'
'   Function:  real_color
'
'   Purpose:  This function returns the actual color that will be
'             plotted based on the color number (integer), for a
'             16 color palette.
'-----

    if mycolor = 0 then return("black")
    if mycolor = 1 then return("blue")
    if mycolor = 2 then return("green")
    if mycolor = 3 then return("cyan")
    if mycolor = 4 then return("red")
    if mycolor = 5 then return("magenta")
    if mycolor = 6 then return("brown")
    if mycolor = 7 then return("white")
    if mycolor = 8 then return("gray")
    if mycolor = 9 then return("bright blue")
    if mycolor = 10 then return("bright green")
    if mycolor = 11 then return("bright cyan")
    if mycolor = 12 then return("bright red")
    if mycolor = 13 then return("pink")
    if mycolor = 14 then return("yellow")
    if mycolor = 15 then return("bright white")

end function

function flipflop(x as integer) as integer
'-----
'
'   Function:  flipflop
'
'   Purpose:  Flip an integer of 1 to a 0 and a 0 to a 1.
'-----

    dim as integer value

    if(x = 0) then
        value = 1
    else
        value = 0
    end if

    return(value)

end function

```

## A.2 Complex Functions

The BI file that gives the program the capability to analyze complex numbers is provided below.

```
Type complex
  dim as double x, y
End Type

dim shared as complex i
  i.x = 0
  i.y = 1

dim shared as complex unity
  unity.x = 1
  unity.y = 0

Declare function sinh(x as double) as double
Declare function cosh(x as double) as double
Declare function tanh(x as double) as double
Declare function coth(x as double) as double
Declare function sech(x as double) as double
Declare function csch(x as double) as double
Declare function cot(x as double) as double

Declare function C_print(z as complex) as double
Declare function C_add(z1 as complex, z2 as complex) as complex
Declare function C_sub(z1 as complex, z2 as complex) as complex
Declare function C_mult(z1 as complex, z2 as complex) as complex
Declare function C_div(z1 as complex, z2 as complex) as complex
Declare function C_conj(z as complex) as complex
Declare function C_exp(z as complex) as complex
Declare function C_ln(z as complex) as complex
Declare function C_real(z as complex) as double
Declare function C_imag(z as complex) as double
Declare function C_abs(z as complex) as double
Declare function C_arg(z as complex) as double
Declare function C_reciprocal(z as complex) as complex
Declare function C_pow_real(z as complex, d as double) as complex
Declare function C_pow(z as complex, q as complex) as complex
Declare function C_sqrt(z as complex) as complex
Declare function C_neg(z as complex) as complex

Declare function C_sin(z as complex) as complex
Declare function C_cos(z as complex) as complex
Declare function C_tan(z as complex) as complex
Declare function C_sec(z as complex) as complex
Declare function C_csc(z as complex) as complex
Declare function C_cot(z as complex) as complex

Declare function C_asin(z as complex) as complex
Declare function C_acos(z as complex) as complex
Declare function C_atan(z as complex) as complex
Declare function C_acot(z as complex) as complex

Declare function C_sinh(z as complex) as complex
Declare function C_cosh(z as complex) as complex
Declare function C_tanh(z as complex) as complex
Declare function C_csch(z as complex) as complex
Declare function C_sech(z as complex) as complex
Declare function C_coth(z as complex) as complex

Declare function C_asinh(z as complex) as complex
Declare function C_acosh(z as complex) as complex
Declare function C_atanh(z as complex) as complex
Declare function C_acoth(z as complex) as complex

function sinh(x as double) as double
  return((exp(x) - exp(-x))/2)
end function

function cosh(x as double) as double
  return((exp(x) + exp(-x))/2)
```

```

end function

function tanh(x as double) as double
    return((exp(x) - exp(-x)) / (exp(x) + exp(-x)))
end function

function coth(x as double) as double
    return((exp(x) + exp(-x)) / (exp(x) - exp(-x)))
end function

function sech(x as double) as double
    return(2/(exp(x) + exp(-x)))
end function

function csch(x as double) as double
    return(2/(exp(x) - exp(-x)))
end function

function cot(x as double) as double
    dim as double value

    value = 1 / tan(x)
    return(x)
end function

function C_print(z as complex) as double
    if (z.y >= 0.0) then
        print z.x;" + ";z.y;"i"
    else
        print z.x;" - ";-z.y;"i"
    end if
end function

function C_add(z1 as complex, z2 as complex) as complex
    dim value as complex
    value.x = z1.x + z2.x
    value.y = z1.y + z2.y
    return (value)
end function

function C_mult(z1 as complex, z2 as complex) as complex
    dim value as complex
    value.x = z1.x * z2.x - z1.y * z2.y
    value.y = z1.x * z2.y + z2.x * z1.y
    return (value)
end function

function C_sub(z1 as complex, z2 as complex) as complex
    dim value as complex
    value.x = z1.x - z2.x
    value.y = z1.y - z2.y
    return (value)
end function

function C_div(z1 as complex, z2 as complex) as complex
    dim value as complex
    dim denominator as double
    denominator = z2.x * z2.x + z2.y * z2.y
    value.x = (z1.x * z2.x + z1.y * z2.y) / denominator
    value.y = (z2.x * z1.y - z1.x * z2.y) / denominator
    return (value)
end function

```

```

function C_conj(z as complex) as complex
  dim value as complex
  value.x = z.x
  value.y = -z.y
  return (value)
end function

function C_exp(z as complex) as complex
  dim value as complex
  value.x = exp(z.x) * cos(z.y)
  value.y = exp(z.x) * sin(z.y)
  return (value)
end function

function C_ln(z as complex) as complex
  dim value as complex
  value.x = 0.5 * log(z.x * z.x + z.y * z.y)
  value.y = atan2(z.y, z.x)
  return (value)
end function

function C_imag(z as complex) as double
  return (z.y)
end function

function C_real(z as complex) as double
  return (z.x)
end function

function C_abs(z as complex) as double
  return (sqr(z.x * z.x + z.y * z.y))
end function

function C_arg(z as complex) as double
  dim as double value, PI = 3.14159265358
  value = atan2(z.y, z.x)

  if(value < 0) then value += 2 * PI

  return (value)
end function

function C_reciprocal(z as complex) as complex
  dim value as complex
  value.x = z.x / (z.x * z.x + z.y * z.y)
  value.y = -z.y / (z.x * z.x + z.y * z.y)
  return (value)
end function

function C_pow_real(z as complex, d as double) as complex
  dim value as complex
  dim r as double
  r = (z.x * z.x + z.y * z.y)^(d/2)
  value.x = r * cos(d * atan2(z.y, z.x))
  value.y = r * sin(d * atan2(z.y, z.x))
  return (value)
end function

```

```

function C_pow(z as complex, q as complex) as complex
  dim as double j, k, m, argument
  dim as complex u, value

  u = C_pow_real(z, q.x)
  m = exp(-q.y * atan2(z.y, z.x))

  argument = (q.y/2) * log(z.x * z.x + z.y * z.y)
  j = cos(argument)
  k = sin(argument)

  value.x = m * (u.x * j - u.y * k)
  value.y = m * (u.y * j + u.x * k)

  return(value)
end function

```

```

function C_sqr(z as complex) as complex
  dim as double m, a, argument
  dim as complex value

  a = 1/2
  m = (z.x * z.x + z.y * z.y)^(a/2)
  argument = a * atan2(z.y, z.x)

  value.x = m * cos(argument)
  value.y = m * sin(argument)

  return(value)
end function

```

```

function C_neg(z as complex) as complex
  dim as complex value

  value.x = - z.x
  value.y = - z.y

  return(value)
end function

```

```

function C_sin(z as complex) as complex
  dim as complex value

  value.x = sin(z.x) * cosh(z.y)
  value.y = cos(z.x) * sinh(z.y)

  return(value)
end function

```

```

function C_cos(z as complex) as complex
  dim as complex value

  value.x = cos(z.x) * cosh(z.y)
  value.y = -sin(z.x) * sinh(z.y)

  return(value)
end function

```

```

function C_tan(z as complex) as complex
  dim as complex value

```

```

dim as double d

d      = cos(2 * z.x) + cosh(2 * z.y)
value.x = sin(2 * z.x) / d
value.y = sinh(2 * z.y) / d

return(value)
end function

function C_sec(z as complex) as complex
dim as complex value
dim as double d1, d2, d

d1      = cos(z.x) * cos(z.x) * cosh(z.y) * cosh(z.y)
d2      = sin(z.x) * sin(z.x) * sinh(z.y) * sinh(z.y)
d       = d1 + d2

value.x = (cos(z.x) * cosh(z.y)) / d
value.y = (sin(z.x) * sinh(z.y)) / d

return(value)
end function

function C_csc(z as complex) as complex
dim as complex value
dim as double d1, d2, d

d1      = sin(z.x) * sin(z.x) * cosh(z.y) * cosh(z.y)
d2      = cos(z.x) * cos(z.x) * sinh(z.y) * sinh(z.y)
d       = d1 + d2

value.x = (sin(z.x) * cosh(z.y)) / d
value.y = -(cos(z.x) * sinh(z.y)) / d

return(value)
end function

function C_cot(z as complex) as complex
dim as complex value
dim as double d

d      = sin(2 * z.x) * sin(2 * z.x) + sinh(2 * z.y) * sinh(2 * z.y)

value.x = sin(2 * z.x) * (cos(2 * z.x) + cosh(2 * z.y)) / d
value.y = -sinh(2 * z.y) * (cos(2 * z.x) + cosh(2 * z.y)) / d

return(value)
end function

function C_asin(z as complex) as complex
dim as complex value, i, neg_i, unity

i.x      = 0
i.y      = 1

neg_i.x  = 0
neg_i.y  = -1

unity.x  = 1
unity.y  = 0

value = C_mult( C_ln( C_add( C_sqr( C_sub( unity, C_mult(z,z) )), C_mult(i, z) )), neg_i)

```



```

    return(value)
end function

function C_acos(z as complex) as complex
    dim as Complex prefix, value
    dim as double PI

    PI = 3.14159265358

    prefix.x = PI/2
    prefix.y = 0

    value = C_sub(prefix, C_asin(z))
    return(value)
end function

function C_atan(z as complex) as complex
    dim as Complex i, iz, one, two, value

    i.x = 0
    i.y = 1

    one.x = 1
    one.y = 0

    two.x = 2
    two.y = 0

    iz = C_mult(i, z)

    value = C_ln( C_div( C_sub( one, iz), C_add( one, iz) ) )
    value = C_mult( C_div(i, two), value)

    return(value)
end function

function C_acot(z as complex) as complex
    dim as double real, PI
    dim as complex value, prefix, neg_prefix

    PI = 3.14159265358

    prefix.x = PI/2
    prefix.y = 0

    neg_prefix.x = -PI/2
    neg_prefix.y = 0

    real = C_real(z)

    if (real >= 0.0) then
        value = C_sub( prefix, C_atan(z))
    else
        value = C_sub(neg_prefix, C_atan(z))
    end if

    return(value)
end function

function C_sinh(z as complex) as complex
    dim value as complex

```

```

    value.x = cos(z.y) * sinh(z.x)
    value.y = sin(z.y) * cosh(z.x)

    return(value)
end function

function C_cosh(z as complex) as complex
    dim value as complex

    value.x = cos(z.y) * cosh(z.x)
    value.y = sin(z.y) * sinh(z.x)

    return(value)
end function

function C_tanh(z as complex) as complex
    dim value as complex
    dim as double d1, d2, d

    d1 = cos(z.y) * cos(z.y) * cosh(z.x) * cosh(z.x)
    d2 = sin(z.y) * sin(z.y) * sinh(z.x) * sinh(z.x)
    d = d1 + d2

    value.x = (sinh(z.x) * cosh(z.x)) / d
    value.y = ( sin(z.y) * cos(z.y)) / d

    return(value)
end function

function C_csch(z as complex) as complex
    dim as complex unity

    unity.x = 1
    unity.y = 0

    return( C_div( unity, C_sinh(z)) )
end function

function C_sech(z as complex) as complex
    dim as complex unity

    unity.x = 1
    unity.y = 0

    return(C_div( unity, C_cosh(z)) )
end function

function C_coth(z as complex) as complex
    dim as complex unity

    unity.x = 1
    unity.y = 0

    return(C_div( unity, C_tanh(z)) )
end function

function C_asinh(z as complex) as complex
    dim as complex value, unity

    unity.x = 1

```

```

unity.y = 0

value = C_ln( C_add( C_sqr( C_add( C_mult( z, z), unity)), z))
return(value)
end function

function C_acosh(z as complex) as complex
dim as complex value, neg_unity

neg_unity.x = -1
neg_unity.y = 0

value = C_ln( C_add( C_sqr( C_add( C_mult( z, z), neg_unity)), z))
return(value)
end function

function C_atanh(z as complex) as complex
dim as Complex value, unity, two

unity.x = 1
unity.y = 0

two.x = 2
two.y = 0

value = C_div( C_ln( C_div( C_add( unity, z), C_sub( unity, z))), two)
return(value)
end function

function C_acoth(z as complex) as complex
dim as complex value, unity, two

unity.x = 1
unity.y = 0

two.x = 2
two.y = 0

value = C_div( C_ln( C_div( C_add( z, unity), C_sub( z, unity))), two)
return(value)
end function

```

## APPENDIX B

### SAMPLE INPUT FILE

Below is the input text file that was used to simulate a 4 m wide flat plate with a torsional spring constant of 50 Nm, damping coefficient of 1 Nms, and mass moment of inertia of 5 kgm<sup>2</sup> in a horizontal flow of 1 m/s. This was the reference case form which all parametric studies were performed.

```
-----  
'  
' Input File for Program STOPSIGN24.bas  
'  
' Color code: 0 = black  
'              1 = blue  
'              2 = green  
'              3 = cyan  
'              4 = red  
'              5 = magenta  
'              6 = brown  
'              7 = white  
'              8 = gray  
'              9 = bright blue  
'             10 = bright green  
'             11 = bright cyan  
'             12 = bright red  
'             13 = pink  
'             14 = yellow  
'             15 = bright white  
'  
' Assumptions:  
' 1. Stopsign does not deflect, except torsionally, about the  
'    z-axis.  
'  
-----  
  
A. Fluid Flow Specifications  
  
A.1 Uniform flow velocity (m/s): 1  
A.2 Initial angle of attack (degrees): 0  
  
B. Ambient Fluid Properties  
  
B.1 Fluid (0 = air, 1 = water, 2 = unity): 0  
B.2 Ambient pressure (Pa): 101325  
B.3 Ambient temperature (K): 293  
  
C. Stopsign Mechanical Properties  
  
C.1 Torsional strength (Pa): 2e10  
C.2 Moment of inertia (kgm^2): 5  
C.3 Torsional damping (Nms/rad): 1  
C.4 Torsional stiffness (Nm/rad): 50  
C.5 Stopsign width (W) in (m): 4
```

D. Plotting Specifications

D.1	Number of streamlines to plot:	40
D.2	Number of radii in height for square plot:	5
D.3	Number of radii in width to square plot:	5
D.4	Number of radii in full plot	2
D.5	Color of streamlines:	1
D.6	Color of cylinder wall:	15
D.7	Color of the background:	15
D.8	Number of points per streamline to plot:	500

E. Simulation Time

E.1	Maximum time for simulation (s):	5000
E.2	Time increment for simulation (s):	0.125
E.3	Print to screen (1) or output.dat (2):	1
E.4	Choice of output plane to initially view (1 to 6):	6
E.5	Choice to show vortices (1 for yes):	1

F. Vortex Properties

F.1	Number of vortices zeroed to trigger asymmetry:	75
F.2	Vortex dissipation factor:	0.999
F.3	Number of radii after which to zero out distant vortices:	100
F.4	Amalgamation distance:	a/20
F.5	Initial guess for release point:	a*0.25

## APPENDIX C

### SAMPLE OUTPUT FILE

The three output files that are appended to with each loop of the program are combined into the following sample output file. The values from left to right represent: plate width in meters, uniform velocity in m/s, mass moment of inertia of the plate in  $\text{kgm}^2$ , damping coefficient in Nms, torsional spring constant in Nm, time elapsed in seconds, deflection angle in radians, the net force in the x direction in Newtons, the net force in the y direction in Newtons, the net moment in Nm, and the coefficient of pressure. This output file corresponds to the input file provided in Appendix B.

```
4 1 5 1 50 0 0 0 0 -0.0251 0.0279
4 1 5 1 50 0.125 0.0001 -0.5739 0.0000 1.0061 0.0345
4 1 5 1 50 0.25 -0.0029 -0.5290 -0.0015 1.2950 0.0388
4 1 5 1 50 0.375 -0.0094 -0.4582 -0.0043 1.1723 0.0421
4 1 5 1 50 0.5 -0.0178 -0.0774 -0.0014 1.2521 0.0449
4 1 5 1 50 0.625 -0.0271 0.1009 0.0027 1.1989 0.0474
4 1 5 1 50 0.75 -0.0357 0.3482 0.0124 1.0613 0.0497
4 1 5 1 50 0.875 -0.0419 0.5464 0.0229 1.0009 0.0519
4 1 5 1 50 1 -0.0446 0.7586 0.0339 0.9712 0.0539
4 1 5 1 50 1.125 -0.0434 0.8136 0.0354 0.8250 0.0558
4 1 5 1 50 1.25 -0.0382 1.2133 0.0463 1.0426 0.0576
4 1 5 1 50 1.375 -0.0304 1.4380 0.0437 1.1491 0.0593
4 1 5 1 50 1.5 -0.0216 1.7121 0.0371 1.2141 0.0610
4 1 5 1 50 1.625 -0.0135 1.7662 0.0239 1.1353 0.0626
4 1 5 1 50 1.75 -0.0070 1.9600 0.0138 1.2136 0.0643
4 1 5 1 50 1.875 -0.0033 2.2826 0.0075 1.4415 0.0661
4 1 5 1 50 2 -0.0036 2.3821 0.0085 1.4716 0.0679
4 1 5 1 50 2.125 -0.0078 2.4266 0.0188 1.4715 0.0698
4 1 5 1 50 2.25 -0.0152 2.2457 0.0340 1.2664 0.0718
4 1 5 1 50 2.375 -0.0239 2.2169 0.0530 1.2247 0.0738
4 1 5 1 50 2.5 -0.0326 2.3399 0.0762 1.3784 0.0759
4 1 5 1 50 2.625 -0.0402 2.3133 0.0931 1.3411 0.0780
```

4 1 5 1 50 2.75 -0.0457 2.3262 0.1063 1.2991 0.0801  
4 1 5 1 50 2.875 -0.0480 2.1635 0.1039 1.1163 0.0821  
4 1 5 1 50 3 -0.0463 2.2262 0.1032 1.1468 0.0842  
4 1 5 1 50 3.125 -0.0411 2.3275 0.0958 1.2049 0.0863  
4 1 5 1 50 3.25 -0.0335 2.4554 0.0822 1.2824 0.0884  
4 1 5 1 50 3.375 -0.0248 2.7777 0.0689 1.5533 0.0905  
4 1 5 1 50 3.5 -0.0173 2.8974 0.0502 1.6306 0.0927  
4 1 5 1 50 3.625 -0.0123 2.9814 0.0368 1.6875 0.0949  
4 1 5 1 50 3.75 -0.0107 2.8252 0.0303 1.5235 0.0973  
4 1 5 1 50 3.875 -0.0122 2.8179 0.0343 1.5247 0.0998  
4 1 5 1 50 4 -0.0164 2.7709 0.0454 1.5018 0.1024  
4 1 5 1 50 4.125 -0.0226 2.6956 0.0609 1.4598 0.1050  
4 1 5 1 50 4.25 -0.0296 2.6067 0.0772 1.4075 0.1078  
4 1 5 1 50 4.375 -0.0363 2.7142 0.0985 1.5486 0.1106  
4 1 5 1 50 4.5 -0.0419 2.6418 0.1109 1.5044 0.1135  
4 1 5 1 50 4.625 -0.0457 2.0260 0.0926 1.1630 0.1196  
4 1 5 1 50 4.75 -0.0459 -2.1593 -0.0992 -1.8403 0.1365  
4 1 5 1 50 4.875 -0.0335 -2.6249 -0.0880 -1.9596 0.1463  
4 1 5 1 50 5 -0.0103 -2.6341 -0.0272 -1.6668 0.1546  
4 1 5 1 50 5.125 0.0189 -2.4704 0.0467 -1.3869 0.1622  
4 1 5 1 50 5.25 0.0488 -2.1447 0.1048 -0.9066 0.1693  
4 1 5 1 50 5.375 0.0733 -1.9553 0.1436 -0.6033 0.1760  
4 1 5 1 50 5.5 0.0879 -1.6762 0.1476 -0.2586 0.1825  
4 1 5 1 50 5.625 0.0894 -1.5431 0.1384 -0.1139 0.1887  
4 1 5 1 50 5.75 0.0777 -1.3251 0.1032 0.0601 0.1948  
4 1 5 1 50 5.875 0.0542 -1.2327 0.0669 0.0614 0.2008  
4 1 5 1 50 6 0.0229 -1.0372 0.0237 0.1504 0.2066  
4 1 5 1 50 6.125 -0.0117 -0.9104 -0.0106 0.1476 0.2124  
4 1 5 1 50 6.25 -0.0440 -0.7908 -0.0348 0.1569 0.2182  
4 1 5 1 50 6.375 -0.0694 -0.7243 -0.0503 0.1498 0.2238  
4 1 5 1 50 6.5 -0.0840 -0.6396 -0.0538 0.1866 0.2296  
4 1 5 1 50 6.625 -0.0860 -0.6004 -0.0518 0.2184 0.2354  
4 1 5 1 50 6.75 -0.0755 -0.5932 -0.0449 0.3024 0.2413  
4 1 5 1 50 6.875 -0.0547 -0.5746 -0.0315 0.2823 0.2474  
4 1 5 1 50 7 -0.0269 -0.5624 -0.0152 0.3313 0.2535  
4 1 5 1 50 7.125 0.0033 -0.5168 0.0017 0.4128 0.2597

4 1 5 1 50 7.25 0.0310 -0.4526 0.0140 0.5110 0.2660  
4 1 5 1 50 7.375 0.0517 -0.3747 0.0194 0.6097 0.2722  
4 1 5 1 50 7.5 0.0622 -0.3006 0.0187 0.7006 0.2784  
4 1 5 1 50 7.625 0.0609 -0.2404 0.0146 0.7740 0.2846  
4 1 5 1 50 7.75 0.0479 -0.2259 0.0108 0.7961 0.2907  
4 1 5 1 50 7.875 0.0255 -0.1379 0.0035 0.8791 0.2969  
4 1 5 1 50 8 -0.0029 -0.1400 -0.0004 0.9280 0.3030  
4 1 5 1 50 8.125 -0.0330 -0.1605 -0.0053 0.8863 0.3092  
4 1 5 1 50 8.25 -0.0600 -0.1573 -0.0095 0.8651 0.3154  
4 1 5 1 50 8.375 -0.0799 -0.1469 -0.0118 0.8578 0.3217  
4 1 5 1 50 8.5 -0.0897 -0.1687 -0.0152 0.7708 0.3280  
4 1 5 1 50 8.625 -0.0880 -0.2861 -0.0252 0.6769 0.3347  
4 1 5 1 50 8.75 -0.0749 -0.3770 -0.0283 0.6356 0.3414  
4 1 5 1 50 8.875 -0.0527 -0.4118 -0.0217 0.6665 0.3484  
4 1 5 1 50 9 -0.0250 -0.4517 -0.0113 0.6973 0.3555  
4 1 5 1 50 9.125 0.0037 -0.4642 0.0017 0.7539 0.3627  
4 1 5 1 50 9.25 0.0288 -0.4824 0.0139 0.8469 0.3700  
4 1 5 1 50 9.375 0.0464 -0.4691 0.0218 0.9221 0.3775  
4 1 5 1 50 9.5 0.0536 -0.4645 0.0249 0.9970 0.3851  
4 1 5 1 50 9.625 0.0494 -0.4352 0.0215 1.0327 0.3929  
4 1 5 1 50 9.75 0.0347 -0.4643 0.0161 1.0350 0.4008  
4 1 5 1 50 9.875 0.0119 -0.4177 0.0050 1.0849 0.4089  
4 1 5 1 50 10 -0.0155 -0.4150 -0.0064 1.0550 0.4173  
4 1 5 1 50 10.125 -0.0431 -0.4189 -0.0181 0.9948 0.4258  
4 1 5 1 50 10.25 -0.0664 -0.4295 -0.0286 0.9096 0.4345  
4 1 5 1 50 10.375 -0.0819 -0.4575 -0.0375 0.8070 0.4436  
4 1 5 1 50 10.5 -0.0869 -0.5040 -0.0439 0.6983 0.4528  
4 1 5 1 50 10.625 -0.0807 -0.5630 -0.0455 0.5956 0.4624  
4 1 5 1 50 10.75 -0.0642 -0.6164 -0.0396 0.5132 0.4722  
4 1 5 1 50 10.875 -0.0398 -0.6516 -0.0260 0.4572 0.4822  
4 1 5 1 50 11 -0.0114 -0.6937 -0.0079 0.3923 0.4925  
4 1 5 1 50 11.125 0.0169 -0.7040 0.0119 0.4046 0.5030  
4 1 5 1 50 11.25 0.0407 -0.6704 0.0273 0.4032 0.5137  
4 1 5 1 50 11.375 0.0565 -0.6372 0.0360 0.4127 0.5245  
4 1 5 1 50 11.5 0.0620 -0.5466 0.0339 0.4232 0.5356  
4 1 5 1 50 11.625 0.0566 -0.5252 0.0298 0.3974 0.5470



4 1 5 1 50 11.75 0.0416 -0.4921 0.0205 0.3350 0.5586  
4 1 5 1 50 11.875 0.0195 -0.4357 0.0085 0.2272 0.5703  
4 1 5 1 50 12 -0.0057 -0.3871 -0.0022 0.1129 0.5823  
4 1 5 1 50 12.125 -0.0297 -0.2928 -0.0087 -0.0982 0.5944  
4 1 5 1 50 12.25 -0.0483 -0.1256 -0.0061 -0.2911 0.6065  
4 1 5 1 50 12.375 -0.0583 0.0265 0.0015 -0.5350 0.6188  
4 1 5 1 50 12.5 -0.0574 0.1911 0.0110 -0.7977 0.6311  
4 1 5 1 50 12.625 -0.0454 0.3483 0.0158 -1.1055 0.6433  
4 1 5 1 50 12.75 -0.0234 0.5923 0.0139 -1.4346 0.6554  
4 1 5 1 50 12.875 0.0060 0.8819 -0.0053 -1.6959 0.6673  
4 1 5 1 50 13 0.0389 1.1944 -0.0465 -1.9919 0.6789  
4 1 5 1 50 13.125 0.0712 1.5355 -0.1095 -2.2912 0.6901  
4 1 5 1 50 13.25 0.0989 1.8688 -0.1853 -2.5739 0.7010  
4 1 5 1 50 13.375 0.1186 2.1292 -0.2537 -2.7892 0.7113  
4 1 5 1 50 13.5 0.1283 2.2589 -0.2913 -2.8811 0.7211  
4 1 5 1 50 13.625 0.1269 2.2148 -0.2827 -2.8031 0.7303  
4 1 5 1 50 13.75 0.1148 2.1641 -0.2496 -2.7110 0.7388  
4 1 5 1 50 13.875 0.0938 2.1730 -0.2044 -2.6515 0.7465  
4 1 5 1 50 14 0.0671 2.1974 -0.1476 -2.5518 0.7532  
4 1 5 1 50 14.125 0.0385 2.2690 -0.0874 -2.4642 0.7589  
4 1 5 1 50 14.25 0.0123 2.4139 -0.0297 -2.4159 0.7636  
4 1 5 1 50 14.375 -0.0077 2.5843 0.0200 -2.3976 0.7671  
4 1 5 1 50 14.5 -0.0188 2.8766 0.0542 -2.5679 0.7695  
4 1 5 1 50 14.625 -0.0190 3.1621 0.0600 -2.8787 0.7706  
4 1 5 1 50 14.75 -0.0074 3.4608 0.0256 -3.2672 0.7706  
4 1 5 1 50 14.875 0.0150 3.7770 -0.0565 -3.6108 0.7693  
4 1 5 1 50 15 0.0455 4.0515 -0.1844 -3.8637 0.7669  
4 1 5 1 50 15.125 0.0801 4.3564 -0.3499 -4.0918 0.7632  
4 1 5 1 50 15.25 0.1142 4.6466 -0.5329 -4.2877 0.7585  
4 1 5 1 50 15.375 0.1431 4.8143 -0.6936 -4.3900 0.7528  
4 1 5 1 50 15.5 0.1628 4.8697 -0.8001 -4.4121 0.7462  
4 1 5 1 50 15.625 0.1707 4.7066 -0.8115 -4.1768 0.7388  
4 1 5 1 50 15.75 0.1652 4.2602 -0.7101 -3.5426 0.7306  
4 1 5 1 50 15.875 0.1453 4.0828 -0.5976 -3.0697 0.7218  
4 1 5 1 50 16 0.1132 3.9597 -0.4502 -2.5222 0.7125  
4 1 5 1 50 16.125 0.0723 3.9694 -0.2874 -2.0275 0.7026

4 1 5 1 50 16.25 0.0275 4.0889 -0.1126 -1.4378 0.6923  
4 1 5 1 50 16.375 -0.0160 4.3314 0.0691 -1.0146 0.6816  
4 1 5 1 50 16.5 -0.0528 4.7155 0.2494 -0.7727 0.6707  
4 1 5 1 50 16.625 -0.0784 5.1178 0.4022 -0.6395 0.6594  
4 1 5 1 50 16.75 -0.0895 5.4914 0.4927 -0.6912 0.6479  
4 1 5 1 50 16.875 -0.0845 5.8827 0.4984 -1.0665 0.6362  
4 1 5 1 50 17 -0.0635 6.2699 0.3989 -1.6072 0.6245  
4 1 5 1 50 17.125 -0.0285 6.7330 0.1918 -2.2020 0.6125  
4 1 5 1 50 17.25 0.0168 6.9805 -0.1171 -2.8396 0.6008  
4 1 5 1 50 17.375 0.0670 7.2082 -0.4838 -3.5121 0.5891  
4 1 5 1 50 17.5 0.1165 7.6665 -0.8975 -4.0124 0.5772  
4 1 5 1 50 17.625 0.1593 7.5983 -1.2209 -4.3341 0.5659  
4 1 5 1 50 17.75 0.1900 7.7489 -1.4901 -4.4880 0.5545  
4 1 5 1 50 17.875 0.2046 7.2270 -1.4997 -4.0779 0.5440  
4 1 5 1 50 18 0.2001 6.3664 -1.2914 -3.2084 0.5341  
4 1 5 1 50 18.125 0.1750 5.7570 -1.0181 -2.4011 0.5246  
4 1 5 1 50 18.25 0.1312 5.2696 -0.6953 -1.5087 0.5156  
4 1 5 1 50 18.375 0.0730 5.0516 -0.3695 -0.6569 0.5069  
4 1 5 1 50 18.5 0.0071 5.2045 -0.0371 0.0367 0.4986  
4 1 5 1 50 18.625 -0.0583 5.5639 0.3250 0.7175 0.4905  
4 1 5 1 50 18.75 -0.1155 6.1909 0.7184 1.2137 0.4828  
4 1 5 1 50 18.875 -0.1574 6.8127 1.0812 1.5200 0.4755  
4 1 5 1 50 19 -0.1789 7.2332 1.3078 1.3475 0.4686  
4 1 5 1 50 19.125 -0.1767 7.2982 1.3031 0.4147 0.4622  
4 1 5 1 50 19.25 -0.1489 7.1692 1.0753 -0.5943 0.4564  
4 1 5 1 50 19.375 -0.0972 7.0597 0.6886 -1.6595 0.4510  
4 1 5 1 50 19.5 -0.0270 7.4330 0.2006 -2.9933 0.4459  
4 1 5 1 50 19.625 0.0548 8.0524 -0.4418 -4.4198 0.4409  
4 1 5 1 50 19.75 0.1397 7.9366 -1.1162 -5.1475 0.4362  
4 1 5 1 50 19.875 0.2170 7.7324 -1.7044 -6.1694 0.4325  
4 1 5 1 50 20 0.2780 -0.1301 0.0371 1.2183 0.4293  
4 1 5 1 50 20.125 0.2915 -44.0608 13.2224 46.9992 0.4264  
4 1 5 1 50 20.25 0.1170 4.8541 -0.5704 -4.5217 0.4270  
4 1 5 1 50 20.375 -0.0574 3.2573 0.1871 -3.4885 0.4266  
4 1 5 1 50 20.5 -0.2081 2.5643 0.5414 -3.0240 0.4255  
4 1 5 1 50 20.625 -0.3142 2.3245 0.7554 -2.2615 0.4242

4 1 5 1 50 20.75 -0.3629 1.8263 0.6935 -2.1045 0.4231  
4 1 5 1 50 20.875 -0.3487 1.5530 0.5646 -2.1479 0.4224  
4 1 5 1 50 21 -0.2751 2.9954 0.8456 -4.0773 0.4222  
4 1 5 1 50 21.125 -0.1490 2.4696 0.3707 -3.2726 0.4223  
4 1 5 1 50 21.25 0.0068 2.9054 -0.0196 -3.6552 0.4222  
4 1 5 1 50 21.375 0.1688 43.2862 -7.3780 -43.9681 0.4211  
4 1 5 1 50 21.5 0.4353 3.8621 -1.7959 -3.9198 0.4185  
4 1 5 1 50 21.625 0.6408 3.4623 -2.5821 -4.0429 0.4160  
4 1 5 1 50 21.75 0.7560 3.4613 -3.2633 -4.4892 0.4139  
4 1 5 1 50 21.875 0.7668 3.2033 -3.0861 -4.1880 0.4139  
4 1 5 1 50 22 0.6732 1.9059 -1.5199 -2.1780 0.4168  
4 1 5 1 50 22.125 0.4859 0.0017 -0.0009 0.1846 0.4207  
4 1 5 1 50 22.25 0.2286 -1.2537 0.2917 0.8430 0.4229  
4 1 5 1 50 22.375 -0.0599 -1.0124 -0.0607 0.4593 0.4225  
4 1 5 1 50 22.5 -0.3336 -0.3872 -0.1342 0.1224 0.4193  
4 1 5 1 50 22.625 -0.5502 -0.2376 -0.1457 -0.2909 0.4126  
4 1 5 1 50 22.75 -0.6767 -0.2735 -0.2197 -0.4875 0.4021  
4 1 5 1 50 22.875 -0.6955 -0.3317 -0.2768 -0.4771 0.3878  
4 1 5 1 50 23 -0.6063 -0.7185 -0.4983 -0.6447 0.3717  
4 1 5 1 50 23.125 -0.4250 -1.1102 -0.5024 -0.6160 0.3560  
4 1 5 1 50 23.25 -0.1814 -1.0503 -0.1926 -0.5772 0.3446  
4 1 5 1 50 23.375 0.0857 0.5263 -0.0452 -1.2010 0.3426  
4 1 5 1 50 23.5 0.3368 1.9325 -0.6767 -1.3293 0.3454  
4 1 5 1 50 23.625 0.5346 1.8103 -1.0718 -0.3952 0.3416  
4 1 5 1 50 23.75 0.6472 1.5465 -1.1688 -0.6037 0.3053  
4 1 5 1 50 23.875 0.6603 1.1775 -0.9143 -0.5203 0.1241  
4 1 5 1 50 24 0.5739 0.3635 -0.2350 0.1185 -0.6410  
4 1 5 1 50 24.125 0.4019 -0.9386 0.3989 0.8836 -0.1637  
4 1 5 1 50 24.25 0.1700 -1.8379 0.3156 0.7185 0.3177  
4 1 5 1 50 24.375 -0.0842 -1.4647 -0.1237 0.0359 0.4526  
4 1 5 1 50 24.5 -0.3196 -0.3801 -0.1258 0.0138 0.5114  
4 1 5 1 50 24.625 -0.5005 -0.2310 -0.1263 -0.5253 0.5498  
4 1 5 1 50 24.75 -0.5991 -0.3307 -0.2258 -0.9030 0.5842  
4 1 5 1 50 24.875 -0.6013 -0.5568 -0.3819 -1.1365 0.6240  
4 1 5 1 50 25 -0.5082 -1.0232 -0.5700 -1.2952 0.6812  
4 1 5 1 50 25.125 -0.3360 -1.4241 -0.4974 -1.2787 0.7785

4 1 5 1 50 25.25 -0.1129 -1.1387 -0.1291 -0.7797 0.9320  
4 1 5 1 50 25.375 0.1244 33.7674 -4.2209 -34.5693 0.9414  
4 1 5 1 50 25.5 0.4423 1.3189 -0.6246 0.0393 0.8591  
4 1 5 1 50 25.625 0.6849 1.2392 -1.0122 1.5348 0.8372  
4 1 5 1 50 25.75 0.8125 1.6220 -1.7125 1.7194 0.8204  
4 1 5 1 50 25.875 0.8079 0.3898 -0.4078 2.3355 0.7876  
4 1 5 1 50 26 0.6732 0.6658 -0.5309 0.8136 0.7475  
4 1 5 1 50 26.125 0.4366 -0.0781 0.0364 -0.1085 0.7043  
4 1 5 1 50 26.25 0.1396 -0.8712 0.1224 0.0464 0.6478  
4 1 5 1 50 26.375 -0.1716 -0.0795 -0.0138 0.0641 0.5656  
4 1 5 1 50 26.5 -0.4493 -0.2205 -0.1063 -1.0782 0.4590  
4 1 5 1 50 26.625 -0.6484 -0.3213 -0.2434 -1.6129 0.4074  
4 1 5 1 50 26.75 -0.7388 -0.4028 -0.3669 -1.7325 0.4923  
4 1 5 1 50 26.875 -0.7092 -0.5774 -0.4955 -1.8073 0.6166  
4 1 5 1 50 27 -0.5667 -0.9555 -0.6080 -1.7732 0.7034  
4 1 5 1 50 27.125 -0.3358 -1.4764 -0.5153 -1.5651 0.7543  
4 1 5 1 50 27.25 -0.0547 -0.9921 -0.0543 -1.1100 0.7837  
4 1 5 1 50 27.375 0.2314 0.3264 -0.0769 -0.5609 0.7979  
4 1 5 1 50 27.5 0.4769 0.8430 -0.4356 -0.3233 0.8060  
4 1 5 1 50 27.625 0.6447 1.1482 -0.8633 -0.4007 0.8090  
4 1 5 1 50 27.75 0.7113 1.0655 -0.9183 0.0697 0.8114  
4 1 5 1 50 27.875 0.6677 0.7117 -0.5613 -0.0551 0.8205  
4 1 5 1 50 28 0.5236 0.0686 -0.0396 -0.1133 0.8302  
4 1 5 1 50 28.125 0.3034 -0.9104 0.2851 -0.1228 0.8244  
4 1 5 1 50 28.25 0.0428 -1.2071 0.0517 -0.2950 0.8060  
4 1 5 1 50 28.375 -0.2171 -0.6944 -0.1532 -0.4752 0.7868  
4 1 5 1 50 28.5 -0.4361 -0.4050 -0.1888 -0.9119 0.7699  
4 1 5 1 50 28.625 -0.5805 -0.4073 -0.2671 -1.1123 0.7542  
4 1 5 1 50 28.75 -0.6295 -0.5846 -0.4258 -1.2399 0.7388  
4 1 5 1 50 28.875 -0.5776 -0.9903 -0.6454 -1.4283 0.7233  
4 1 5 1 50 29 -0.4345 -1.5852 -0.7357 -1.5647 0.7074  
4 1 5 1 50 29.125 -0.2239 -2.0465 -0.4661 -1.2994 0.6905  
4 1 5 1 50 29.25 0.0196 -1.4865 0.0292 -0.5901 0.6720  
4 1 5 1 50 29.375 0.2560 45.4331 -11.8939 -47.2135 0.6527  
4 1 5 1 50 29.5 0.5916 11.7767 -7.9127 -12.1033 0.6325  
4 1 5 1 50 29.625 0.8657 1.2558 -1.4756 1.5457 0.6122

## APPENDIX D

### SAMPLE BATCH FILE

Below is the batch file used to run different cases of varying torsional spring constant. Several input files were created (see Appendix B) with everything held constant except for the  $\kappa$  values. They were run in the following order by executing the file batchfile.bat.

```
        COPY 4-1-5-1-10-INPUT.TXT INPUT-STOPSIGN23.TXT /Y
STOPSIGN23 > 4-1-5-1-10-OUTPUT
RENAME DATA 4-1-5-1-10-DATA

        COPY 4-1-5-1-12-INPUT.TXT INPUT-STOPSIGN23.TXT /Y
STOPSIGN23 > 4-1-5-1-12-OUTPUT
RENAME DATA 4-1-5-1-12-DATA

        COPY 4-1-5-1-25-INPUT.TXT INPUT-STOPSIGN23.TXT /Y
STOPSIGN23 > 4-1-5-1-25-OUTPUT
RENAME DATA 4-1-5-1-25-DATA

        COPY 4-1-5-1-37-INPUT.TXT INPUT-STOPSIGN23.TXT /Y
STOPSIGN23 > 4-1-5-1-37-OUTPUT
RENAME DATA 4-1-5-1-37-DATA

        COPY 4-1-5-1-50-INPUT.TXT INPUT-STOPSIGN23.TXT /Y
STOPSIGN23 > 4-1-5-1-50-OUTPUT
RENAME DATA 4-1-5-1-50-DATA

        COPY 4-1-5-1-75-INPUT.TXT INPUT-STOPSIGN23.TXT /Y
STOPSIGN23 > 4-1-5-1-75-OUTPUT
RENAME DATA 4-1-5-1-75-DATA

        COPY 4-1-5-1-100-INPUT.TXT INPUT-STOPSIGN23.TXT /Y
STOPSIGN23 > 4-1-5-1-100-OUTPUT
RENAME DATA 4-1-5-1-100-DATA

        COPY 4-1-5-1-125-INPUT.TXT INPUT-STOPSIGN23.TXT /Y
STOPSIGN23 > 4-1-5-1-125-OUTPUT
RENAME DATA 4-1-5-1-125-DATA

        COPY 4-1-5-1-150-INPUT.TXT INPUT-STOPSIGN23.TXT /Y
STOPSIGN23 > 4-1-5-1-150-OUTPUT
RENAME DATA 4-1-5-1-150-DATA

        COPY 4-1-5-1-175-INPUT.TXT INPUT-STOPSIGN23.TXT /Y
STOPSIGN23 > 4-1-5-1-175-OUTPUT
RENAME DATA 4-1-5-1-175-DATA

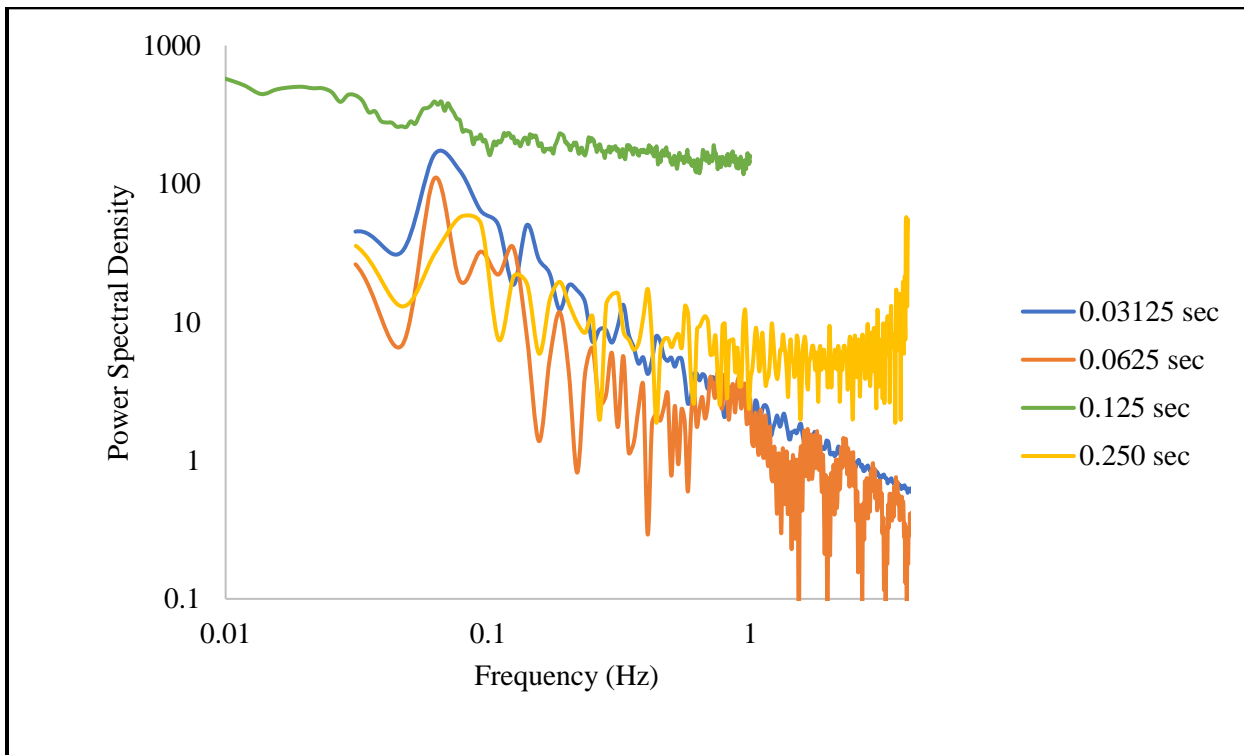
        COPY 4-1-5-1-200-INPUT.TXT INPUT-STOPSIGN23.TXT /Y
STOPSIGN23 > 4-1-5-1-200-OUTPUT
RENAME DATA 4-1-5-1-200-DATA
```

## APPENDIX E

### FFT AUTOSPECTRA

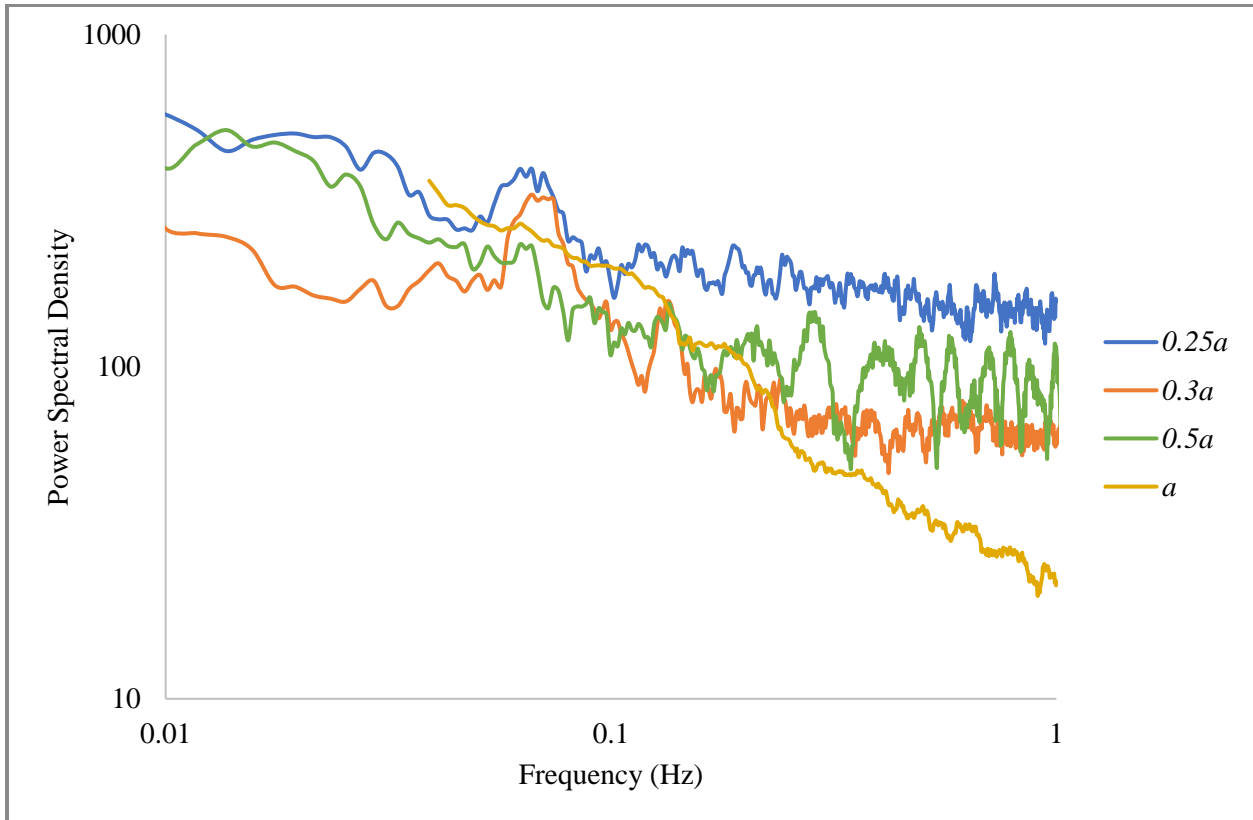
In Section 3.2, stationary plates are compared with varying parameters to validate the minimal impact of those parameters on the flow field. This was determined by measuring the Strouhal frequencies of the wake using the pressure probe tool.

Table 6 compares the Strouhal frequencies for various time steps. The autospectra used to obtain those numerical values are shown in the figure below. These curves all have peaks at approximately the same frequency.



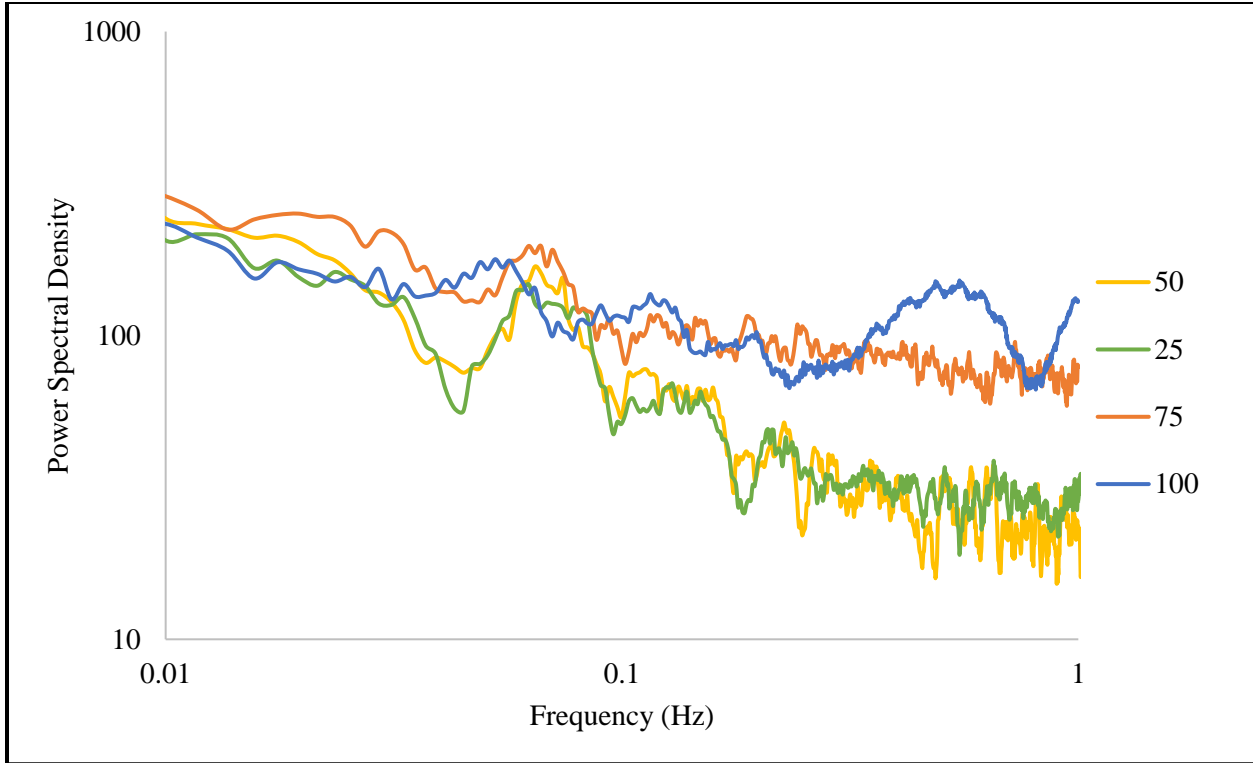
**Figure 33 FFT: Time step comparison**

In Table 7, the  $z_6$ -planes are shown for several cases using different distances of offset for the nascent vortices and the approximate Strouhal frequencies of the wake are given. These values for frequency are found from the peaks of the following autospectra, all of which have peaks at approximately the same frequency.



**Figure 34 FFT: Offset distance of nascent vortices comparison**

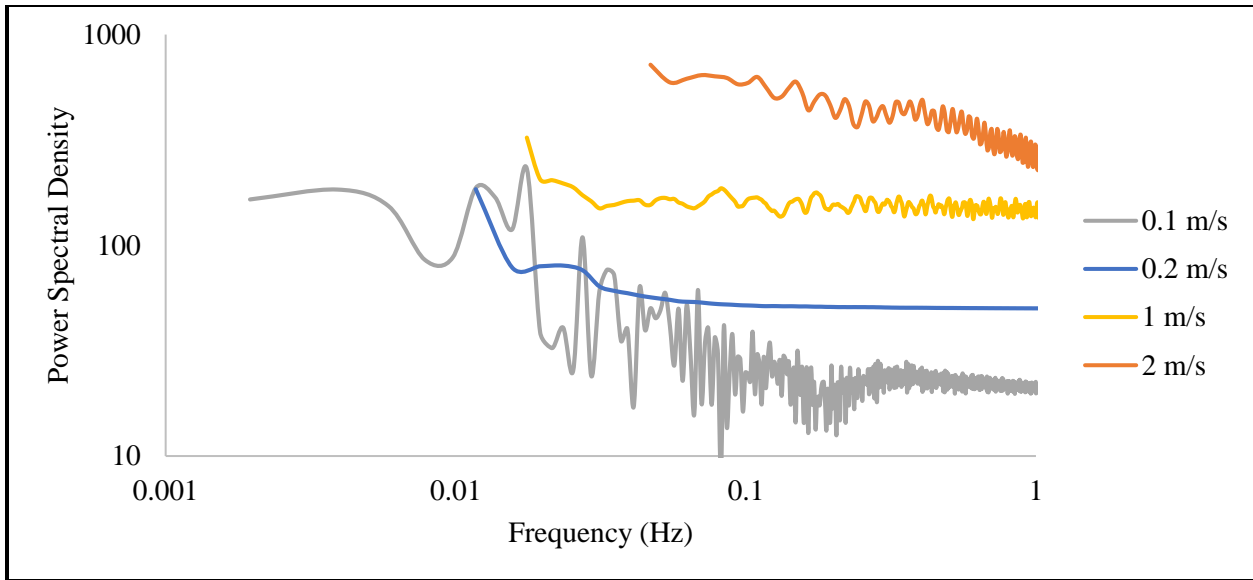
In Table 8, the  $z_6$ -planes are shown for several cases using different numbers of time steps in which the left tip vortices are eliminated to trigger an asymmetric wake. Approximate Strouhal frequencies of each wake are also given. These values for frequency are found from the peaks of the following autospectra, all of which have peaks at approximately the same frequency.



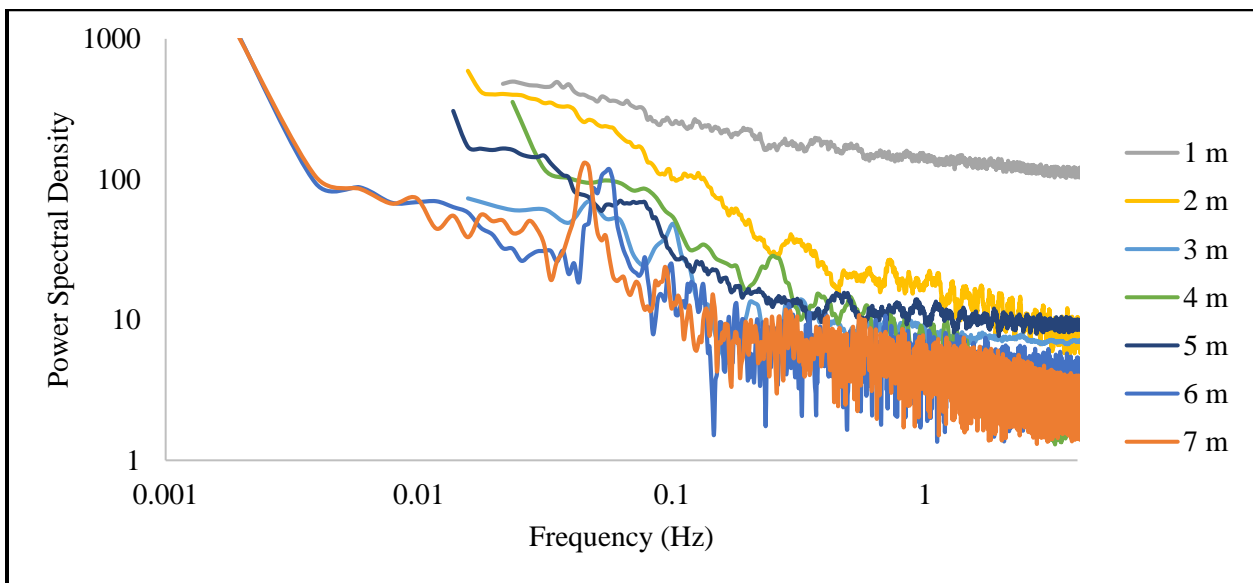
**Figure 35 FFT: Number of time steps to trigger offset comparison**

In Table 9, the theoretical and computed Strouhal numbers of several wakes are given. The DVM values for Strouhal number are found from the peaks of the following autospectra of the deflection angle data from the program. For the velocity variation, the higher velocities result in peaks at higher frequencies indicating higher Strouhal numbers. For plate width variation, a wider plate resulted in a wake with a lower frequency and a lower Strouhal number. These trends correspond with the right half of the theoretical plot of Reynolds number and Strouhal number provided in Figure 5.



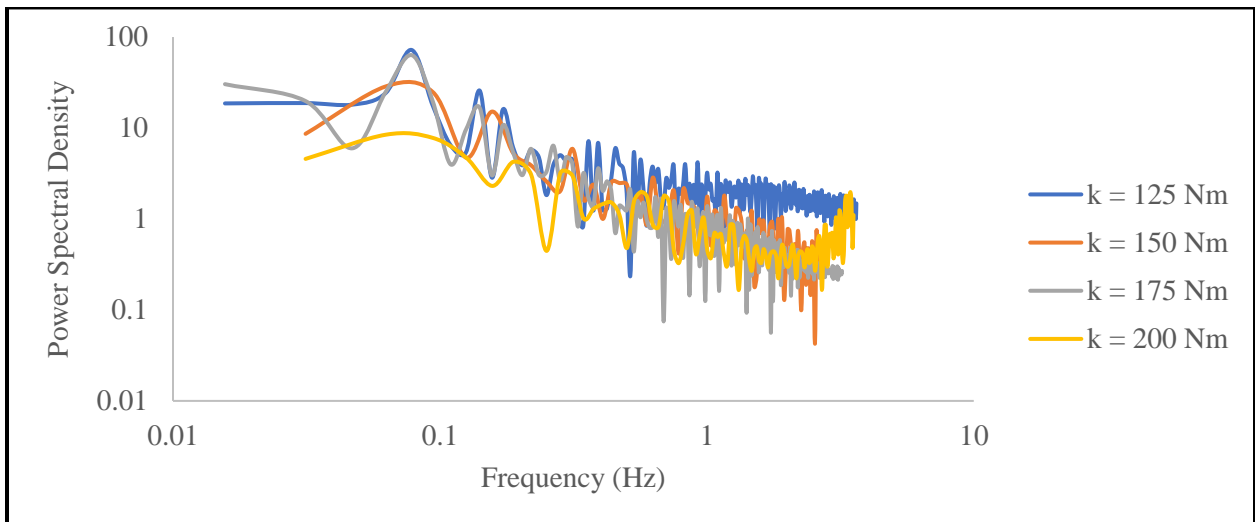
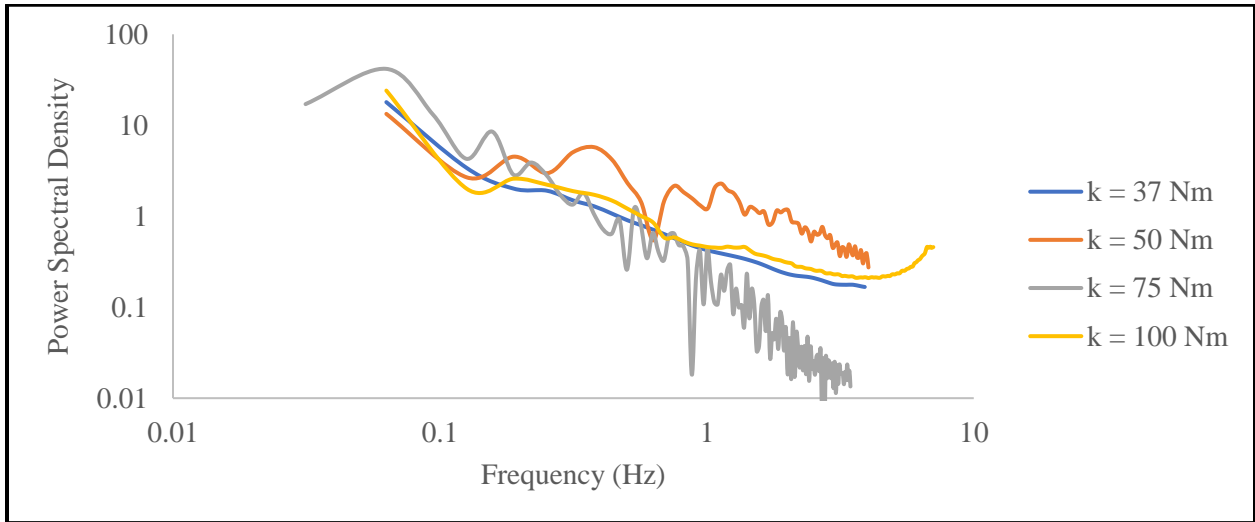
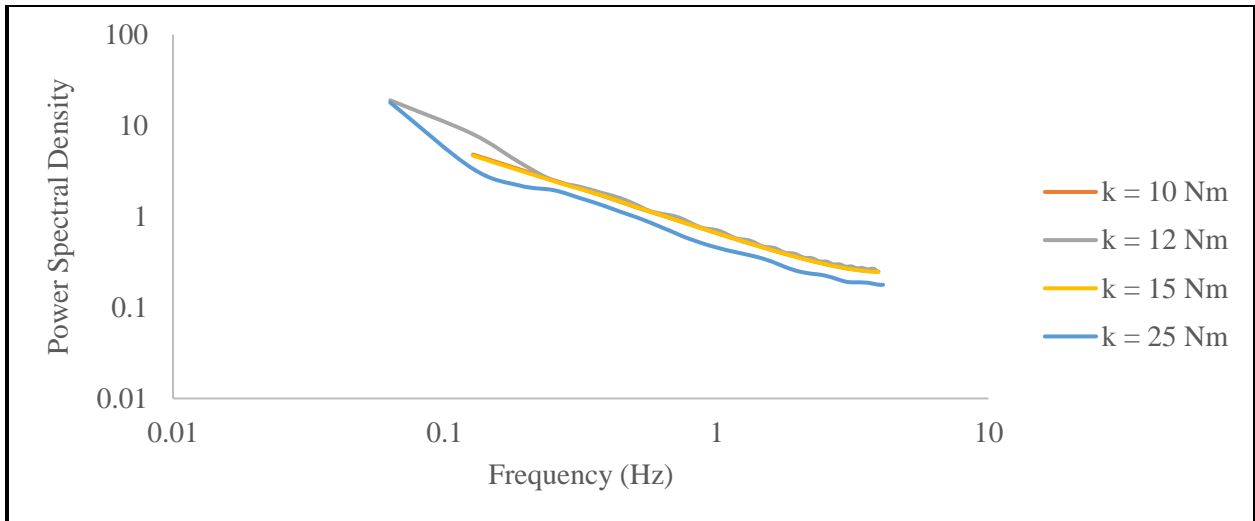


**Figure 36 FFT: Uniform velocity comparison**

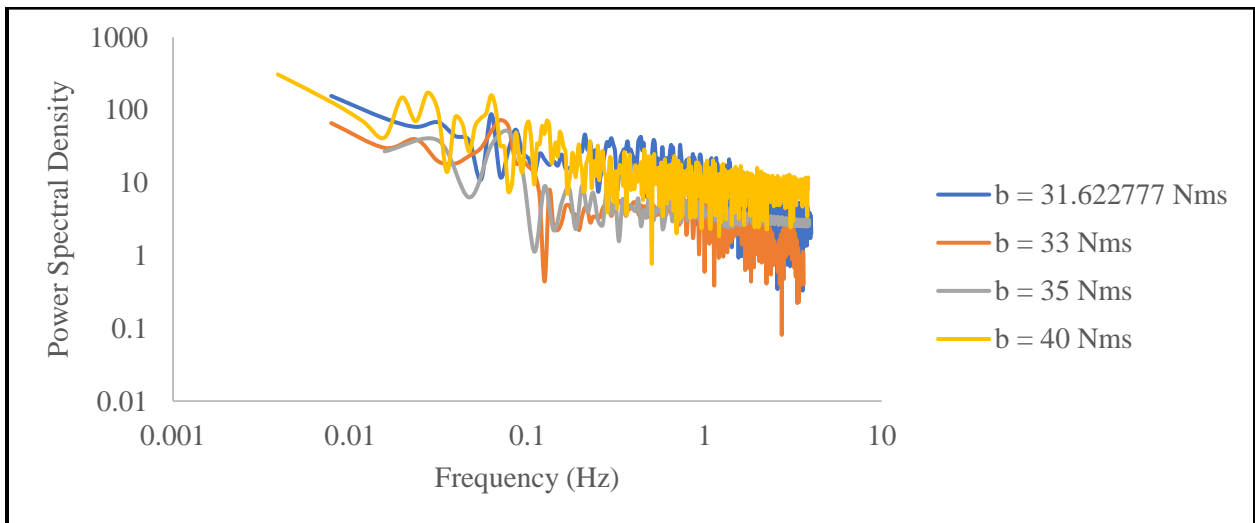
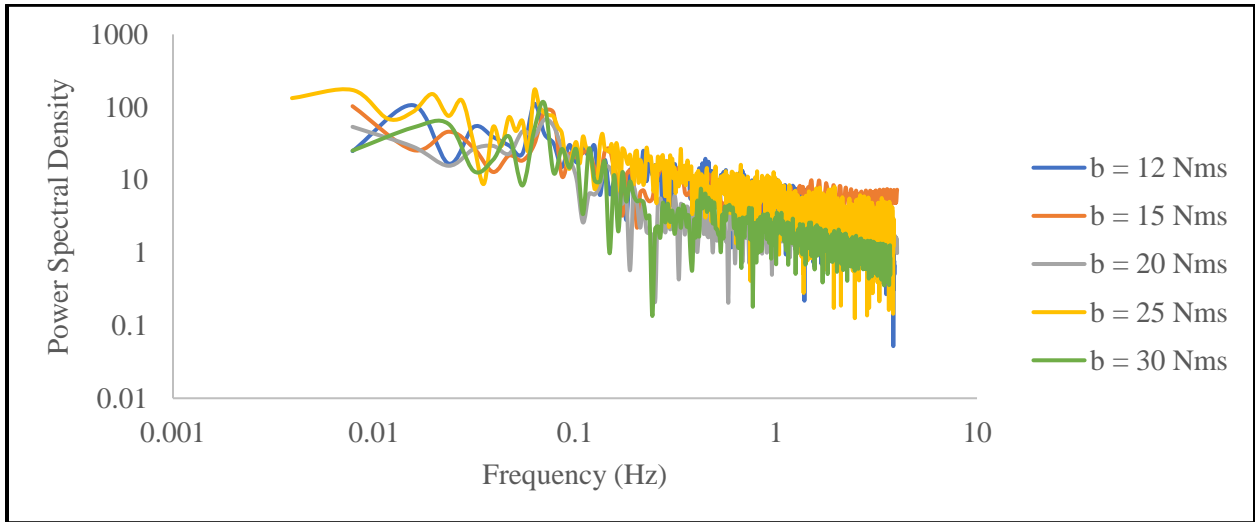
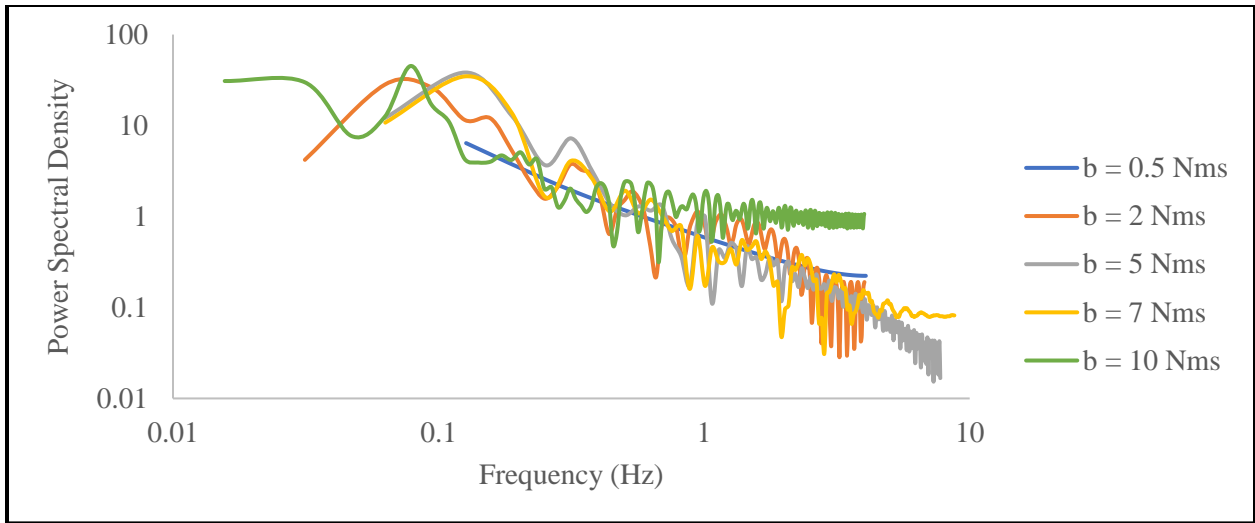


**Figure 37 FFT: Plate width comparison**

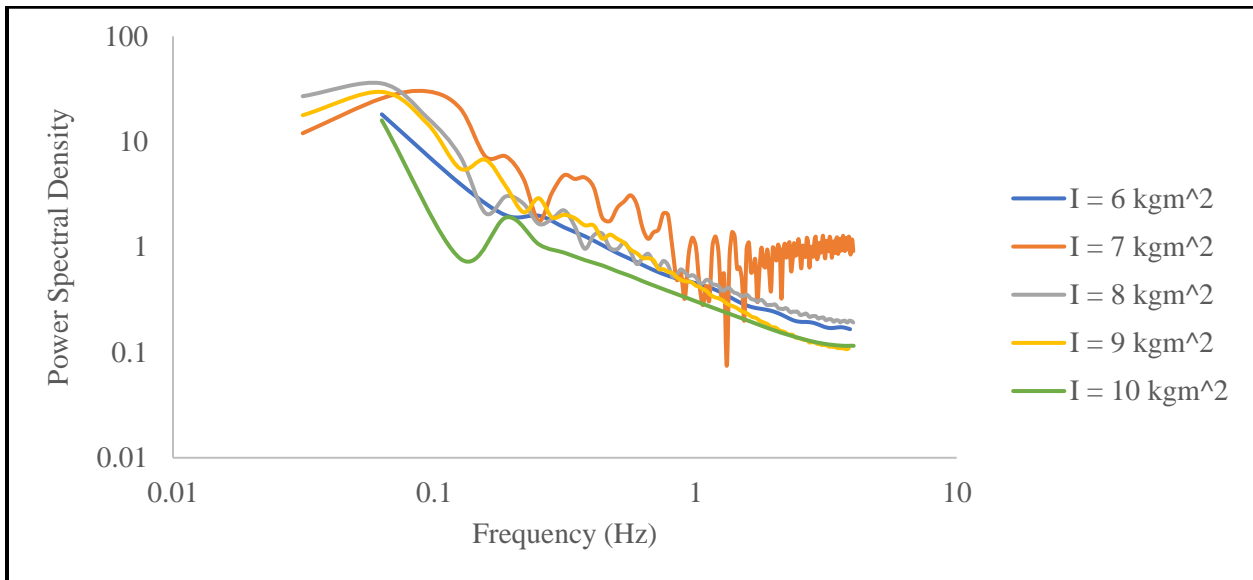
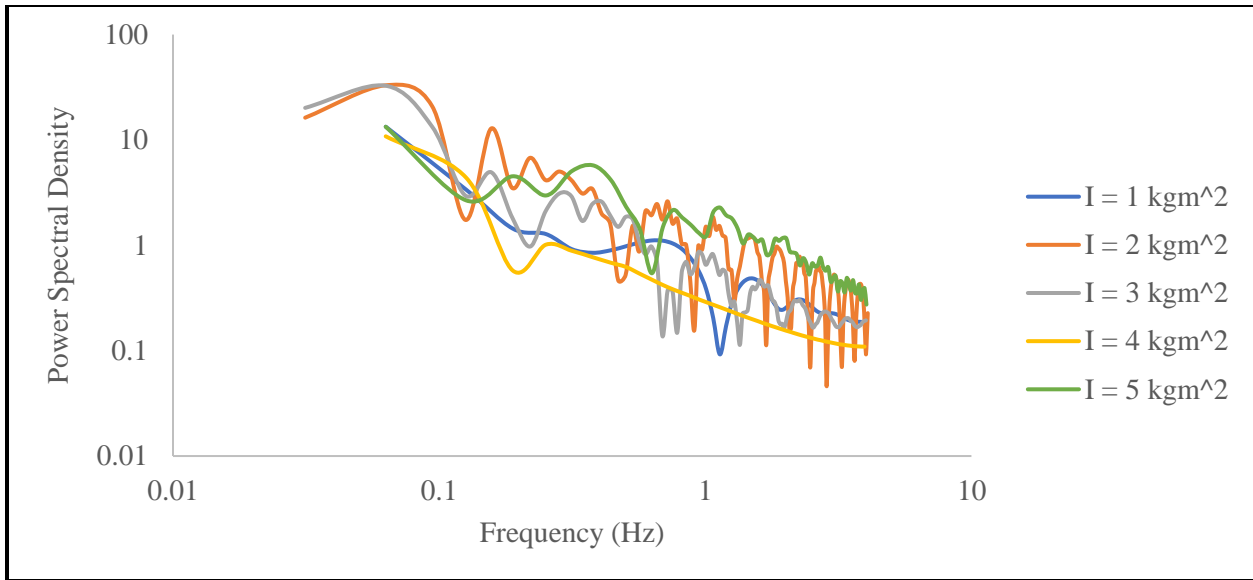
In Section 3.3, varying parameters were tested. For each specimen, the Strouhal number of the wake was approximated using a FFT on the data set compiled from the pressure measurements taken at the probe location during each time step. The FFTs used are shown below. By the proximity of the peaks of the FFT plots, it is clear that variation of the stop sign's mechanical properties does not have a significant effect on the Strouhal number of the wake.



**Figure 38 FFT: Strouhal numbers for varying torsional stiffness**

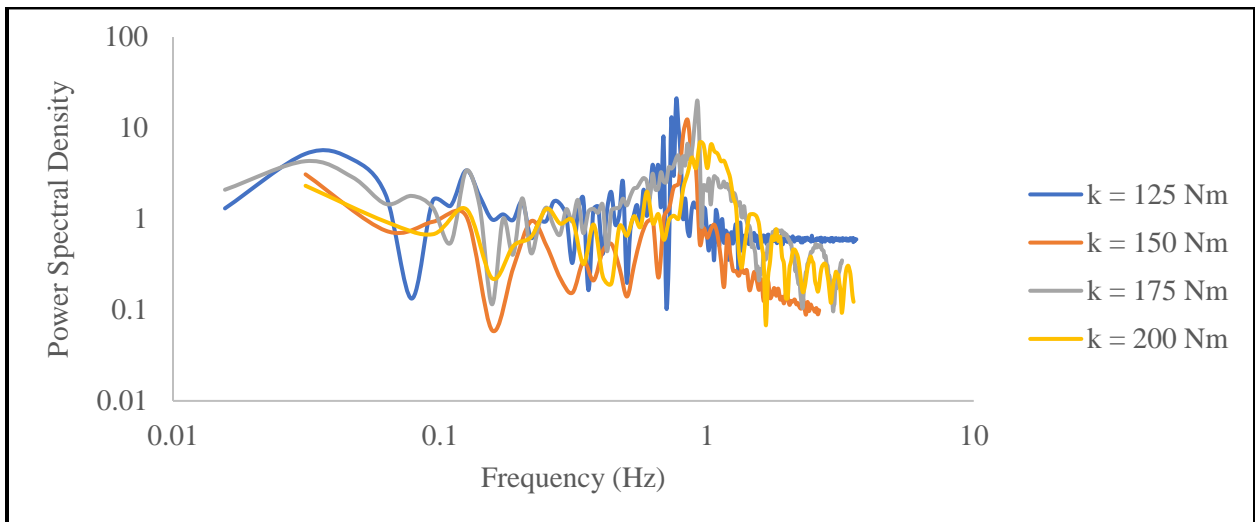
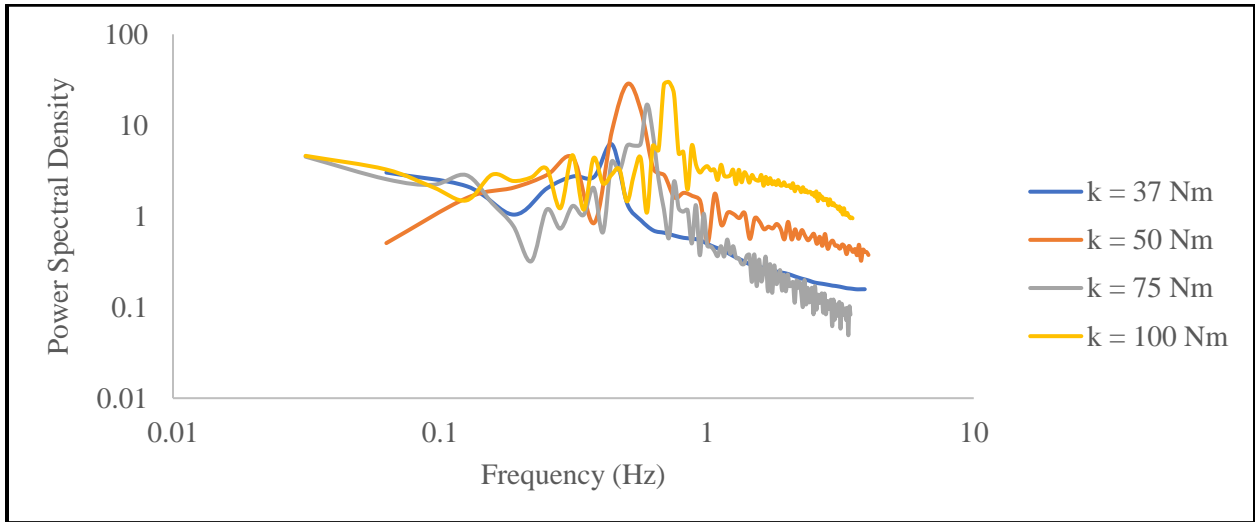
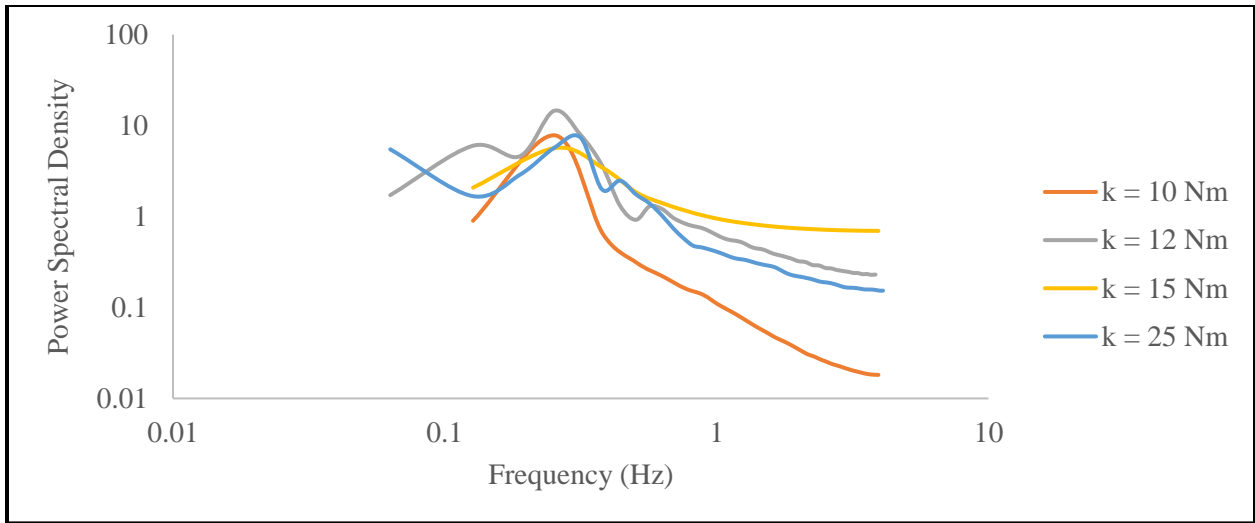


**Figure 39 FFT: Strouhal numbers for varying damping coefficient**

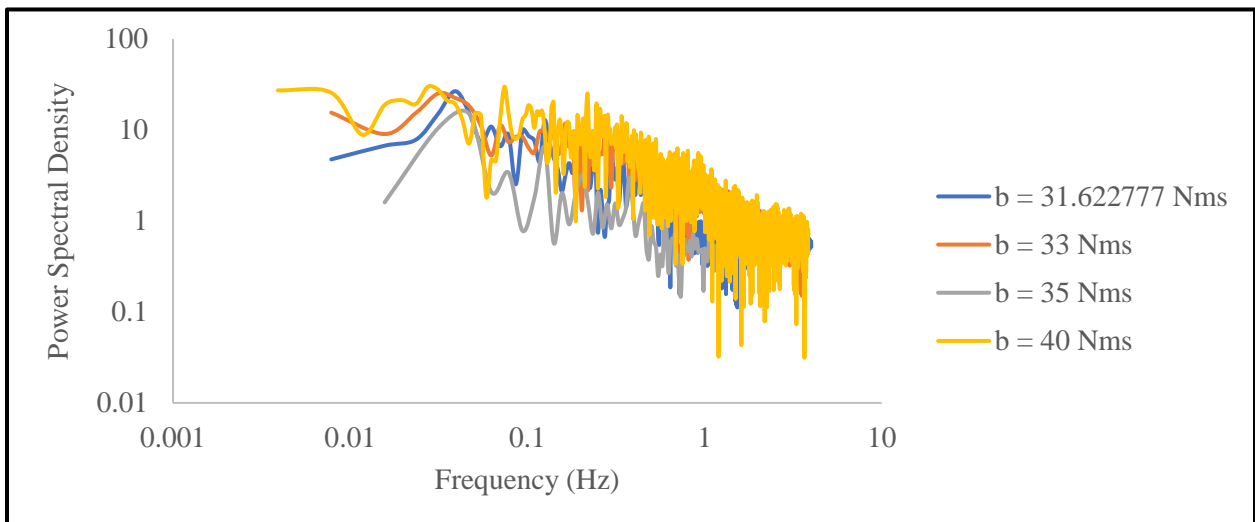
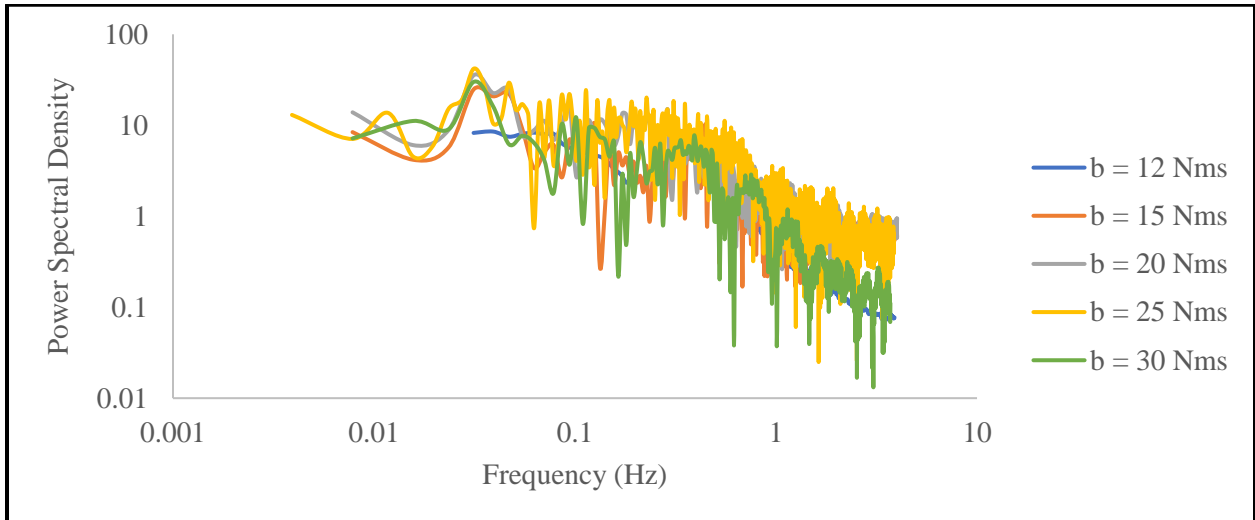
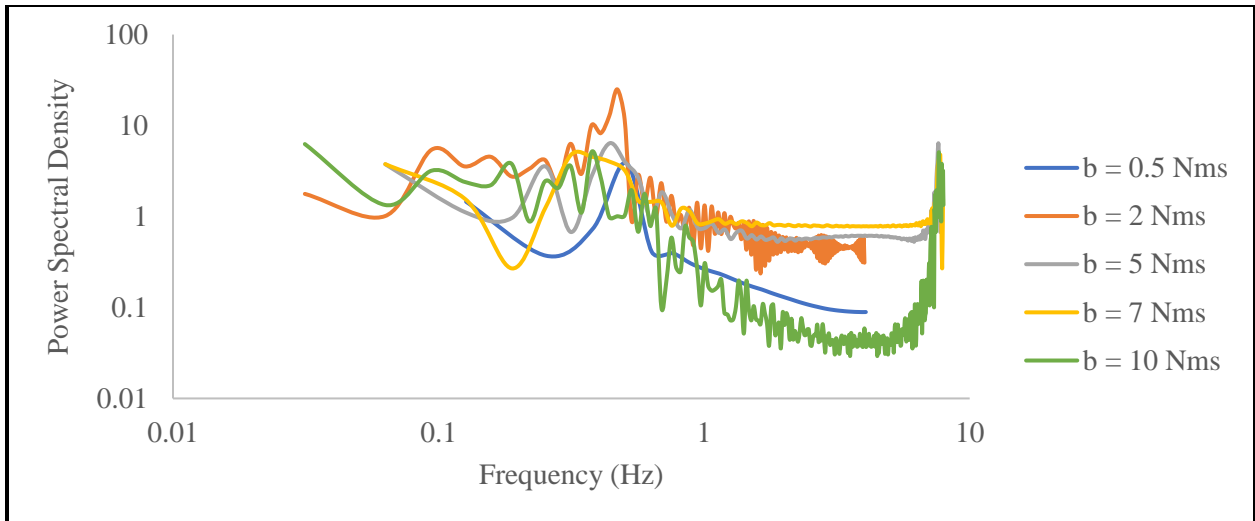


**Figure 40 FFT: Strouhal numbers for varying mass moment of inertia**

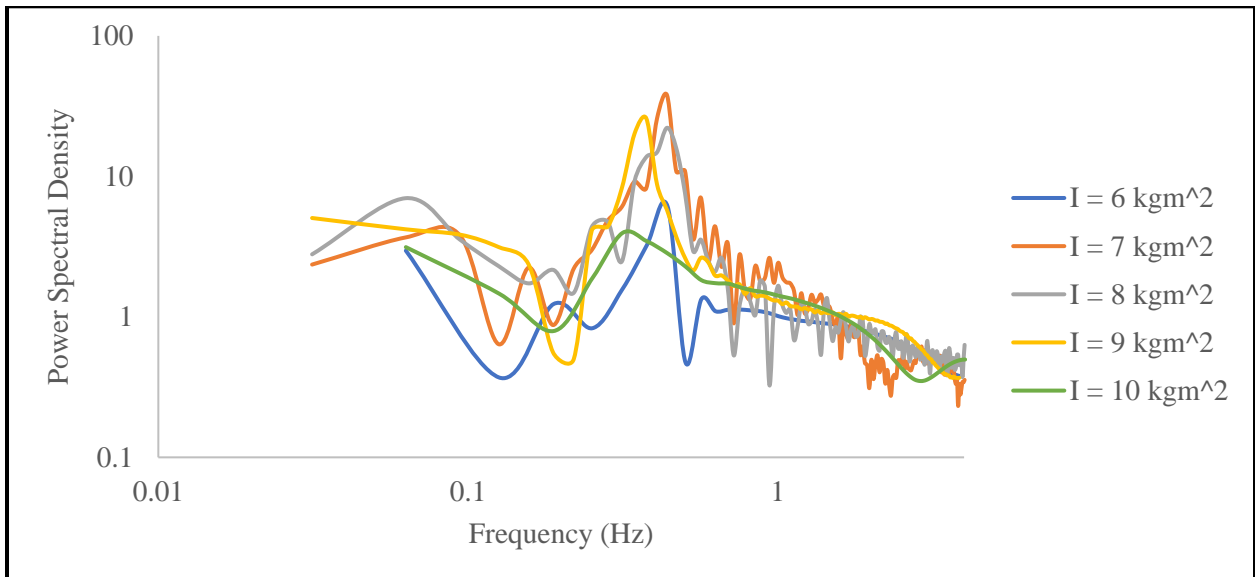
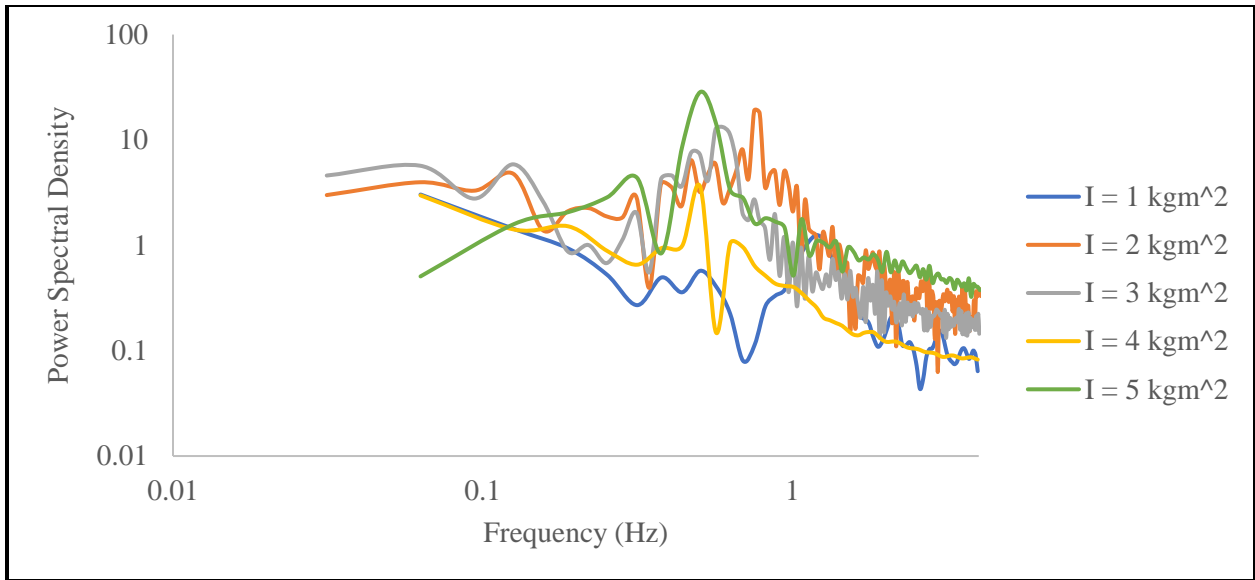
The damped frequencies were found from FFTs of the deflection angle recordings taken in each time step. In particular, the oscillations of the flat plate towards the end of the specimen's life are demonstrative of the damped frequency of the specimen. The locations of the peak shows that the damped frequency increases with torsional stiffness and decreases as damping coefficient and mass moment of inertia increase. These relationships match the theoretical relationships given by Equation 27.



**Figure 41 FFT: Damped frequency for varying torsional stiffness**



**Figure 42 FFT: Damped frequency for varying damping coefficient**



**Figure 43 FFT: Damped frequency for varying mass moment of inertia**

## REFERENCES

- [1] Morris, R. A. (1997). *Aeroelastic modeling and flutter control in aircraft with low aspect ratio composite wings* (Doctoral dissertation, Virginia Polytechnic Institute and State University, 1996). Ann Arbor, MI: UMI Microform. Retrieved May 23, 2019, from <http://ezproxy.library.unlv.edu>.
- [2] Gonzales, A. B. (2002). *A proposal for aircraft early accurate rapid flutter analysis* (Master's thesis, California State University, Long Beach, 2001). Ann Arbor, MI: ProQuest Information and Learning Company. Retrieved May 23, 2019, from <http://ezproxy.library.unlv.edu>.
- [3] Rosenhead, L. "The formation of vortices from a surface of discontinuity," *Proceedings of the Royal Society of London*, Vol. 134, No. 823, 1931, pp. 170-192. doi: 10.1098/rspa.1931.0189.
- [4] Chorin, A. J. (1973). Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, 57(4), 785–796. doi: 10.1017/s0022112073002016
- [5] Sarpkaya, T. (1975). An inviscid model of two-dimensional vortex shedding for transient and asymptotically steady separated flow over an inclined plate. *Journal of Fluid Mechanics*, 68(01), 109–130. doi: 10.1017/s0022112075000717
- [6] Sarpkaya, T., & Schoaff, R. L. (1979). Inviscid Model of Two-Dimensional Vortex Shedding by a Circular Cylinder. *AIAA Journal*, 17(11), 1193-1200. doi:10.2514/3.61300.
- [7] Walther, J. H., & Larsen, A. (1997). Two dimensional discrete vortex method for application to bluff body aerodynamics. *Journal of Wind Engineering and Industrial Aerodynamics*, 67-68, 183-193. doi:10.1016/s0167-6105(97)00072-x.
- [8] Lin, H., "Prediction of separated flows around pitching aerofoils using a discrete vortex method," Ph.D. Dissertation, University of Glasgow, Glasgow, 1997.
- [9] Anderson, J. D. (2011). *Fundamentals of Aerodynamics* (5th ed.). New York, NY: McGraw-Hill Education.
- [10] Vallentine, H. R. (1969). *Applied hydrodynamics* (SI ed.). London: Butterworths.
- [11] Agrawal, B. N., & Platzer, M. F. (2018). *Standard handbook for aerospace engineers* (2nd ed.). New York: McGraw-Hill Education.



- [12] Schlichting, H. (1979). *Boundary Layer Theory* (7th ed.). New York, NY: McGraw-Hill, 1979.
- [13] Chapra, S. C., & Canale, R. P. (1998). *Numerical Methods for Engineers* (2nd ed.). Boston: McGraw-Hill.
- [14] Hodges, D. H., & Pierce, G. A. (2011). *Introduction to Structural Dynamics and Aeroelasticity*. Cambridge University Press. Retrieved May 25, 2019, from <http://ebookcentral.proquest.com>.
- [15] FreeBASIC Development Team. (2016). *FreeBASIC*. Retrieved August 3, 2019, from <https://freebasic.net/>.
- [16] Que, A. (2017). Fast Fourier Transform for LibreOffice Calc. Retrieved November 20, 2019, from <http://fft4loc.drque.net/>.
- [17] Engineering ToolBox. (2003). *Air - Dynamic and Kinematic Viscosity*. Retrieved December 18, 2019, from [https://www.engineeringtoolbox.com/air-absolute-kinematic-viscosity-d\\_601.html](https://www.engineeringtoolbox.com/air-absolute-kinematic-viscosity-d_601.html).
- [18] Fatemi, A. *Fatigue from Variable Amplitude Loading*. Retrieved December 28, 2019, from [https://www.efatigue.com/training/Chapter\\_9.pdf](https://www.efatigue.com/training/Chapter_9.pdf).

## CURRICULUM VITAE

Graduate College  
University of Nevada, Las Vegas

Emma C. Chao

### Contact Information

4505 S Maryland Pkwy  
Howard R. Hughes College of Engineering  
University of Nevada, Las Vegas  
Las Vegas, NV 89154  
Email: echao97@gmail.com

### Education

Bachelor of Science in Engineering – Mechanical Engineering, 2019  
University of Nevada, Las Vegas

### Thesis Title

Discrete Vortex Modeling of Aerodynamic Flutter of Flat Plate with Damped Oscillations

### Thesis Examination Committee

Chairperson, Dr. William Culbreth  
Committee Member, Dr. Woosoon Yim  
Committee Member, Dr. Mohamed Trabia  
Graduate Faculty Representative, Dr. Evangelos Yfantis